RESF: Regularized-Entropy-Sensitive Fingerprinting for Black-Box Tamper Detection of Large Language Models

Pingyi Hu[1†] **Xiaofan Bai**[1†] **Xiaojing Ma**[1*] **Chaoxiang He**[2*]
Dongmei Zhang[3] **Bin Benjamin Zhu**[3]
[1]Huazhong University of Science and Technology
[2]Shanghai Jiao Tong University [3]Microsoft Corporation
{xiaofanbai, pingyihu, linda}@hust.edu.cn
hechaoxiang@sjtu.edu.cn, {dongmeiz, binzhu}@microsoft.com

Abstract

The proliferation of Machine Learning as a Service (MLaaS) has enabled widespread deployment of large language models (LLMs) via cloud APIs, but also raises critical concerns about model integrity and security. Existing black-box tamper detection methods, such as watermarking and fingerprinting, rely on the stability of model outputs—a property that does not hold for inherently stochastic LLMs.

We address this challenge by formulating black-box tamper detection for LLMs as a hypothesis-testing problem. To enable efficient and sensitive fingerprinting, we derive a first-order surrogate for KL divergence—the entropy-gradient norm—to identify prompts most responsive to parameter perturbations. Building on this, we propose **Regularized Entropy-Sensitive Fingerprinting (RESF)**, which enhances sensitivity while regularizing entropy to improve output stability and control false positives. To further distinguish tampering from benign randomness, such as temperature shifts, RESF employs a lightweight two-tier sequential test combining support-based and distributional checks with rigorous false-alarm control.

Comprehensive analysis and experiments across multiple LLMs show that RESF achieves up to 98.80% detection accuracy under challenging conditions, such as minimal LoRA fine-tuning with five optimized fingerprints. RESF consistently demonstrates strong sensitivity and robustness, providing an effective and scalable solution for black-box tamper detection in cloud-deployed LLMs.

1 Introduction

The widespread adoption of Machine Learning as a Service (MLaaS) has greatly simplified the deployment and use of Deep Neural Networks (DNNs). In particular, cloud-hosted models—including Large-scale Language Models (LLMs) such as GPT or LLaMA—are increasingly accessed via APIs for applications ranging from virtual assistants to code generation. However, this deployment paradigm introduces a critical security concern: both model owners and end users must rely on trust that the cloud-hosted model has not been tampered with or substituted by an adversary (He et al., 2019).

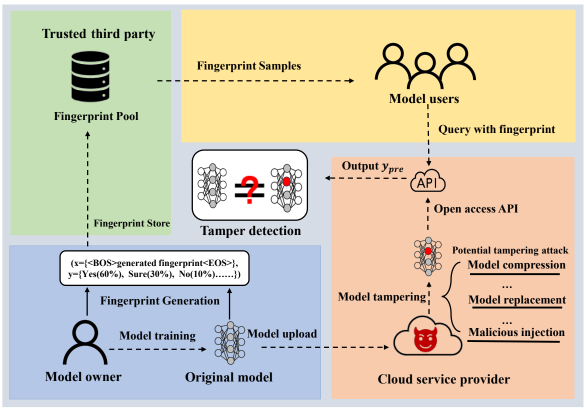

Figure 1: Illustration of tampering attacks on cloud-deployed LLMs and fingerprinting for tamper detection.

In practice, model tampering can manifest in various forms. For example, a dishonest service provider might covertly replace a high-performance model with a compressed or less accurate variant to reduce operational costs. Alternatively, an adversary could inject backdoors by fine-tuning the owner's original model on poisoned data, embedding malicious behaviors while maintaining normal performance on standard inputs. Consider a backdoored face recognition system: it may classify benign samples correctly, yet consistently misidentify any individual wearing a specific trigger—such as a particular style of glasses—as a predefined target. Such tampering often leaves the model seemingly functional on standard benchmarks while silently compromising its integrity and safety guarantees. As a result, detecting tam-

*Corresponding Authors. † Equal Contribution. All experimental work was performed by Pingyi Hu and Xiaofan Bai in Huazhong University of Science and Technology.

pering in cloud-deployed models under black-box access remains a critical and highly challenging open problem.

Substantial efforts have been devoted to addressing this concern in black-box settings, which align well with the MLaaS paradigm. Existing black-box model integrity verification methods primarily rely on *watermarking* or *fingerprinting* techniques (illustrated in Fig. 1). Among these, fingerprinting has attracted increasing attention due to its non-intrusive nature. Both approaches share a common underlying assumption: the *stability* of the model's input-output behavior—specifically, that the parameterized mapping function $f(\cdot)$ produces consistent outputs for identical inputs. This assumption generally holds for traditional deterministic classifiers, where repeated queries with the same input yield the same prediction. However, *the inherent stochasticity in the generation process of LLMs fundamentally undermines this premise*, rendering detection methods designed for deterministic models ineffective for LLM tamper detection.

