GPW-SP: Gaussian Pancakes Watermarking with Salted Phase

Anonymous Authors

Abstract

We study private watermarking for autoregressive language models, where a provider embeds a key-dependent statistical signal during decoding and later verifies provenance using a secret key. We introduce *Gaussian Pancakes Watermarking with Salted Phase* (GPW-SP), a lightweight sampling-time method that biases token selection via a smooth periodic function of token embedding projections onto a secret direction. To reduce predictability, GPW-SP incorporates a context-dependent salted phase derived from a keyed pseudorandom function, producing position-varying preferences without modifying model weights.

GPW-SP is implemented as a modular logits-bias component compatible with standard decoding schemes and evaluated using calibrated hypothesis testing at low false positive rates. In a GPT-2 pilot on C4 prompts, phase salting substantially improves clean detectability over an unsalted variant (AUC 0.964 vs. 0.878, TPR@1Under meaning-preserving edits, detectability degrades smoothly, remaining robust to moderate synonym replacement and word deletion, while stronger paraphrasing and text mixing significantly reduce detection power.

Overall, GPW-SP provides an efficient geometry-based watermark baseline and a reproducible evaluation protocol aligned with recent watermarking benchmarks, reporting robustness curves under transformations.

1 Introduction

While Large Language Models (LLMs) enhance productivity, they also facilitate spam, plagiarism, and disinformation. This necessitates detection of machine-written text and/or attribution of that text to a specific model. Unlike brittle post-hoc classifiers that fail under paraphrasing or domain shift [13], *watermarking* embeds a signal during generation by modifying sampling behavior . With a secret key, a verifier can detect or or attribute this signal to a source without model retraining, often via a modular logits processor.

Designing effective watermarks is challenging because the channel is adversarial and stochastic. Text is easily edited through paraphrasing or truncation, and decoding methods like nucleus sampling introduce randomness. Consequently, watermarking creates a *statistical bias* rather than a deterministic signature, framing detection as a hypothesis test. This creates a fundamental tension between *utility* (text quality), *detectability* (true positive rate), and *robustness* (resistance to transformation). Recent work has also emphasized that watermark conclusions can vary significantly across prompt sets, decoding parameters, and transformation attacks. This motivates benchmark-driven evaluation: reporting not only "clean" detectability, but robustness under realistic editing pipelines and model-based attacks, ideally as curves over attack strength using standardized suites and tooling (e.g., WaterBench, WaterPark, MarkLLM).

Our approach: Gaussian Pancakes Watermarking with Salted Phase (GPW-SP). We propose GPW-SP, a watermark tied to the geometry of the model's token embedding space. It projects embeddings onto a secret keyed direction w and applies a periodic cosine score to bias

logits. To ensure unpredictability, we use a *salted phase*—a context-dependent shift derived from a pseudorandom function.

Our contributions : (i) GPW-SP, a simple embedding-geometry watermark that requires no retraining and integrates as a logits-bias module; (ii) a private detection procedure framed as a calibrated hypothesis test with user-chosen false positive rates; (iii) optional extensions for semantic coupling (SR) and payload encoding; and (iv) an evaluation protocol with robustness curves under common text transformations, plus practical stress tests such as deletion and splicing/mixing.

2 Background and Problem Setup

2.1 Watermarking for autoregressive LMs (private provenance)

We consider an autoregressive language model that, given a prefix $x_{1:t-1}$, outputs logits $\ell_t \in \mathbb{R}^{|\mathcal{V}|}$ and induces a next-token distribution $p_t = \mathrm{softmax}(\ell_t/\tau)$ (temperature τ). A *watermarked sampler* modifies *inference-time* sampling—most commonly by adding a key-dependent bias $b_t \in \mathbb{R}^{|\mathcal{V}|}$ to the logits—without changing model weights:

$$\tilde{\ell}_t = \ell_t + b_t, \qquad \tilde{p}_t = \mathrm{softmax}(\tilde{\ell}_t/\tau).$$

The verifier, given the secret key k, tests whether a candidate text contains statistical evidence that it was sampled from $\tilde{p}_t$ rather than p_t [4].

We focus on **private detection** in a Kerckhoffs-style setting: the watermarking algorithm is public, but the key is secret. This matches common provenance deployments where only the provider (or an authorized auditor) can verify. Because decoding is stochastic, watermarking induces a *bias* rather than a deterministic signature, so detection is naturally framed as hypothesis testing:

$$H_0: \text{ text is unwatermarked} \quad \text{vs.} \quad H_1: \text{ text is watermarked},$$

with calibration to achieve a target false positive rate (FPR), typically extremely low. Accordingly, we report detection power as TPR at fixed low FPR (e.g., 1% and below), along with threshold-free metrics such as AUC where appropriate [4].

