δ-STEAL: LLM Stealing Attack with Local Differential Privacy

Kieu Dang[⋆] VDANG@ALBANY.EDU
Phung Lai[⋆] LAI@ALBANY.EDU
NhatHai Phan[‡] PHAN@NJIT.EDU
Yelong Shen[§] YELONG.SHEN@MICROSOFT.COM
Ruoming Jin[¶] RJIN1@KENT.EDU
Abdallah Khreishah[‡] ABDALLAH@NJIT.EDU
[⋆]*University at Albany,* [‡]*New Jersey Institute of Technology,* [§]*Microsoft,* [¶]*Kent State University*

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Large language models (LLMs) demonstrate remarkable capabilities across various tasks. However, their deployment introduces significant risks related to intellectual property. In this context, we focus on model stealing attacks, where adversaries replicate the behaviors of these models to steal services. These attacks are highly relevant to proprietary LLMs and pose serious threats to revenue and financial stability. To mitigate these risks, the watermarking solution embeds imperceptible patterns in LLM outputs, enabling model traceability and intellectual property verification.

In this paper, we study the vulnerability of LLM service providers by introducing δ-STEAL, a novel model stealing attack that bypasses the service provider's watermark detectors while preserving the adversary's model utility. δ-STEAL injects noise into the token embeddings of the adversary's model during fine-tuning in a way that satisfies local differential privacy (LDP) guarantees. The adversary queries the service provider's model to collect outputs and form input-output training pairs. By applying LDP-preserving noise to these pairs, δ-STEAL obfuscates watermark signals, making it difficult for the service provider to determine whether its outputs were used, thereby preventing claims of model theft. Our experiments show that δ-STEAL with lightweight modifications achieves attack success rates of up to 96.95% without significantly compromising the adversary's model utility. The noise scale in LDP controls the trade-off between attack effectiveness and model utility. This poses a significant risk, as even robust watermarks can be bypassed, allowing adversaries to deceive watermark detectors and undermine current intellectual property protection methods.

Keywords: LLMs; Stealing Attacks; Watermarks; Local Differential Privacy

1. Introduction

Large language models (LLMs), such as ChatGPT (OpenAI, 2024) or Gemini (Google, 2024), have demonstrated remarkable capabilities in diverse tasks such as text generation, machine translation, and reasoning (L. et al., 2024). Despite their widespread adoption, training these models demands significant computational resources and human effort, leading to their common deployment as services through APIs (Azure, 2021). While API access restricts direct access to model weights, it does not prevent adversaries from exploiting the model's outputs, undermining the intellectual property protection of proprietary models. This prevents a service provider from protecting ownership over its generated content. Adversaries can exploit generated outputs to fine-tune their local models, effectively replicating the LLM's behavior in specific domains if collecting sufficient data (Carlini et al., 2024; Jovanović et al., 2024). These deployment risks highlight significant concerns about the intellectual property protection of the service provider's proprietary LLMs.

To mitigate risks, service providers have implemented various strategies, including watermarking, encryption, and restricted API access (Kirchenbauer et al., 2023; Xue et al., 2022). Among them, watermarking techniques (Christ et al., 2024; Kirchenbauer et al., 2023; Kuditipudi et al., 2023) are especially effective, offering traceability and scalability. Typically, watermarks inject imperceptible patterns into LLM's outputs. Thanks to these unique patterns, once a watermark is applied, the service provider can use watermark detectors to verify the existence of watermarks in generated text, enabling traceability and detection of intellectual property violations through content tracking.

To bypass watermark detectors and replicate the behavior of LLMs, adversaries employ various techniques, including watermark removal (Krishna et al., 2024; Pan et al., 2024; Zhang et al., 2024; Kirchenbauer et al., 2023) and model stealing (Jovanović et al., 2024; Carlini et al., 2024; W. and C., 2024; Birch et al., 2023). Watermark removal typically involves modifying outputs through synonym substitution, paraphrasing, or sentence restructuring to erase traceable patterns. Model stealing, on the other hand, attempts to extract watermark components from service providers via API queries. However, these attacks face key challenges as they often degrade model utility by increasing perplexity or distorting the semantic meaning of watermarked outputs (Ren et al., 2024; Zhang et al., 2024), and they are typically limited to specific watermark types, recovering only small portions of the original model. As a result, it becomes difficult for adversaries to replicate the behaviors of LLMs without impacting their functionality or output quality.

Key contributions. To balance the trade-off between attack effectiveness and model utility, we introduce a novel model stealing attack, called δ-STEAL, designed to bypass watermark detectors while maintaining high model utility. The key idea is to leverage the concept of local differential privacy (LDP) to obscure the differences between watermarked and non-watermarked LLM outputs. By injecting LDP-preserving noise into the token embeddings of the adversary's model during fine-tuning, our attack makes watermarked outputs indistinguishable from non-watermarked ones, hindering accurate detection. This prevents service providers from verifying model ownership and protects the stolen model from being detected. Our key contributions are summarized as follows.

- **δ-STEAL**: We propose δ-STEAL LLM stealing attack, which incorporates LDP techniques to ensure that the adversary's model retains its effectiveness, functionality, and semantic quality of its outputs, closely resembling that of the service provider's model.

- **Controlling the Trade-off between Attack Effectiveness and Model Utility**: Adjusting the noise scale δ, we can balance the trade-off between attack effectiveness and utility, offering a promising solution for adversaries to steal model behaviors without sacrificing utility.

- **Generality and Flexibility**: δ-STEAL is model-agnostic and watermark-agnostic, allowing it to function across different LLMs and watermarks. This versatility makes it highly adaptable and practically applicable in a variety of real-world scenarios.

- **Experimental Validation**: Our experiments demonstrate that through lightweight modifications during the fine-tuning process of the adversary's local LLM, δ-STEAL effectively bypasses existing watermarks across different LLMs and attacks, achieving attack success rates of up to 96.95% while being supported by theoretical guarantees.

Code and appendices are at: https://github.com/kirudang/LDP_Stealing_Attack

Table 1: A summary of δ-STEAL and related work. WM stands for watermark.