In this paper, we introduce a novel and efficient fingerprinting method for black-box tamper detection in LLMs. We formulate tamper detection as a hypothesis-testing problem and establish a connection between the sensitivity of fingerprint samples and the KL divergence between output token distributions before and after potential tampering. To address the intractability of KL divergence without prior knowledge of tampering, we derive a *first-order surrogate*—the **entropy-gradient norm**, defined as $S(\mathbf{x}) = \|\nabla_\theta H(\mathbf{x};\theta_0)\|_2$, where $H(\mathbf{x};\theta_0)$ denotes the Shannon entropy (see Section 4.2). By leveraging the Cauchy–Schwarz inequality, we show that $S(\mathbf{x})$ provides a tight lower bound on the Fisher trace. This formulation eliminates computationally expensive second-order operations, significantly reducing overall cost.

Building on this, we propose **Regularized Entropy-Sensitive Fingerprinting (RESF)**, a method that enhances sensitivity to model tampering while promoting a sharp (low-entropy) original output distribution to explicitly control type-I (false-positive) errors. To distinguish temperature-induced variations from actual tampering, we design a **two-tier sequential testing framework**: *Rule E* flags early occurrences of out-of-support tokens, and *Rule P* performs an online Pearson goodness-of-fit test against a one-parameter temperature family. An α-spending strategy is adopted to keep the global false-positive rate below any user-specified level α. Finally, we provide both a theoretical analysis and a comprehensive experimental validation to demonstrate the effectiveness and efficiency of the proposed method.

Our major contributions can be summarized as follows:

1. We formalize black-box tamper detection for LLMs as a hypothesis-testing problem, establishing a rigorous theoretical foundation for verifying model integrity under the inherent stochasticity of LLM outputs.

2. We introduce a novel fingerprinting objective that employs a first-order surrogate for KL divergence, maximizing tampering sensitivity while explicitly controlling type-I (false positive) errors and addressing the challenge of output variability in LLMs.

3. We design a detection framework that integrates support-based checking (Rule E) with distributional goodness-of-fit testing (Rule P), enabling robust discrimination between malicious tampering and benign temperature-induced variations under rigorous α-spending control of the global false-alarm rate.

4. We provide theoretical analysis with explicit performance guarantees and validate RESF through extensive experiments across multiple LLMs and tampering scenarios. RESF achieves up to 98.80% detection accuracy under challenging conditions such as minimal LoRA fine-tuning with only five optimized fingerprints, and consistently demonstrates strong sensitivity and robustness, making it an effective and scalable solution for black-box LLM tamper detection.

2 Related Work

Model tampering detection. The objective of *tamper detection* is to ensure that a deployed model remains identical to the original version provided by the model owner, i.e., it has not been altered either maliciously or inadvertently. In practical deployments, models are often accessed through restricted interfaces such as cloud-based APIs or on-device services, where internal details—including architecture and parameters—are inaccessible. These constraints motivate verification in a *black-box* setting, where integrity must be assessed solely from the model's outputs on a limited set of queries without direct access to the model internals.

Black-box tamper detection techniques can be broadly divided into two categories. The first is *watermarking*, which embeds a fragile pattern into the model's parameters so that any modification disrupts this pattern (Yin et al., 2023). While watermarking can be effective, it is inherently invasive and often degrades the model's performance. The second category, *fingerprinting*, avoids altering the model by instead constructing a set of *sensitive inputs* whose outputs serve as the model's unique "fingerprint." Since our goal is to develop non-invasive detection mechanisms, we focus exclusively on fingerprinting-based approaches.

Fingerprinting for Tamper Detection of Classification Models. Fingerprinting for black-box tamper detection was pioneered by sensitive-sample fingerprinting (SSF) (He et al., 2019), first introduced for image classifiers. The central idea is to craft a set of inputs whose *predicted labels* are highly sensitive to small changes in model parameters. This sensitivity is typically achieved by (i) maximizing gradient-based sensitivity metrics (He et al., 2019; Docena et al., 2021; Kuttichira et al., 2022), (ii) placing inputs near decision boundaries (Wang et al., 2023; Xiaofan et al., 2024; Aramoon et al., 2021; Bai et al., 2025), or (iii) leveraging prediction variance across a local ensemble of models (He et al., 2024). During detection, any label flip in these fingerprinted samples is taken as evidence of model tampering.

Remark. Fingerprinting has also been extensively studied for intellectual property (IP) protection (Yang et al., 2022; Pan et al., 2022; Li et al., 2024, 2023; Zhang et al., 2024). Such methods aim to verify whether a suspect model originates from a protected one, typically to assert ownership. Although both application domains share methodological similarities, IP protection focuses on lineage verification, whereas tamper detection targets unauthorized modifications to a specific deployed model.

Challenges in applying existing fingerprinting to LLMs. Existing fingerprinting methods rely on a stable input–output mapping to verify model integrity. However, the inherent stochasticity of LLM outputs breaks this assumption, significantly reducing the effectiveness of fingerprinting techniques originally designed for deterministic models when applied to LLMs.

In summary, existing approaches either assume deterministic behavior or target intellectual property (IP) protection rather than tamper detection. To address the unique challenges posed by LLMs, we introduce **RESF**, a fingerprinting scheme tailored for black-box tamper detection of LLMs.