2.2 Threat model and attack surfaces

We distinguish *benign* post-processing (copy-editing, formatting, truncation) from *adversarial* post-processing aimed at watermark removal while preserving meaning and fluency. Our primary focus is the realistic and widely studied **black-box / key-hidden** setting: the attacker may know the watermarking algorithm and can transform text or query the model, but does not know k. This is analogous to the security guarantees of **undetectable backdoors** [3], which demonstrate that it is possible to embed statistical signals that are computationally indistinguishable from the natural distribution to any adversary lacking the verification key.

We organize transformations into a parameterized family $\mathcal{A}(\cdot;\gamma)$ where γ controls *attack strength*. Following standard robustness practice, we evaluate robustness as a *curve* over γ, rather than a single point [11, 6, 13]. We consider the following commonly used attack families:

- **Lexical edits** (e.g., synonym substitution, word deletion, minor phrase edits), parameterized by replacement/deletion rate.

- **Model-based paraphrasing** (rewrite via another LLM or paraphraser), parameterized by decoding strength and/or rewrite aggressiveness.
- **Summarization / expansion** (compress or elaborate while preserving meaning), parameterized by target length ratio.
- **Translation / back-translation** (cross-lingual perturbations), parameterized by language and round-trip pipeline.
- **Truncation / cropping / concatenation** (drop or splice spans), parameterized by keep ratio and splice pattern.

We also treat **model extraction via distillation** as a first-class threat model: an attacker trains a student on mixtures of watermarked/unwatermarked teacher outputs (optionally after paraphrasing the training data) and then generates from the student [9].

2.3 Evaluation goals and reporting protocol

A watermark is judged along three coupled axes: **utility** (fluency and diversity), **detectability** (high TPR at very low FPR), and **robustness** (resistance to the attacks)[13]. These goals compete: increasing watermark strength often improves detectability but may introduce quality costs or statistical artifacts, while robustness to stronger paraphrasing typically requires more structure, redundancy, or semantic coupling [10].

To reduce evaluation variance and enable apples-to-apples comparison, we follow benchmark practice: (i) match prompts and decoding settings across methods, (ii) calibrate detection thresholds on *null* (unwatermarked) samples at a target FPR, and (iii) report robustness as curves over attack strength [11, 8, 6].

Robustness–utility trade-offs and unified robustness platforms. Recent work formalizes "no free lunch" trade-offs: robustness, detectability, and quality cannot all be jointly optimized, and strong robustness to powerful paraphrasing typically comes with noticeable costs [10]. In parallel, unified robustness platforms such as WaterPark systematize attack suites and enable comprehensive, comparable stress testing across watermarkers [6].

Distillation and watermark inheritance. Model extraction motivates studying whether watermark evidence persists when teacher outputs are reused as training data. Recent work shows inheritance can hold in some settings but can be weakened substantially by paraphrasing training data or by post-distillation neutralization strategies, motivating explicit inheritance-curve reporting [9].

2.4 Positioning and baseline set for experiments

Our method is a sampling-time watermark (no retraining), but it changes the *carrier* from discrete token partitions to a smooth, key-dependent structure in embedding geometry with a context-dependent salted phase. This is intended to reduce brittleness under meaning-preserving edits while maintaining low deployment overhead. To support clear empirical positioning, our primary comparison set includes: **KGW** (vocabulary partition) [4], **Unigram** (provable robustness) [12], **RDF / distortion-free** (distribution-preserving) [5], and **SemStamp/SIR** (representative semantic schemes family) [7]. We evaluate all methods under matched prompts/decoding, robustness curves

over standardized edits, and inheritance curves under distillation, using benchmark/toolkit-aligned pipelines [11, 8, 6].

3 Method

We describe our watermarking sampler as a sequence of increasingly structured designs. We start from a naive "Gaussian pancakes" sampler, then add a salted phase for unpredictability. Throughout, we focus on **private detection**: a verifier with a secret key can test a text, while an attacker who does not know the key should not be able to reliably predict or remove the watermark without substantially rewriting the text.

3.1 Problem setup and notation

We consider an autoregressive language model with vocabulary $\mathcal{V}$. At generation step t, given a prefix $x_{1:t-1}$, the model outputs logits $\ell_t \in \mathbb{R}^{|\mathcal{V}|}$, and a base distribution $p_t(i) = \mathrm{softmax}(\ell_t/\tau)_i$, with temperature $\tau > 0$. Let $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ be the model's token embedding matrix, where token i has embedding $e_i \in \mathbb{R}^d$.

A watermarking sampler produces a modified distribution q_t that is close to p_t (to preserve quality) but biased in a way that can be tested with the key. We implement watermarking as an *additive logit bias*:

$$\ell'_t(i) = \ell_t(i) + b_t(i), \qquad q_t(i) \propto \exp(\ell'_t(i)/\tau).$$

We choose $b_t(i)$ so that (i) the text remains natural, and (ii) a verifier can accumulate evidence across tokens.