Attack	Objectives	Scope	Guarantee	Modifications
Dipper (Krishna et al., 2024)	Sentence-level WM removal	WM-agnostic	✗	Modify WM sentences using its pre-trained model
Substitution (Pan et al., 2024)	Token-level WM removal	WM-agnostic	✗	Modify WM text with synonyms
WMRemoval (Zhang et al., 2024)	Token-level WM removal	WM-agnostic	✗	Modify WM text with quality check
CoRPG (Lin et al., 2021)	Paragraph-level WM removal	WM-agnostic	✗	Modify WM text based on sentence relationship
Spoofing (Jovanović et al., 2024)	Steal WM rules	Green-red WM (KGW family (Kirchenbauer et al., 2023))	✗	Learn WM rules from target LLM's output
Color-Substitution (W. and C., 2024)	Steal WM vocabulary lists	Green-red WM (KGW family (Kirchenbauer et al., 2023))	✓	Observe LLM's outputs by repeating API queries
Part Stealing (Carlini et al., 2024)	Steal LLM's last layer	WM-agnostic	✗	Extract LLM's last layer via API
δ-STEAL (Ours)	WSteal LLM's behaviors	WM-agnostic	✓	Lightweight modifications (No modifications on LLMs)

2. Related Work

There are two main types of attacks that threaten LLM intellectual property protection. First, *watermark removal attacks* (Krishna et al., 2024; Pan et al., 2024; Zhang et al., 2024) modify outputs to disrupt traceable watermark patterns using techniques like paraphrasing, synonym substitution, and sentence restructuring. While effective at weakening watermark detection, these methods often degrade text quality. Second, *stealing attacks* involve extracting watermark components, such as watermarking rules, vocabulary lists, or parts of a production LLM via APIs (Jovanović et al., 2024; Carlini et al., 2024; W. and C., 2024; Birch et al., 2023). The goal is to replicate the behaviors of the service provider model and bypass watermark detectors, but existing methods are limited to specific watermark types and can only extract small portions of an LLM.

Table 1 highlights key differences between δ-STEAL and related work. Our attack follows the second direction but focuses on stealing LLM behaviors in a model-agnostic and watermark-agnostic manner. It only introduces a lightweight modification by adding LDP noise into the token embeddings, without changing the tokens, modifying LLM itself, or requiring model access. Furthermore, it can be easily applied to any LLM or watermark, enhancing its generalizability and scalability.

3. Background

3.1. Model Stealing Attacks

Due to the high cost of training, LLMs are typically offered via APIs (Azure, 2021), but this does not prevent model stealing attacks (C., 2025; Zhang et al., 2023). Such attacks can cause intellectual property violations, service replication, fee evasion, and system exploitation, underscoring the need for robust defenses. Model stealing attacks (Zhang et al., 2023; Jovanović et al., 2024; Carlini et al., 2024) are also referred to as imitation attacks (Wallace et al., 2020; X. et al., 2022) or model extraction attacks (Krishna et al., 2020; Birch et al., 2023). To steal a model θ, adversaries query the API to collect N input-output pairs $\{x_i, y_i\}_{i=1}^{N}$ and train or fine-tune a surrogate model θ_{adv}. The goal is to replicate the behavior of θ while bypassing watermark detectors.

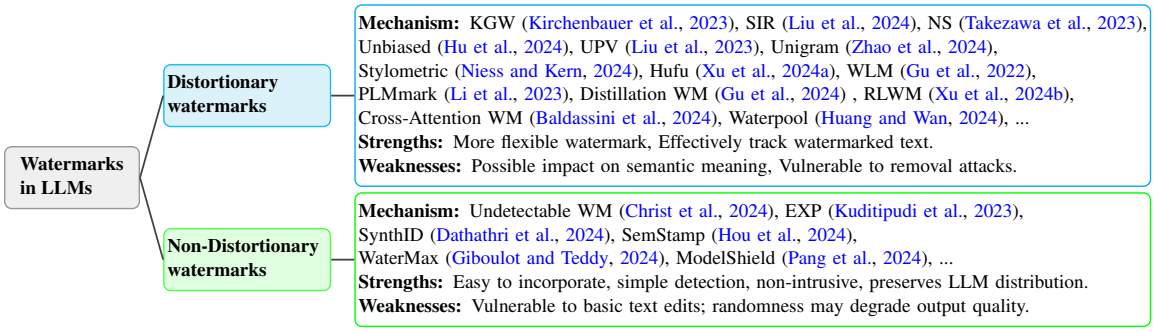


Figure 1: Taxonomy of Watermarks in LLMs.

3.2. Watermarking Techniques

Recent work has shown that watermarks are effective in defending against model stealing attacks (He et al., 2022b,a; Kuditipudi et al., 2023). Given a prompt x, the service provider uses its model θ and a watermarking function $\mathcal{W}$ to generate a watermarked output $y^{wm} = \theta(x, \mathcal{W})$. A detector $\mathcal{D}(\cdot)$, based on statistical tests or classifiers (Kirchenbauer et al., 2023; Kuditipudi et al., 2023), is then used to verify the presence of the watermark. By analyzing detection rates, service providers can set thresholds to identify whether a suspect model has stolen their model's behavior.

Fig. 1 shows the taxonomy of watermarks in LLMs, along with their advantages and disadvantages. Watermarks can be classified into distortionary and non-distortionary types. *Distortionary watermarks* modify LLM weights or logits, affecting its functioning. Examples include logits-based watermarks (Kirchenbauer et al., 2023; Liu et al., 2024; Zhao et al., 2024), which change token generation logits, and training-based watermarks (Baldassini et al., 2024; Gu et al., 2024; Xu et al., 2024b), which alter the training process. While effective for tracking, these watermarks may impact semantic meaning and are vulnerable to removal attacks (Carlini et al., 2024; Krishna et al., 2024; Zhang et al., 2024). *Non-distortionary watermarks* inject watermarks without altering output distributions, weights, or logits. Examples include sampling-based watermarks (Hou et al., 2024; Kuditipudi et al., 2023), which modify token or sentence sampling strategies, and prompt-based or multiple-output watermarks (Giboulot and Teddy, 2024; Pang et al., 2024), which generate multiple candidate outputs and select the one with the most identifiable watermark. These methods are harder to detect but can be vulnerable to text modifications and may reduce reliability.

3.3. Local Differential Privacy (LDP)

LDP is a mathematically provable method for ensuring data privacy (E. et al., 2014), based on randomized survey responses (Warner, 1965), where noise is added to data to protect confidentiality and prevent accurate inference of individual data. The LDP definition is as follows:

Definition 1. *A randomized algorithm $\mathcal{A}$ satisfies ϵ-LDP, if for any two inputs x and x', and for all possible outputs $\mathcal{O} \in Range(\mathcal{A})$, we have: $Pr[\mathcal{A}(x) = \mathcal{O}] \leq e^{\epsilon} Pr[\mathcal{A}(x') = \mathcal{O}]$, where ϵ is a privacy budget and $Range(\mathcal{A})$ denotes every possible output of $\mathcal{A}$.*

The privacy budget ϵ controls the difference between the distributions induced by inputs x and x'. A smaller ϵ results in a larger gap between the outputs and vice versa.

In this work, we apply LDP to perturb the token embeddings of adversaries' data, bounding the differences between tokens and their associate outputs under LDP guarantees. We employ the

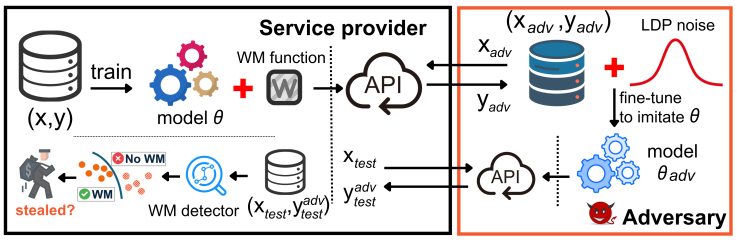


Figure 2: System architecture of δ-STEAL.

Laplace mechanism (Dwork et al., 2014), a commonly used LDP approach, to add Laplace noise. For all inputs x and x' in the domain of $\mathcal{A}$, where d is the input dimension, the Laplace noise $Lap(0, \delta)$ has a zero mean and a noise scale δ calculated as follows:

$$\delta = \frac{\max_{\forall x, x' \in R^d} \|\mathcal{A}(x) - \mathcal{A}(x')\|_1}{\epsilon} \tag{1}$$

4. δ-STEAL: LLM Stealing Attack with Local Differential Privacy

We introduce δ-STEAL, a novel model stealing attack that balances attack effectiveness and model utility. To achieve this, we address three fundamental questions: *1) How can we optimize this trade-off? 2) Where should LDP noise be added for the best balance?* and *3) How can we bound output differences to evade watermark detectors?* Our approach involves: *1)* Leveraging LDP concepts and varying levels of LDP guarantees by adjusting the noise scale δ to better control the trade-off, *2)* Adding LDP noise to token embeddings without modifying the tokens themselves, preserving model utility, and *3)* Using LDP guarantees to bound the differences between the outputs generated by the adversaries and by the service provider, bypassing watermark detectors to enhance attack success.

4.1. Setting

This work considers a service provider offering its LLM θ via an API, where users can query a prompt x and receive the corresponding output y. To protect its intellectual property, the service provider injects a watermark, producing $y^{wm} = \theta(x, \mathcal{W})$ to users. Model stealing attacks then aim to replicate the behavior of the target LLM θ given a prompt x in this black-box setting, where users have no access to the model's internal structure, parameters, or inference details (e.g., temperature, cached memory). This setting reflects practical deployments widely adopted by commercial providers.

4.2. Threat Models

Adversary's goals. The adversary aims to 1) replicate the service provider's model without direct access, 2) bypass watermark detectors, 3) obfuscate model ownership to hinder infringement claims, and 4) preserve utility comparable to the original. Achieving these goals enables the adversary to launch a competing service, causing potential revenue loss for the service provider's model θ.

Adversary's capabilities. We assume an attacker who can query the service provider's LLM via API like any user and also has no knowledge of LLM's internal specifications. The LLM discloses no details in its outputs, ensuring no information leakage through the output y.

4.3. System Operation of δ-STEAL

Fig. 2 and Algorithm 1 illustrate the system architecture and pseudo-code of our δ-STEAL.

At the service provider, the model θ is trained on proprietary data (Line 5) and deployed via an API or MLaaS platform. To protect intellectual property, outputs are watermarked with $\mathcal{W}$ (Line 6). In addition, the service provider employs a watermark detector $\mathcal{D}$ to identify whether an arbitrary output y_{adv} from a suspect model θ_{adv} contains a watermark; detection indicates that θ_{adv} was trained on watermarked data (Lines 8–11), enabling prompt detection of intellectual property violations.

At the adversary, the goal is to replicate the service provider's model θ while preserving high utility. To achieve this, the adversary queries the service provider's model θ with N prompts $\{x_i\}_{i=1}^N$ and collects watermarked responses $y_i^{wm} = \mathcal{W}(\theta(x_i))$ (Line 15). These prompt-output pairs $\{x_i, y_i^{wm}\}_{i=1}^N$ are then used to fine-tune the adversary's model θ_{adv}. By leveraging these correlations, the adversary can mimic the service provider model's functionality and behavior during training. Unlike conventional attacks (Zhang et al., 2024; Pan et al., 2024), which typically modify tokens with synonyms but often compromise text coherence, δ-STEAL adds LDP noise to token embeddings without modifying the tokens themselves, thus effectively maintaining utility (Line 16). It is worth noting that the LDP noise is only added once before fine-tuning θ_{adv}. This prevents accumulation of the privacy budget ϵ, ensuring tighter LDP guarantees on the output differences between θ and θ_{adv}.

4.4. Bounding Output Differences with Local Differential Privacy (LDP)

Theoretical Bound. Watermark detectors exploit distinctive statistical patterns between the provider's model θ and the adversary's surrogate θ_{adv}. An effective attack must obscure these patterns while preserving utility, a non-trivial trade-off. Appx. A shows that applying an ϵ-LDP to embeddings bounds the change in output distributions by a factor of $\exp^{\epsilon}$, thereby constraining watermark detectability. This provides a theoretical link between ϵ, δ and the likelihood of successful evasion.

Noise Calibration. Our choice of $\delta \in \{0.001, 0.01, 0.05, 0.1\}$ is guided by the global sensitivity of embedding vectors detailed in Appx. B, capturing the transition from negligible perturbation to semantic distortion. Fig. 5b in Section 5.4.6 further connects this to empirical sensitivity differences across LLMs, showing how embedding-level statistics (e.g., mean and variance) from different LLM profiles could inform δ selection to balance watermark evasion with utility preservation.

Embedding-layer Noise. We inject noise at the embedding layer because it offers stronger protection and better utility retention than other deeper layers. First, output-level or latent-level noise is more vulnerable to query averaging and reconstruction attacks (Dwork et al., 2014), whereas adding noise to the embedding, we can hide the noise in model parameter though training, which would enhance the attack. Second, prior work shows that calibrating LDP on embeddings preserves semantics and yields favorable utility (Meisenbacher et al., 2024; Feyisetan et al., 2020). In addition, training-time noise at this layer also acts as a form of regularization (Bishop, 1995), allowing early distortion control while avoiding averaging at inference. Lastly, as highlighted, embedding statistics provide δ guideline, reinforcing why the embedding layer is the most effective place to inject noise.

5. Experiments

Our extensive experiments shed light on **1)** δ-STEAL's effectiveness against watermarks; **2)** Its impact on model utility, including text generation and downstream tasks; **3)** Comparisons with existing attacks; **4)** The trade-off between attack success and utility under LDP guarantees; and **5)** The influence of δ-STEAL's components on this trade-off.