3.2 Stage 1: Gaussian Pancakes Watermarking (GPW)

Keyed secret direction. Given a secret key K, we derive a pseudorandom unit vector $w \in \mathbb{R}^d$:

$$g \leftarrow \mathcal{N}(0, I_d) \text{ seeded by } K, \qquad w = \frac{g}{\|g\|}.$$

Intuition: w defines a hidden axis in embedding space known only to the verifier.

Token projection. For each vocabulary token i, we compute and cache its projection on the secret axis:

$$s_i = \langle e_i, w \rangle.$$

This is computed once and reused for all generations.

Periodic "pancake" score. We define a periodic score on the projection coordinate:

$$g(i) = \cos(\omega s_i),$$

where $\omega > 0$ controls frequency. Large ω yields many thin alternating bands; smaller ω yields fewer thick bands. We refer to these bands as "pancakes" because the cosine creates parallel high-score slices along w. This geometry is inspired by the hard-to-learn "parallel pancakes" distributions constructed in statistical query lower bounds [2].

Logit bias and sampling. Given strength $\alpha \geq 0$, we bias logits toward high-score tokens:

$$\ell'_t(i) = \ell_t(i) + \alpha\, g(i) \quad \Rightarrow \quad q_t(i) \propto p_t(i) \exp(\alpha g(i)).$$

We then sample from q_t using the same decoding settings as usual (temperature, top-k, top-p, etc.).

Why the naive design works (and where it fails). Under the base model distribution, the cosine score behaves like noise that averages to near zero over many tokens (especially when prompts vary). Under the biased distribution, tokens with higher $g(i)$ become more likely, so the sum of scores tends to be positive. However, the naive scheme has two weaknesses: (1) the preference pattern is *static* across positions, which can be exploited if an attacker can estimate the pattern from many samples; and (2) because it is static, it can create small but consistent distortions that may be easier to wash out with paraphrasing. These motivate the next stage.

3.3 Stage 2: Salted-phase Gaussian Pancakes (GPW-SP)

To prevent a static and predictable pattern, we make the cosine *phase* depend on the local context through the secret key.

Salted phase from context. At each time step t, we compute a phase $\phi_t \in [0, 2\pi)$:

$$\phi_t = 2\pi \cdot \mathrm{Unif}\big(\mathrm{PRF}_K(\mathrm{ctx}_t)\big),$$

where $\mathrm{PRF}_K(\cdot)$ is a keyed pseudorandom function and ctx_t is a short fingerprint of context. In practice, ctx_t can be: (i) the previous token x_{t-1}, (ii) a hash of the last n tokens, or (iii) a rolling hash of the prefix (to reduce collisions). The output of the PRF is converted to a uniform real in $[0, 1)$ and then scaled to $[0, 2\pi)$.

Salted pancake score. We now define a time-varying score:

$$g_t(i) = \cos(\omega s_i + \phi_t).$$

This makes the "preferred" bands move with context, so the watermark no longer corresponds to a fixed global partition.

Watermarked sampling distribution. We bias logits using the salted score:

$$\ell'_t(i) = \ell_t(i) + \alpha\, g_t(i), \qquad q_t(i) \propto p_t(i) \exp(\alpha g_t(i)).$$

Implementation is still simple: cache s_i, compute ϕ_t each step, evaluate $g_t(i)$, and add $\alpha g_t(i)$ to logits.

Optional efficiency restriction (top-k / nucleus). To reduce computation and limit distortion, we can apply the bias only to a candidate set $\mathcal{C}_t$ (e.g., top-k tokens under ℓ_t or the nucleus set under p_t):

$$\ell'_t(i) = \ell_t(i) + \alpha\, g_t(i) \cdot \mathbb{I}[i \in \mathcal{C}_t].$$

This preserves the same structure while improving speed and keeping the watermark inside the model's likely choices.

Detection statistic. Given a candidate text $x_{1:T}$ and key K, the verifier:

1. Reconstructs w from K.
2. For each token x_t, computes ϕ_t from ctx_t and K.
3. Computes per-token alignment
$$a_t = \cos(\omega s_{x_t} + \phi_t).$$
4. Aggregates
$$S(x_{1:T}) = \sum_{t=1}^{T} a_t.$$

Hypothesis test and calibration. We test

$$H_0 : \text{text is unwatermarked} \quad \text{vs.} \quad H_1 : \text{text is generated by GPW-SP under key } K.$$

Under H_0, S is approximately centered near 0 and concentrates as T grows. Under H_1, the bias makes $\mathbb{E}[a_t] > 0$, so S tends to be larger. In practice we normalize

$$Z(x_{1:T}) = \frac{S(x_{1:T}) - \mu_0(T)}{\sigma_0(T)},$$

where $\mu_0(T), \sigma_0(T)$ are estimated from a held-out corpus of human or non-watermarked text under the same tokenizer and preprocessing. We then choose a threshold $z_\star$ such that $\Pr_{H_0}(Z \geq z_\star)$ matches a target false positive rate. As an alternative view, we also report *random-key p-values*: we evaluate S under many random keys and compute the fraction that produce a score at least as large as the observed one.

3.4 Payload encoding (optional extension)

GPW-SP can be used for either *presence detection* (one-bit: watermarked or not) or for embedding a short payload.

Segment-wise phase shifts. Let a payload be $m \in \{0,1\}^k$. We encode it (optionally) with an error-correcting code (ECC) to obtain a codeword $c \in \{0,1\}^L$. We then divide generation into segments of length R tokens. For segment index j, we set

$$\Delta_j = \pi\, c_j,$$

and use

$$g_t(i) = \cos(\omega s_i + \phi_t + \Delta_j), \qquad \text{for } t \in \text{segment } j.$$

This makes the watermark statistic favor one of two phase hypotheses per segment. The verifier tests both hypotheses and decodes the resulting bit sequence.

Decoding. Given a text, the verifier computes for each segment j two correlation scores (for $\Delta_j = 0$ and $\Delta_j = \pi$), selects the higher-scoring hypothesis as the recovered bit $\hat{c}_j$, then ECC-decodes $\hat{c}$ to obtain $\hat{m}$. We can also add repetition (use multiple segments per bit) to increase robustness to deletions and truncation.

3.5 Summary of parameters

Our sampler uses a small set of interpretable parameters:

- α (strength): larger values increase detectability but may affect quality.
- ω (frequency): controls pancake "thinness" and the granularity of the periodic pattern.
- Choice of ctx_t for salted phase: trades off unpredictability vs. stability.
- Optional restriction set $\mathcal{C}_t$ (top-k/top-p): reduces compute and limits distortion.

3.6 What we implement and evaluate

In experiments, we evaluate: (i) the base GPW sampler (static phase) as a simple baseline; and (ii) GPW-SP (salted phase) as our main presence-detection method.

4 Method

We describe our watermarking sampler as a sequence of increasingly structured designs. We start from a naive "Gaussian pancakes" sampler, then add a salted phase for unpredictability. Throughout, we focus on **private detection**: a verifier with a secret key can test a text, while an attacker who does not know the key should not be able to reliably predict or remove the watermark without substantially rewriting the text.

4.1 Problem setup and notation

We consider an autoregressive language model with vocabulary $\mathcal{V}$. At generation step t, given a prefix $x_{1:t-1}$, the model outputs logits

$$\ell_t \in \mathbb{R}^{|\mathcal{V}|},$$

and a base distribution

$$p_t(i) = \mathrm{softmax}(\ell_t/\tau)_i,$$

with temperature $\tau > 0$. Let $E \in \mathbb{R}^{|\mathcal{V}|\times d}$ be the model's token embedding matrix, where token i has embedding $e_i \in \mathbb{R}^d$.

A watermarking sampler produces a modified distribution q_t that is close to p_t (to preserve quality) but biased in a way that can be tested with the key. We implement watermarking as an *additive logit bias*:

$$\ell'_t(i) = \ell_t(i) + b_t(i), \qquad q_t(i) \propto \exp(\ell'_t(i)/\tau).$$

We choose $b_t(i)$ so that (i) the text remains natural, and (ii) a verifier can accumulate evidence across tokens.

4.2 Stage 1: Gaussian Pancakes Watermarking (GPW)

Keyed secret direction. Given a secret key K, we derive a pseudorandom unit vector $w \in \mathbb{R}^d$:

$$g \leftarrow \mathcal{N}(0, I_d) \text{ seeded by } K, \qquad w = \frac{g}{\|g\|}.$$

Intuition: w defines a hidden axis in embedding space known only to the verifier.

Token projection. For each vocabulary token i, we compute and cache its projection on the secret axis:

$$s_i = \langle e_i, w \rangle.$$

This is computed once and reused for all generations.

Periodic "pancake" score. We define a periodic score on the projection coordinate:

$$g(i) = \cos(\omega s_i),$$

where $\omega > 0$ controls frequency. Large ω yields many thin alternating bands; smaller ω yields fewer thick bands. We refer to these bands as "pancakes" because the cosine creates parallel high-score slices along w.