Algorithm 1 δ-STEAL Algorithm

Inputs: Service provider's model θ, adversary's model θ_{adv}, N adversary's prompts $\{x_i\}_{i=1}^{N}$, N_{test} service provider's testing prompts $\{x_j\}_{j=1}^{N_{test}}$, training iterations T, watermark $\mathcal{W}$, detector $\mathcal{D}$, loss $\mathcal{L}$
Outputs: Adversary's model θ_{adv} and watermark detector results $\mathbb{I}(\theta_{adv}, \theta)$
At the Service Provider:
  Watermark Injection
    Initialize model parameters and fine-tune θ using training data available at the service provider
    Inject a watermark $\mathcal{W}$ to outputs before releasing them to users: $y^{wm} = \theta(x, \mathcal{W})$
  Watermark Detection
    for $j = 1, \ldots, N_{test}$ **do**
      Query the adversary's model to collect its outputs: $y_j = \theta_{adv}(x_j)$
      Check if y_j is watermarked: $\mathbb{I}\big(\mathcal{D}(y_j) = 1\big)$
  Return $\mathbb{I}(\theta_{adv}, \theta)$ based on $\mathbb{I}\big(\mathcal{D}(y_j) = 1\big)$ for all $\{y_j\}_{j=1}^{N_{test}}$
At the Adversary:
  Initialize model parameters θ_{adv}
  for $i = 1, \ldots, N$ **do**
    Query the service provider's model to collect its outputs: $y_i^{wm} = \theta(x_i, \mathcal{W})$
    Add Laplace LDP noise into token embeddings with a noise scale δ:
      $\{\bar{x}_i, \bar{y}_i\} = \{x_i + Lap(0, \delta), y_i^{wm} + Lap(0, \delta)\}$
  Form a training set: $D_{adv} = \{\bar{x}_i, \bar{y}_i\}_{i=1}^{N}$
  for $t = 1, \ldots, T$ **do**
    Randomly select a set of training samples $D_t \subseteq D_{adv}$
    $\theta_{adv}^{t} = \theta_{adv}^{t-1} - \eta \bigtriangledown_{\theta_{adv}} \mathcal{L}(\theta_{adv}^{t}, D_t)$
  Return $\theta_{adv} = \theta_{adv}^{T}$

5.1. Baselines

We evaluate six state-of-the-art (SOTA) watermarks: **1)** *KGW* (Kirchenbauer et al., 2023), which splits the vocabulary into green/red tokens and biases logits toward green ones; **2)** *EXP* (Kuditipudi et al., 2023), which maps sequence keys to tokens during sampling; **3)** *SIR* (Liu et al., 2024), which adjusts logits based on the semantics of previous tokens; **4)** *SemStamp* (Hou et al., 2024), which accepts sentences mapped to valid semantic regions; **5)** *TW* Yang et al. (2023): A binary text watermark that replaces tokens with context-aware synonyms, guided by a Bernoulli-based random encoding; and **6)** *DeepTextMark* Munyer et al. (2024) (referred to as *DTM*), which substitutes tokens with synonyms using Word2Vec Mikolov et al. (2013). Each method includes its own watermark detector and watermarked output under attack-free environment is denoted as the *Baseline*.

In addition, we compare δ-STEAL with three SOTA watermarking attacks: **1)** *WMremoval* (Zhang et al., 2024), which paraphrases outputs while using a quality oracle to preserve fluency; **2)** *Dipper* (Krishna et al., 2024), which paraphrases the outputs through context reordering and lexical changes; and **3)** *Substitution*, adapted from (Pan et al., 2024), which replaces words with WordNet synonyms (Miller, 1995). An LLM output without watermarks or attacks is denoted as the *Original*.

5.2. Dataset and Model Configurations

We evaluate δ-STEAL on text generation and downstream tasks. *For text generation*, we randomly select $10,000$ training samples and $2,000$ test samples from the C4 dataset (Dodge et al., 2021), using the first 200 tokens as prompts and generating up to 200 tokens. This setup applies to Original and

Baseline. *For the downstream task*, we use the massive multi-task language understanding MMLU (Hendrycks et al., 2021), a multiple-choice benchmark across 57 subjects of varying difficulty.

We conduct experiments on two LLMs, including LLaMA-2 7B (Touvron et al., 2023) and Mistral 7B (Jiang et al., 2023), as the Original models. We employ a Laplace mechanism (Dwork et al., 2014), widely used in LDP, to add noise to the token embeddings. We vary the Laplace noise $Lap(0, \delta)$ with different noise scales $\delta \in \{0.001, 0.01, 0.05, 0.1\}$. These noise scales are associated with ϵ values of $\{300, 30, 6, 3\}$ for the LLaMA-2 and $\{50, 5, 1, 0.5\}$ for the Mistral. Details on how δ is computed from noise scales are provided in Appx. B. We fine-tune adversary models using LoRA (Hu et al., 2022) with the Adam optimizer on 10 epochs with a learning rate of 10^{-5}.

5.3. Evaluation Metrics

A model stealing attack is considered successful if it achieves *high attack success rates* while *preserving utility*, ensuring strong performance on both the main task (e.g., text generation) and downstream tasks (e.g., MMLU). To evaluate δ-STEAL, we examine three key aspects. *First*, for effectiveness, we compute the attack success rate (AttackSR) as follows:

$$\text{AttackSR} = 1 - \frac{\sum_{i=1}^{N_{test}} \mathbb{I}\Big(\mathcal{D}(\theta_{adv}(x_i)) = 1\Big)}{N_{test}} \tag{2}$$

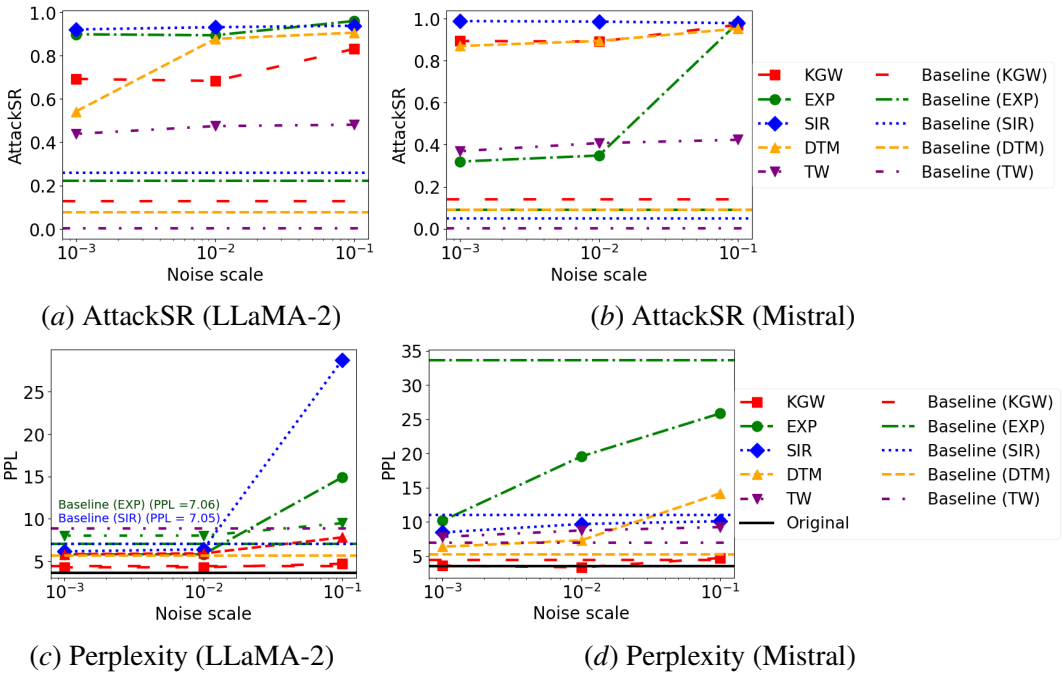
where N_{test} is the number of test samples located at the service provider and $\mathcal{D}(\theta_{adv}(x_i))$ represents the watermark detector, such that $\mathcal{D}(\theta_{adv}(x_i)) = 1$ if the output of θ_{adv} given x_i, denoted as $\theta_{adv}(x_i)$, is watermarked and $\mathcal{D}(\theta_{adv}(x_i)) = 0$ otherwise. In addition, $\mathbb{I}$ is the indicator function, where $\mathbb{I}(x) = 1$ if x is True, and $\mathbb{I}(x) = 0$ otherwise. Eq. 2 defines AttackSR, which measures the failure rate of watermark detection on the adversary's outputs; higher values indicate better evasion. *Second*, we assess utility using: 1) Perplexity (PPL) for text generation, and 2) Average accuracy on MMLU (Hendrycks et al., 2021). *Third*, we further perform qualitative analysis through side-by-side visualization of prompt-output pairs across watermarks and attacks on different LLMs.