Logit bias and sampling. Given strength $\alpha \geq 0$, we bias logits toward high-score tokens:

$$\ell'_t(i) = \ell_t(i) + \alpha\, g(i) \quad \Rightarrow \quad q_t(i) \propto p_t(i) \exp(\alpha g(i)).$$

We then sample from q_t using the same decoding settings as usual (temperature, top-k, top-p, etc.).

Why the naive design works (and where it fails). Under the base model distribution, the cosine score behaves like noise that averages to near zero over many tokens (especially when prompts vary). Under the biased distribution, tokens with higher $g(i)$ become more likely, so the sum of scores tends to be positive. However, the naive scheme has two weaknesses: (1) the preference pattern is *static* across positions, which can be exploited if an attacker can estimate the pattern from many samples; and (2) because it is static, it can create small but consistent distortions that may be easier to wash out with paraphrasing. These motivate the next stage.

4.3 Stage 2: Salted-phase Gaussian Pancakes (GPW-SP)

To prevent a static and predictable pattern, we make the cosine *phase* depend on the local context through the secret key.

Salted phase from context. At each time step t, we compute a phase $\phi_t \in [0, 2\pi)$:

$$\phi_t = 2\pi \cdot \mathrm{Unif}\big(\mathrm{PRF}_K(\mathrm{ctx}_t)\big),$$

where $\mathrm{PRF}_K(\cdot)$ is a keyed pseudorandom function and ctx_t is a short fingerprint of context. In practice, ctx_t can be: (i) the previous token x_{t-1}, (ii) a hash of the last n tokens, or (iii) a rolling hash of the prefix (to reduce collisions). The output of the PRF is converted to a uniform real in $[0, 1)$ and then scaled to $[0, 2\pi)$.

Salted pancake score. We now define a time-varying score:

$$g_t(i) = \cos(\omega s_i + \phi_t).$$

This makes the "preferred" bands move with context, so the watermark no longer corresponds to a fixed global partition.

Watermarked sampling distribution. We bias logits using the salted score:

$$\ell'_t(i) = \ell_t(i) + \alpha\, g_t(i), \qquad q_t(i) \propto p_t(i) \exp(\alpha g_t(i)).$$

Implementation is still simple: cache s_i, compute ϕ_t each step, evaluate $g_t(i)$, and add $\alpha g_t(i)$ to logits.

Optional efficiency restriction (top-k / nucleus). To reduce computation and limit distortion, we can apply the bias only to a candidate set $\mathcal{C}_t$ (e.g., top-k tokens under ℓ_t or the nucleus set under p_t):

$$\ell'_t(i) = \ell_t(i) + \alpha\, g_t(i) \cdot \mathbb{I}[i \in \mathcal{C}_t].$$

This preserves the same structure while improving speed and keeping the watermark inside the model's likely choices.

Detection statistic. Given a candidate text $x_{1:T}$ and key K, the verifier:

1. Reconstructs w from K.
2. For each token x_t, computes ϕ_t from ctx_t and K.
3. Computes per-token alignment
$$a_t = \cos(\omega s_{x_t} + \phi_t).$$
4. Aggregates
$$S(x_{1:T}) = \sum_{t=1}^{T} a_t.$$

Hypothesis test and calibration. We test

$$H_0 : \text{text is unwatermarked} \quad \text{vs.} \quad H_1 : \text{text is generated by GPW-SP under key } K.$$

Under H_0, S is approximately centered near 0 and concentrates as T grows. Under H_1, the bias makes $\mathbb{E}[a_t] > 0$, so S tends to be larger. In practice we normalize

$$Z(x_{1:T}) = \frac{S(x_{1:T}) - \mu_0(T)}{\sigma_0(T)},$$

where $\mu_0(T), \sigma_0(T)$ are estimated from a held-out corpus of human or non-watermarked text under the same tokenizer and preprocessing. We then choose a threshold $z_\star$ such that $\Pr_{H_0}(Z \geq z_\star)$ matches a target false positive rate. As an alternative view, we also report *random-key p-values*: we evaluate S under many random keys and compute the fraction that produce a score at least as large as the observed one.

4.4 Payload encoding (optional extension)

GPW-SP can be used for either *presence detection* (one-bit: watermarked or not) or for embedding a short payload.

Segment-wise phase shifts. Let a payload be $m \in \{0,1\}^k$. We encode it (optionally) with an error-correcting code (ECC) to obtain a codeword $c \in \{0,1\}^L$. We then divide generation into segments of length R tokens. For segment index j, we set

$$\Delta_j = \pi\, c_j,$$

and use

$$g_t(i) = \cos(\omega s_i + \phi_t + \Delta_j), \qquad \text{for } t \in \text{segment } j.$$

This makes the watermark statistic favor one of two phase hypotheses per segment. The verifier tests both hypotheses and decodes the resulting bit sequence.

Decoding. Given a text, the verifier computes for each segment j two correlation scores (for $\Delta_j = 0$ and $\Delta_j = \pi$), selects the higher-scoring hypothesis as the recovered bit $\hat{c}_j$, then ECC-decodes $\hat{c}$ to obtain $\hat{m}$. We can also add repetition (use multiple segments per bit) to increase robustness to deletions and truncation.

4.5 Summary of parameters

Our sampler uses a small set of interpretable parameters:

- α (strength): larger values increase detectability but may affect quality.
- ω (frequency): controls pancake "thinness" and the granularity of the periodic pattern.
- Choice of ctx_t for salted phase: trades off unpredictability vs. stability.
- Optional restriction set $\mathcal{C}_t$ (top-k/top-p): reduces compute and limits distortion.

4.6 What we implement and evaluate

In experiments, we evaluate: (i) the base GPW sampler (static phase) as a simple baseline; and (ii) GPW-SP (salted phase) as our main presence-detection method.

5 Experimental Evaluation

This section reports our current experimental results and the evaluation suite we will extend. We structure evaluation to (i) validate clean detectability and calibration, (ii) measure robustness under realistic text transformations as *curves* over attack strength, and (iii) study sensitivity to decoding and design choices (salting). Our main metric is TPR at a target false positive rate (FPR), obtained by calibrating a threshold on a held-out null set under the same attack. We also report ROC/AUC.

5.1 Tooling and benchmarks (context)

We align our reporting with recent evaluation practice and toolkits (e.g., MarkLLM, WaterBench, WaterPark), and we will integrate our implementation into these frameworks for direct baseline comparisons in the final version.

Table 1: Pilot results on GPT-2 with C4 prompts (40 prompts, $\tau = 0.9$, top-$k = 50$, top-$p = 0.95$, 120 tokens). Threshold is calibrated to FPR=1% on the corresponding null set for each row.

Variant	Attack	AUC	TPR@1%FPR	Median(WM)	Median(null)
GPW (no salt)	clean	0.878	0.275	98.81	85.07
GPW-SP	clean	0.964	0.700	29.01	1.18

5.2 Models, prompts, and decoding settings

Base model and prompts. Our current pilot uses GPT-2 (`gpt2`) and short prompts sampled from C4. Unless stated otherwise, each condition uses 40 prompts for variant comparisons, with smaller pilot sizes (15–20 prompts) for hyperparameter and decoding sensitivity sweeps due to compute constraints.

Decoding. Our default decoding is temperature sampling with top-k and top-p filtering: $\tau = 0.9$, top-$k = 50$, top-$p = 0.95$, and a maximum of 120 new tokens.

Watermark configuration. Unless stated otherwise, we use GPW-SP with $(\alpha, \omega) = (1.2, 10.0)$ and context salting enabled.

5.3 Clean detectability and salt ablation

We first validate that GPW-SP produces a reliable keyed statistical signal on unedited outputs and that phase salting improves calibration.

Salting substantially reduces the null score magnitude while preserving strong positive correlation under watermarking, enabling more reliable calibration at low FPR. The unsalted variant exhibits large scores for both watermarked and null text, consistent with a predictable static pattern that shifts the null distribution.

5.4 Robustness to synonym replacement and paraphrasing

We evaluate robustness under two meaning-preserving transformation families: (i) lexical synonym replacement (WordNet-based) and (ii) full paraphrasing (T5 paraphraser).

Fixed-strength attacks (40 prompts). At 50% synonym replacement, GPW-SP remains detectable but degrades notably: AUC 0.903 and TPR@1%FPR 0.275. Under T5 paraphrasing (beam 5), detectability further drops: AUC 0.729 and TPR@1%FPR 0.375 (Table 1).

5.5 Decoding sensitivity (pilot)

We next test whether detectability is stable under common decoding changes (15 prompts). GPW-SP remains detectable across settings, but performance varies meaningfully. For example, increasing temperature to $\tau = 1.1$ improves separation (AUC 1.00, TPR@1%FPR 1.00), while lowering to $\tau = 0.7$ reduces low-FPR performance (AUC 0.929, TPR@1%FPR 0.467). This suggests detectability depends on *decoding entropy*: when the model has less freedom in token choice, watermark bias has less opportunity to accumulate.

5.6 Deletion and splicing/mixing stress tests

Beyond synonym/paraphrase, we include two stress tests that occur in real pipelines: word deletion and concatenation-based mixing.

Word deletion (20 prompts). Deleting 20% of words reduces detectability but preserves meaningful signal (AUC 0.970, TPR@1%FPR 0.60). At 40% deletion, detectability degrades further (AUC 0.863, TPR@1%FPR 0.45). This is consistent with evidence accumulation: fewer original tokens remain to contribute to the detection statistic.