5.4. Experimental Results

5.4.1. δ-STEAL AGAINST EXISTING WATERMARKS

In Fig. 3, δ-STEAL demonstrates effectiveness across watermarks, LLMs, and noise scales, achieving high AttackSR and low PPL (where lower is better). For example, with LLaMA-2 at $\delta = 0.001$, δ-STEAL yields 69.28% AttackSR and 4.29 PPL on KGW, but achieves 89.90% and 92.13% AttackSR with 5.82 and 6.17 PPL on EXP and SIR, respectively. These results show that δ-STEAL is more evasive with SIR, achieving a higher AttackSR with minimal impact on PPL. In addition, as the noise scale δ increases, AttackSR improves while PPL slightly worsens yet remains close to the Baseline and Original outputs. For instance, with KGW on Mistral, as the noise scale δ increases from 0.001 to 0.1, AttackSR rises from 89.28% to 96.95% while PPL slightly increases from 3.61 to 4.73. We observe similar trends with EXP and SIR, showing increases of $1.72\% - 6.14\%$ in AttackSR and $9.08 - 23.41$ in PPL values. Additional results from DTM and TW remained consistent, achieving high AttackSR and minimal utility degradation, further confirming δ-STEAL's effectiveness.

Furthermore, the performance does not exhibit a linear trade-off between AttackSR and PPL. This non-linear response is expected due to the inherent randomness of noise injection and the discrete nature of text. If AttackSR is already near its maximum, further increases in noise will not significantly improve evasion, resulting in flat regions. Similarly, for certain watermarks, moderate



(*a*) AttackSR (LLaMA-2) (*b*) AttackSR (Mistral)

(*c*) Perplexity (LLaMA-2) (*d*) Perplexity (Mistral)

Figure 3: δ-STEAL performance on different watermarks.

noise may be sufficient to destroy the detector, so AttackSR rises quickly while PPL remains relatively stable over a range of δ. These patterns reflect threshold effects rather than linear trade-offs.

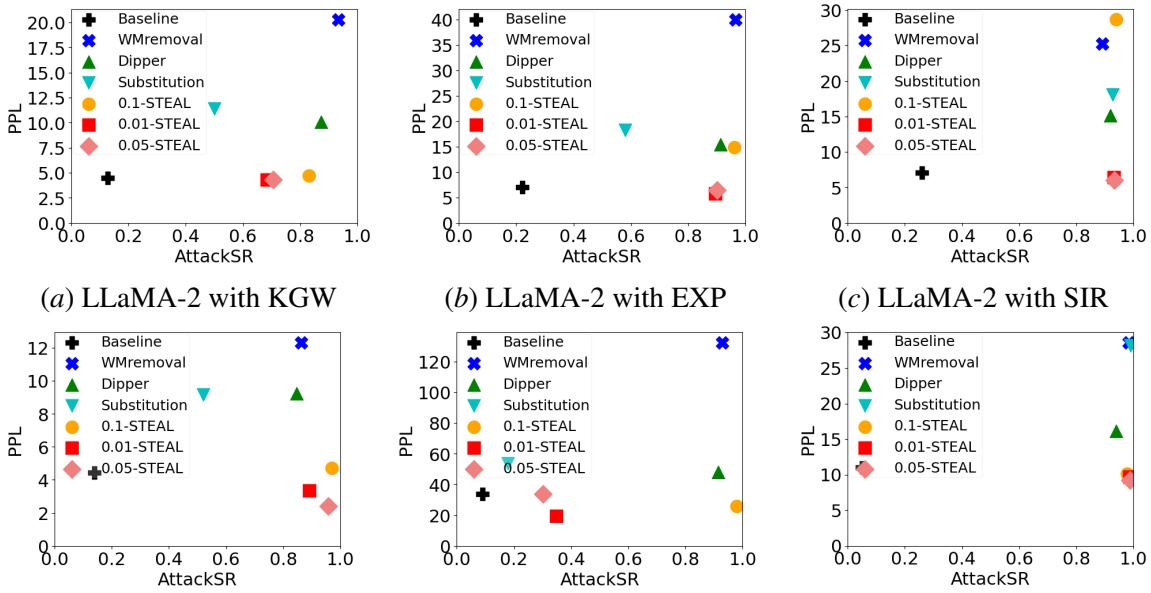
To validate our work, we conducted statistical significance testing on a representative setting of KGW under LLaMA-2 with $\delta = 0.01$. The experiment was repeated five times, comparing the mean AttackSR with the reported 68.34% (Table 2). The observed mean of 68.94% is close to the reported value, and a one-sample t-test yields a p-value of 0.486, indicating no significant difference. These results confirm the consistency and reliability of our findings beyond a single trial.

Table 2: AttackSR significance testing of KGW on LLaMA-2 with $\delta = 0.01$.

Reported AttackSR	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	p-value
68.34%	68.85%	69.00%	68.10%	67.01%	71.75%	0.486

Intuitively, δ-STEAL introduces noise into token embeddings, and as the noise scale increases (i.e., higher values of δ), the modifications become more substantial. This makes it increasingly difficult for watermark detectors to detect the watermarks, thereby improving AttackSR. *In distortionary watermarks* such as KGW or SIR the generated watermarked tokens depend on preceding context. Perturbing token embeddings can alter token selection, causing noise accumulation across tokens and disrupting the watermark signature. Similarly, for *non-distortionary watermarks* such as EXP, watermarked tokens are chosen via a predefined sampling process while preserving the probability distribution. However, the introduction of LDP noise into token embeddings causes deviations in the rule-based token sampling process. This results in inconsistencies in token choices and weakens the injected watermark patterns, reducing detectability of watermark detectors.

Despite these perturbations, δ-STEAL highly preserves model utility for two reasons. First, the perturbed outputs stay close to the original distribution while the training pipeline remains unchanged, allowing the surrogate θ_{adv} to capture semantic and syntactic patterns of the provider's model. Second, since the noise is added only during fine-tuning in a controllable LDP-preserving manner, the core learning signal remains intact. This allows the stolen model's outputs to retain fluency, coherence, and task performance, even while obfuscating watermark patterns. As a result, δ-STEAL empirically achieves a strong balance between watermark evasion and model functionality, making it highly effective for stealing proprietary LLM behavior without degrading output quality.