Mixing / concatenation (20 prompts). We mix watermarked and non-watermarked text by concatenating a watermarked prefix with a non-watermarked suffix, varying the watermarked fraction. Detectability collapses quickly as the watermarked fraction decreases: at fraction 0.75, AUC is 0.833 but TPR@1%FPR is only 0.05; at 0.5 and 0.25, detection fails (AUC ≤ 0.21, TPR@1%FPR $= 0$). This highlights a fundamental vulnerability of single-statistic presence tests: if only a minority of tokens carry the watermark, the aggregate evidence can be dominated by null text. This motivates segment-level testing and payload-style decoding as a potential defense, which we plan to study next.

Table 2 summarizes our pilot results across clean detectability, robustness curves, and stress tests (deletion and mixing).

6 Discussion

What the results show. Our pilot evaluation supports three main conclusions. First, *phase salting improves calibration*: compared to an unsalted variant, GPW-SP reduces null statistic magnitude while preserving a positive shift under watermarking, enabling stronger low-FPR detection. Second, *robustness is transformation-dependent*: detectability degrades smoothly under synonym replacement and word deletion, but drops more sharply under full paraphrasing and aggressive truncation, consistent with prior observations on token-level statistical carriers [6, 10]. Third, *detectability depends on decoding entropy*: changes in temperature and filtering substantially affect low-FPR performance, suggesting watermark evidence accumulates most reliably when decoding retains sufficient stochasticity.

Mixing as a failure mode. Concatenation of watermarked and non-watermarked text rapidly degrades detection, even when the watermarked fraction is nontrivial. This reflects a general limitation of global hypothesis tests under evidence dilution rather than a GPW-SP–specific weakness. A natural mitigation is to move from a single global statistic to localized (segment-level) testing or payload-style decoding with redundancy, which can recover signals in mixed-source documents.

Security model and limitations. We focus on private detection, where the algorithm is public but the secret key remains hidden, matching many provider-side provenance settings. Key compromise enables targeted removal but is not unique to our approach and is best addressed through operational controls such as key management and rotation. Publicly verifiable watermarking involves fundamentally different threat models and trade-offs [1]. Our results are limited to pilot-scale experiments on GPT-2 with moderate prompt counts and lightweight attacks; stronger paraphrasing, translation-based edits, white-box removal, and comprehensive utility evaluation remain future work.

Table 2: Summary of pilot results across attacks and settings. We report ROC/AUC and TPR at a calibrated threshold achieving FPR=1% on the corresponding (attacked) null set for that row. Medians are for the detection statistic on watermarked vs. null texts. Sample size n varies across sub-suites due to compute constraints and is shown per row.

Variant	Attack	Strength	n	AUC	TPR@1%FPR	Median(WM)	Median(null)
Core ablations (GPT-2, C4 prompts, $\tau = 0.9$, top-k=50, top-p=0.95, 120 tokens; $n = 40$)							
GPW (no salt)	clean	–	40	0.878	0.275	98.81	85.07
GPW (no salt)	synonym_replace	0.5	40	0.804	0.125	94.17	86.06
GPW (no salt)	paraphrase_t5	beam5	40	0.609	0.075	28.70	23.68
GPW-SP	clean	–	40	0.964	0.700	29.01	1.18
GPW-SP	synonym_replace	0.5	40	0.903	0.275	19.66	1.27
GPW-SP	paraphrase_t5	beam5	40	0.729	0.375	7.87	1.12
Synonym replacement curve (GPT-2; $n = 40$)							
GPW-SP	synonym_replace	0.0	40	0.964	0.700	29.01	1.18
GPW-SP	synonym_replace	0.3	40	0.939	0.475	23.00	-1.43
GPW-SP	synonym_replace	0.5	40	0.903	0.275	19.66	1.27
Paraphrase robustness (T5; $n = 30$)							
GPW-SP	paraphrase_t5	beam5	30	0.727	0.333	5.92	0.86
GPW-SP	paraphrase_t5	beam1	30	0.706	0.467	8.45	0.62
Truncation (prefix keep ratio; GPW-SP; $n = 40$)							
GPW-SP	truncate_prefix	1.00	40	0.962	0.675	30.34	4.76
GPW-SP	truncate_prefix	0.50	40	0.868	0.700	13.41	2.32
GPW-SP	truncate_prefix	0.25	40	0.675	0.325	4.72	2.20
Length scaling (GPW-SP; $n = 60$)							
GPW-SP	clean	L=40	60	0.875	0.283	9.43	1.79
GPW-SP	clean	L=120	60	0.960	0.733	29.29	1.63
GPW-SP	clean	L=200	60	0.968	0.783	47.70	2.20
Decoding sensitivity (GPW-SP; $n = 15$; 120 tokens)							
GPW-SP	clean	τ=0.7	15	0.929	0.467	32.97	-1.67
GPW-SP	clean	τ=0.9 (base)	15	0.991	0.933	32.64	3.72
GPW-SP	clean	τ=1.1	15	1.000	1.000	32.31	2.06
Deletion (GPW-SP; $n = 20$)							
GPW-SP	delete	0.0	20	0.995	0.900	33.52	8.20
GPW-SP	delete	0.2	20	0.970	0.600	20.95	6.38
GPW-SP	delete	0.4	20	0.863	0.450	15.69	4.06
Mixing / concatenation (GPW-SP; $n = 20$; watermarked prefix fraction)							
GPW-SP	mix_concat	1.00	20	0.990	0.800	32.79	8.63
GPW-SP	mix_concat	0.75	20	0.833	0.050	25.87	11.75
GPW-SP	mix_concat	0.50	20	0.210	0.000	17.65	25.04
GPW-SP	mix_concat	0.25	20	0.045	0.000	10.86	29.70