(*a*) LLaMA-2 with KGW (*b*) LLaMA-2 with EXP (*c*) LLaMA-2 with SIR

(*d*) Mistral with KGW (*e*) Mistral with EXP (*f*) Mistral with SIR

Figure 4: AttackSR and Perplexity results. (Best attacks are in the bottom-right.)

5.4.2. COMPARISON OF δ-STEAL AND EXISTING ATTACKS

Fig. 4 compares the performance of δ-STEAL with other attacks (WMremoval, Dipper, and Substitution). Notably, a higher AttackSR and lower PPL indicate a better attack; therefore, attacks appearing in the bottom-right corner are considered more effective. As shown, δ-STEAL achieves a high AttackSR while maintaining low PPL, comparable to that of the Baselines. For instance, with $\delta = 0.1$, δ-STEAL achieves an AttackSR of 83.14% and a PPL of 4.71, compared to 4.46 of LLaMA-2 with KGW Baseline. Similarly, for Mistral with KGW, δ-STEAL achieves an AttackSR of 96.95% at a PPL of 4.73, compared to the Baseline PPL of 4.43. Meanwhile, other attacks usually exhibit lower values of AttackSR but much higher values of PPL, indicating a substantial impact on the watermarked outputs, reduced attacked output quality. For instance, WMremoval achieves a high AttackSR of 93.44% for LLaMA-2 with KGW but raises PPL to 20.31 from the Baseline of 4.46. It is also computationally expensive, requiring paraphrasing and quality checks for every token to generate non-watermarked outputs. Dipper encounters similar issues, reaching an AttackSR of 87.37% at the cost of 10.09 PPL while significantly degrading model utility on downstream tasks (Table 3). Substitution attack performs worst, showing high PPL or low AttackSR across settings due to random token replacements that disrupt semantics and reduce utility.

5.4.3. δ-STEAL ON DOWNSTREAM TASKS

Table 3: MMLU accuracy across LLMs and watermarks.

Accuracy (%)		LLaMA-2	Mistral
Original		46.70	58.90
KGW	Baseline	44.50	58.60
	0.001-STEAL	44.10	42.30
	0.01-STEAL	43.00	39.30
EXP	Baseline	45.90	58.70
	0.001-STEAL	45.50	47.70
	0.01-STEAL	43.30	29.20
Dipper		9.60	

Table 3 compares MMLU performance under three settings: *1)* the *Original*, *2) Baseline*, and *3)* δ-STEAL attack applied to the Baseline with noise scales $\delta \in [0.001, 0.01]$. Overall, δ-STEAL maintains model utility with only a subtle drop in accuracy compared with the Original and Baseline. For LLaMA-2, δ-STEAL maintains strong performance across all watermarks, with a small accuracy drop of $1.5 - 2.6\%$ at $\delta = 0.01$. Meanwhile, Mistral exhibits greater sensitivity, showing a more significant accuracy drop up to 29.5% at $\delta = 0.01$ and 16.30% at $\delta = 0.001$.

The MMLU task is characterized by low entropy and more deterministic text, since outputs are multiple-choice answers, thus making it highly sensitive to noise. Even small perturbations can lead to noticeable drops in performance, especially in models like Mistral. In contrast, Dipper, the most effective attack in the early experiments, performs poorly on this task, achieving only 9.6% accuracy. This is due to its aggressive paraphrasing process, which significantly alters the semantics of the text and disrupts the model's ability to select the correct multiple-choice answers. These observations further emphasize δ-STEAL's effectiveness in maintaining model utility unlike other attacks.

5.4.4. TRADE-OFF BETWEEN ATTACK EFFECTIVENESS AND MODEL UTILITY IN δ-STEAL

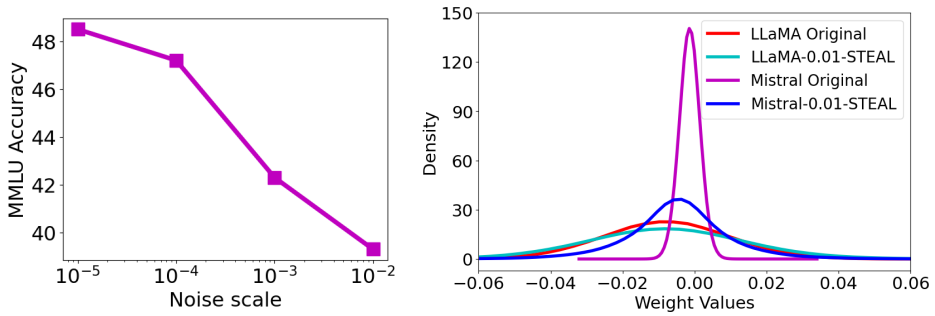
Throughout all experiments, we observe a trade-off between attack effectiveness and model utility. As the noise scale δ increases, AttackSR improves, while model utility decreases. Intuitively, increasing the noise scale δ introduces greater perturbations to the token embeddings. This increases the difference between outputs generated by the service provider and those generated by the adversary, making it more challenging for watermark detectors to detect the presence of watermarks, thereby improving the AttackSR. However, the added noise can distort text quality, making it crucial to choose an optimal noise scale δ that could efficiently balance attack effectiveness and model utility. In this work, we present empirical results with $\delta \in [0.001, 0.1]$, and discuss its effect on the embedding space in section 5.4.6, leaving the theoretical analysis of optimal noise for future work.

5.4.5. SEMANTIC PRESERVATION OF δ-STEAL OUTPUTS

Table 4 presents different watermarked baseline examples of LLaMA-2. Additional examples are provided in Table 6 (Appx.C). We observe that δ-STEAL effectively preserves semantic meaning compared to the Baselines, as indicated by the large green portions in text, while the red portions, despite textual changes, retain the original content. For instance, the δ-STEAL attack on KGW introduces slight modifications while preserving the overall semantic meaning, replacing "elections in the city" with "polls in Delhi" (based on earlier context mentioning Delhi) or "We have to show that the BJP's victory" with "We cannot ignore the BJP's victory." Examples with EXP and SIR follow a similar pattern. For instance, in the EXP example, given a financial disclosure context, "Trump denies the allegation" is modified to "Cohen recently revealed that Trump personally reimbursed him," subtly shifting the focus but retaining the core idea. In the SIR example, "traditional television platform" is changed to "major network television" without significantly changing the meaning.

Table 4: Prompts and watermarked outputs w/wo δ-STEAL. Green is similarities, red is differences.

Prompt	Watermarked output (W)	δ-Steal ($\delta = 0.01$) output (A)	PPL(W)	PPL(A)
... do that for the MCD polls, " said a senior party leader. Another reason behind not targeting Modi in the MCD polls is the massive victory the B...	**KGW:** JP achieved in the 2014 Lok Sabha polls in Delhi , where the party won all seven seats. " We have to take into account the ground reality and the political equation in the city. We can not ignore the BJP 's victory in the 2014 Lok Sabha [...continues]	**KGW:** JP registered in the 2014 Lok Sabha elections in the city. " We have to convince the people of Delhi that why we are the best option to govern the city. We have to show that the BJP 's victory in the 2014 Lok Sabha elections was a [...continues]	3.33	3.67
... Daniels, who was born Stephanie Clifford, was paid 130,000 by Cohen after she signed a nondisclosure agreement barring her from talking about her alleg...	**EXP:** ed affair with Trump. Trump denies the allegation. Cohen has admitted making the payment to Daniels, which he said was done to protect Trump' s campaign from the allegations. It' s possible that the reimbursement payment revealed in the financial disclosure may [...continues]	**EXP:** ed affair with Trump. Cohen recently revealed that Trump personally reimbursed him for the payment to Daniels. As a result of the payment to Cohen, the government ethics office sent a reminder to Trump that he must disclose in his annual financial disclosure report any payments made [...continues]	4.12	5.05
... other shows, it's a GoPro on a windshield, " Foley said referring to Ride Along. " I think if this was a show that was on...	**SIR:** a traditional television platform, we would be able to do more with it. As it stands, it's a while we do any new episodes of Holy Folesy! " Foley also addressed why his daughter Noelle is n't pursuing a career [...continues]	**SIR:** a major network television, we would be able to do more with it. I think we would have a bigger budget and be able to do some cool things with it. " Foley also talked about why his daughter Noelle is n't pursuing an [...continues]	8.12	8.23



(*a*) MMLU Accuracy (Mistral) (*b*) Embedding Distribution across LLMs

Figure 5: MMLU accuracy and Sensitivity of different LLMs.

5.4.6. VARYING COMPONENTS OF δ-STEAL

Varying LDP Levels. Table 3 and Fig. 3 show that as the noise scale δ increases, AttackSR improves while PPL values rise. For instance, in text generation, with $\delta = 0.001$, δ-STEAL achieves 69.28% AttackSR and 4.29 PPL under LLaMA-2 with KGW. When δ increases to 0.1, AttackSR and PPL also increase to 83.14% and 4.71 respectively. Similarly, in the MMLU downstream task, at $\delta = 0.001$, δ-STEAL achieves 44.10% in average accuracy, but the performance drops to 43.00% as δ is increased to 0.01. This observation is consistent with different watermarks and LLMs.

Varying LLMs. Table 3 and Figs. 3, 4 exhibit similar trade-offs between attack effectiveness and utility across watermarks and LDP levels across LLMs. However, Mistral demonstrates larger performance gaps across noise scales, whereas those of LLaMA-2 remains relatively stable. For instance, in Table 3, the average accuracy of MMLU using LLaMA-2 only drops by 0.8% between noise scales of 0.01 and 0.001, while Mistral's average accuracy drops by 3.0%. Fig. 5a further shows that Mistral maintains high MMLU accuracy at smaller noise scales, achieving results comparable to the Baseline. At $\delta = 10^{-4}$, δ-STEAL on Mistral achieves 85.99% AttackSR with a low PPL of 4.44.

Table 5: Effect of adversary training data size on AttackSR and PPL (LLaMA-2, KGW, $\delta = 0.01$).

Training size	100	1,000	5,000	10,000	20,000	50,000	100,000
AttackSR (%)	97.95	89.94	78.30	68.33	62.58	56.08	45.30
PPL	4.54	4.36	4.56	4.30	5.09	5.14	5.17

The performance gap between LLaMA-2 and Mistral under δ-STEAL stems from differences in their embedding layer weight distributions. In Fig. 5b, Mistral's embedding weights follow a sharply peaked distribution with a significantly smaller standard deviation of 0.0027, while LLaMA-2's distribution is flatter, with a standard deviation of 0.01681, approximately $6.22\times$ higher. As a result, Mistral is more sensitive to perturbations, and applying the same noise scale to this LLM would cause greater distortion. Therefore, analyzing the embedding weight distribution is a practical approach to guide the choice of noise scale δ and to anticipate its impact on attack performances.

SemStamp results are also reported in Appx. C. We used the authors' pre-trained sentence embedder without fine-tuning, which is not well-suited for our setup. While fine-tuning a stronger embedder would likely improve detection, it is beyond the scope of this work. Nonetheless, the results provide useful insights into SemStamp's performance under these conditions.

5.4.7. VARYING TRAINING DATA SIZE

Our experiments assume an attacker with a maximum query budget of 10^4, which is substantial and potentially costly in practice. However, an adversary could accumulate more prompts over time to train a local model. To study this, we vary the number of training samples from 100 to $100,000$ using LLaMA-2 with KGW and a noise scale of 0.01. In Table 5, as the training size increases, the AttackSR decreases. This is because larger training datasets may lead the adversary model to overfit watermark-related patterns, making it more detectable. These results highlight an important caution, in which more data does not necessarily equate to more effective attacks and underscore the need for overfitting-resistant training strategies in large-scale deployments of δ-STEAL.

6. Discussion

Novelty and Contribution. While our work has limited novelty in terms of algorithmic mechanics, its contribution lies in being the first to integrate LDP theory with model stealing as a lightweight and empirically effective watermark evasion method. Prior studies on LDP have not considered this threat model, and we view our results as opening a new direction for the watermark security research.
Theoretical Limitations. Our framework does not fully formalize the link between ϵ-LDP guarantees and the disruption of watermark-specific statistical features. The present work relies mainly on empirical evidence and intuition about embedding-layer noise to break detector correlations. Although Appx. A provides a bound on distributional shifts under ϵ-LDP, it is not a formal proof that watermark detectability is eliminated. Establishing rigorous theoretical links between ϵ, δ parameters and bounded watermark detection remains an important direction for future research.
Practical Deployment and Scalability. We acknowledge that real-world deployment may introduce additional complexity, such as diverse model architectures, varying query patterns, and adaptive watermarking defenses. However, the initial results from δ-STEAL, showing high attack success rates with minimal utility loss, suggest practical feasibility in deployment-like settings. While finer-grained δ values could provide more detailed insights, they would significantly increase training

cost; the noise levels were carefully calibrated using embedding sensitivity, capturing the range from negligible perturbation to semantic distortion. In addition, leveraging efficient LoRA fine-tuning with modest hardware, δ-STEAL remains scalable to larger models and datasets.

Ablation on Noise Location. To better understand the effect of noise injection location, we conduct a study comparing three strategies on LLaMA-2 with KGW at $\delta = 0.01$ on (1) noisy embeddings during fine-tuning (δ-STEAL), (2) noisy pre-logits, and (3) noisy embeddings applied at inference. Adding noise only at inference produces the lowest utility (highest PPL 5.27) and the lowest AttackSR (67.69%), since it directly affects model performance at deployment. Injecting noise at the pre-logits layer raises AttackSR by nearly 9% (74.75%) but also increases PPL by 13% (PPL 4.86), reflecting a less favorable utility–robustness balance. By contrast, δ-STEAL setting achieves the best trade-off, confirming the effectiveness of embedding-layer perturbation. Details are reported in Appx. D.