7 Conclusion

We introduced GPW-SP, a private watermarking method for autoregressive language models that biases decoding using a smooth periodic function of embedding-space projections onto a secret direction, with a context-dependent salted phase. The method is lightweight, requires no retraining, and integrates as a modular logits-bias compatible with standard decoding schemes. Detection is framed as calibrated hypothesis testing at user-specified low false positive rates, with robustness evaluated as curves over attack strength.

Pilot experiments confirm that phase salting substantially improves calibration and low-FPR detectability relative to an unsalted baseline, while robustness degrades smoothly under moderate lexical edits and more sharply under strong paraphrasing, truncation, and text mixing. Next steps include adding semantic coupling, scaling to stronger models, and evaluating against established baselines within standardized benchmarking frameworks. More broadly, our findings suggests that

in LLM watermarking: detectability, robustness, and utility are inherently coupled, and credible evaluation requires careful calibration and transparent reporting under realistic transformations.

References

[1] M. Christ, T. Gunn, A. Ilyas, and N. Carlini. Undetectable watermarks for language models. 2023. URL https://arxiv.org/abs/2306.09194.

[2] I. Diakonikolas, D. M. Kane, and A. Stewart. Statistical query lower bounds for robust estimation of high-dimensional gaussians and gaussian mixtures, 2017. URL https://arxiv.org/abs/1611.03473.

[3] S. Goldwasser, M. P. Kim, V. Vaikuntanathan, and O. Zamir. Planting undetectable backdoors in machine learning models, 2024. URL https://arxiv.org/abs/2204.06974.

[4] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. URL https://arxiv.org/abs/2301.10226.

[5] R. Kuditipudi, J. Thickstun, T. Hashimoto, P. Liang, and T. Ma. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=YyWbA5R8gx.

[6] J. Liang, Z. Wang, S. Hong, S. Ji, and T. Wang. Watermark under fire: A robustness evaluation of LLM watermarking. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21050–21074, Suzhou, China, Nov. 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1148. URL https://aclanthology.org/2025.findings-emnlp.1148/.

[7] A. Liu, L. Pan, X. Hu, S. Meng, and L. Wen. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6p8lpe4MNf.

[8] L. Pan, A. Liu, X. Hu, and L. Wen. MarkLLM: An open-source toolkit for LLM watermarking. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–70, Miami, Florida, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-demo.8. URL https://aclanthology.org/2024.emnlp-demo.8/.

[9] L. Pan, A. Liu, X. Hu, and L. Wen. Can LLM watermarks robustly prevent unauthorized knowledge distillation? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13524–13543, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.658. URL https://aclanthology.org/2025.acl-long.658/.

[10] W. Pang, X. Hu, Y. Chen, Y. Nie, J. Xu, L. Wen, and A. Liu. No free lunch in LLM watermarking: Trade-offs in watermarking design choices. In *Advances in Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=LEwb4yQp6z.

[11] Z. Tu, Y. Huang, L. Yang, J. Li, T. Li, Z. Zhang, Z. J. Zhang, W. Chen, and Y. Cheng. Waterbench: Towards holistic evaluation of watermarks for large language models. In *Proceedings*

of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7792–7824, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.424. URL https://aclanthology.org/2024.acl-long.424/.

[12] X. Zhao, Z. Wang, L. Li, Z. Chen, H. Lu, J. Wen, H. Tang, Y. Xu, Y. Chen, et al. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=SsmT8aO45L.

[13] X. Zhao, S. Gunn, M. Christ, J. Fairoze, A. Fabrega, N. Carlini, S. Garg, S. Hong, M. Nasr, F. Tramer, S. Jha, L. Li, Y.-X. Wang, and D. Song. Sok: Watermarking for ai-generated content, 2025. URL https://arxiv.org/abs/2411.18479.