7. Conclusion

This study introduces δ-STEAL, a novel model stealing attack that leverages LDP to bypass watermark detectors. By applying LDP to token embeddings without altering the tokens themselves, δ-STEAL effectively preserves model utility. In addition, we bound the differences between watermarked and non-watermarked outputs, making it difficult for the service provider to distinguish whether the adversary's model was trained on the service provider's watermarked data, thereby evading detection. We show that even with small noise, δ-STEAL can evade watermark detectors with a high AttackSR, reaching up to 96.95% and an average of over 80%, while maintaining model utility similar to the original model without attacks. Furthermore, δ-STEAL outperforms existing attacks across different watermarks, LDP levels, and LLMs, enhancing its practical applicability in various scenarios.

Although our empirical analysis quantitatively shows the connection between the injected noise scale δ and attack effectiveness, establishing formal theoretical connections remains a challenging and open problem. In addition, deriving certified bounds that link embedding sensitivity, perturbation magnitude, attack success, and downstream performance degradation could provide deeper insights into these trade-offs and is an important direction for future work.

References

Azure. https://aka.ms/AzureMLModelInterpretability, 2021.

Folco Bertini Baldassini et al. Cross-attention watermarking of LLMs. In *ICASSP*, 2024.

Lewis Birch et al. Model leeching: An extraction attack targeting llms. *CAMLIS*, 2023.

C. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 1995.

Simon C. Good models borrow, great models steal: Ip rights and gai. *Policy & Society*, 2025.

Nicholas Carlini, Daniel Paleka, et al. Stealing part of a production language model. *ICML*, 2024.

Miranda Christ et al. Undetectable watermarks for language models. In *COLT*, 2024.

Sumanth Dathathri et al. Scalable watermarking for identifying llm outputs. *Nature*, 634, 2024.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, and et al. Documenting large webtext corpora: A case study on the colossal clean crawled corpus, 2021.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *FToCS*, 2014.

Úlfar E. et al. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, 2014.

Oluwaseyi Feyisetan, Borja Balle, Thomas Drake, and Tom Diethe. Privacy-and utility-preserving textual analysis via calibrated multivariate perturbations. In *Proceedings of WSDM*, 2020.

Eva Giboulot and Furon Teddy. Watermax: breaking the llm watermark detectability-robustness-quality trade-off. In *NeurIPS*, 2024.

Google. Google gemini. https://bard.google.com/chat/, 2024.

Chenchen Gu et al. On the learnability of watermarks for language models. *ICLR*, 2024.

Chenxi Gu et al. Watermarking pre-trained language models with backdooring. *arXiv*, 2022.

Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. Protecting intellectual property of language generation apis with lexical watermark. In *AAAI*, 2022a.

Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, et al. Cater: Intellectual property protection on text generation apis via conditional watermarks. *NeurIPS*, 2022b.

Dan Hendrycks, Collin Burns, et al. Measuring massive multitask language understanding, 2021.

Abe Bohan Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, and et al. Semstamp: A semantic watermark with paraphrastic robustness for text generation. *NAACL*, 2024.

Edward J Hu, Yelong Shen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 2022.

Zhengmian Hu, Lichang Chen, Xidong Wu, et al. Unbiased watermark for llms. *ICLR*, 2024.

Baizhou Huang and Xiaojun Wan. Waterpool: A watermark mitigating trade-offs among imperceptibility, efficacy and robustness. *arXiv*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, et al. Mistral 7b, 2023.

Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in llms. In *ICML*, 2024.

John Kirchenbauer, Jonas Geiping, et al. A watermark for large language models. In *ICML*, 2023.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *NeurIPS*, 36, 2024.

Kalpesh Krishna et al. Thieves on sesame street! model extraction of bert-based apis. *ICLR*, 2020.

Rohith Kuditipudi et al. Robust distortion-free watermarks for language models. *TMLR*, 2023.

Junyi L. et al. Pre-trained lms for text generation: A survey. *ACM Computing Surveys*, 2024.

Peixuan Li, Pengzhou Cheng, Fangqi Li, Wei Du, Haodong Zhao, et al. Plmmark: a secure and robust black-box watermarking framework for pre-trained language models. In *AAAI*, 2023.

Zhe Lin, Yitao Cai, and Xiaojun Wan. Towards document-level paraphrase generation with sentence rewriting and reordering. *EMNLP*, 2021.

Aiwei Liu, Leyi Pan, et al. An unforgeable publicly verifiable watermark for LLMs. In *ICLR*, 2023.

Aiwei Liu et al. A semantic invariant robust watermark for llms. *ICLR*, 2024.

Stephen Meisenbacher, Nihildev Nandakumar, and et al. A comparative analysis of word-level metric differential privacy: Benchmarking the privacy-utility trade-off. *LREC-COLING*, 2024.

Tomas Mikolov et al. Efficient estimation of word representations in vector space. *arXiv*, 2013.

George A Miller. Wordnet: a lexical database for english. *CACM*, 38(11):39–41, 1995.

Travis Munyer et al. Deeptextmark: A deep learning-driven text watermarking approach for identifying llm generated text. *IEEE Access*, 2024.

Georg Niess and Roman Kern. Stylometric watermarks for large language models. *arXiv*, 2024.

OpenAI. Openai api, 2024. https://openai.com/index/openai-api/.

Leyi Pan, Aiwei Liu, et al. Markllm: An open-source toolkit for llm watermarking. *arXiv*, 2024.

Kaiyi Pang et al. Adaptive and robust watermark against model extraction attack. *arXiv*, 2024.

Jie Ren et al. A robust semantics-based WM for LLM against paraphrasing. *NAACL*, 2024.

Yuki Takezawa et al. Necessary and sufficient watermark for llms. *arXiv*, 2023.

Hugo Touvron, Louis Martin, , et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

Qilong W. and Varun C. Bypassing LLM watermarks with color-aware substitutions. In *ACL*, 2024.

Eric Wallace, Mitchell Stern, and Dawn Song. Imitation attacks and defenses for black-box machine translation systems. *EMNLP*, 2020.

S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

Qiongkai X. et al. Student surpasses teacher: Imitation attack for black-box nlp apis. *COLING*, 2022.

Hengyuan Xu, Liyao Xiang, Xingjun Ma, Borui Yang, and Baochun Li. Hufu: A modality-agnositc watermarking system for pre-trained transformers via permutation equivariance. *arXiv*, 2024a.

Xiaojun Xu et al. Learning to wm llm-generated text via reinforcement learning. *arXiv*, 2024b.

Mingfu Xue, Zhiyu Wu, Yushu Zhang, and et al. Advparams: An active dnn intellectual property protection technique via adversarial perturbation based parameter encryption. *TETC*, 2022.

Xi Yang et al. Watermarking text generated by black-box language models. *arXiv*, 2023.

Hanlin Zhang, Benjamin L Edelman, Danilo Francati, Daniele Venturi, and et al. Watermarks in the sand: Impossibility of strong watermarking for generative models. *ICML*, 2024.

Jiliang Zhang et al. Apmsa: Adversarial perturbation against model stealing attacks. *TIFS*, 2023.

Xuandong Zhao et al. Provable robust watermarking for AI-generated text. In *ICLR*, 2024.