3 Threat Model

Aligning with existing works and as illustrated in Fig. 1, we adopt a white-box assumption for fingerprint generation and a black-box assumption for tamper detection. Specifically, during fingerprint prompt generation, we assume that the model parameters and corresponding gradients are fully accessible. This is reasonable since fingerprint samples are generated by the model owner. In contrast, tampering detection operates under a black-box setting, where only API access to the suspect model and its text outputs is available.

Following (Wang et al., 2023; Xiaofan et al., 2024; He et al., 2024), we further assume a trusted third party securely stores and distributes fingerprint samples for *public tamper detection*, while the cloud service provider may be untrustworthy and could tamper with the uploaded LLM. Additionally, adversaries may attempt to acquire or craft fingerprint samples in an effort to evade detection.

4 Effective Fingerprinting for LLMs

Let $\theta_0 \in \Theta \subset \mathbb{R}^d$ denote the parameters of the language model to be protected. The model generates output tokens from a finite vocabulary V, where $|V| \geq 2$. Given an input prompt $\mathbf{x} = (x_1, \ldots, x_n)$, the model defines a conditional probability distribution $p_\theta(\cdot \mid \mathbf{x})$ over the next output token. We define a *fingerprint sample* as a carefully constructed sequence of input tokens designed to be highly sensitive to changes in the model parameters. For a given fingerprint sample $\mathbf{x}$, the probability of observing output y is $p_\theta(y \mid \mathbf{x})$.

For clarity and without loss of generality, we focus on the first generated token as the tamper-detection token throughout this section. The analysis naturally generalizes to cases where multiple tokens or positions are used for detection.

4.1 Problem Formulation

Tamper detection can be cast as a hypothesis-testing problem, where tampering corresponds to a perturbation of the original parameters θ_0:

$$\mathcal{H}_0 : \theta = \theta_0 \quad \text{vs.} \quad \mathcal{H}_1 : \theta = \theta_0 + \Delta, \tag{1}$$

where $\Delta \neq 0$ denotes the parameter shift introduced by model tampering. In this setting, a single query to the model with a fingerprint sample yields an output token $y \sim p_\theta(\cdot \mid \mathbf{x})$.

Based on the *Neyman–Pearson Lemma* (Neyman and Pearson, 1933), for a fixed Δ, among all input samples, the one that maximizes the *expected* log-likelihood ratio $\log \frac{p_{\theta_0}(\cdot|\mathbf{x})}{p_{\theta_0+\Delta}(\cdot|\mathbf{x})}$ under $\mathcal{H}_0$ (hence minimizes the optimal β-error) maximizes the Kullback–Leibler divergence $D_{\mathrm{KL}}\big(p_{\theta_0} \| p_{\theta_0+\Delta}\big)$. More details can be found in Appendix A.

4.2 First-Order Approximation of KL Divergence

As we do not have any prior on Δ, by expanding the KL divergence in a Taylor series at θ_0, we have

$$D_{\mathrm{KL}}\big(p_{\theta_0} \| p_{\theta_0+\Delta}\big) = \tfrac{1}{2}\Delta^\top F_{\theta_0}(\mathbf{x})\Delta + O(\|\Delta\|^3), \tag{2}$$

where $F_{\theta_0}(\mathbf{x}) \in \mathbb{R}^{d\times d}$ is the *Fisher information matrix* with entries $F_{ij} = \sum_{y\in V} \partial_{\theta^i} \log p_{\theta_0}(y \mid \mathbf{x})\, \partial_{\theta^j} \log p_{\theta_0}(y \mid \mathbf{x})\, p_{\theta_0}(y \mid \mathbf{x})$. See Appendix B for details.

By Eq. 2, to maximize the outputs' difference before and after model tampering, we can maximize the KL divergence by maximizing the $||F_{\theta_0}(\mathbf{x})||_2$ (equivalent to $\mathrm{tr}(F_{\theta_0})$). However, directly computing $\mathrm{tr}(F_{\theta_0})$ requires evaluating and summing all second derivatives—or equivalently all pairwise products of first derivatives—for every token y.

Lemma 4.1 (Cauchy–Schwarz lower bound)**.** *For a finite vocabulary and differentiable token distribution p_θ, maximizing the entropy gradient norm $\|\nabla_\theta H(\mathbf{x};\theta_0)\|_2$ maximizes a Cauchy–Schwarz lower bound on $\mathrm{Tr}(F_{\theta_0})$, i.e.,*

$$\mathrm{Tr}\big[F_{\theta_0}(\mathbf{x})\big] \;\geq\; \frac{\|\nabla_\theta H(\mathbf{x};\theta_0)\|_2^2}{(\log|V|)^2}. \tag{3}$$

Proof. Recall $\nabla_\theta H = -\sum_y \big(1 + \log p_\theta(y \mid \mathbf{x})\big)\, \nabla_\theta p_\theta(y \mid \mathbf{x})$. Let $a_y = 1 + \log p_\theta(y \mid \mathbf{x})$ and $b_y = \|\nabla_\theta p_\theta(y \mid \mathbf{x})\|_2$. Observing $|a_y| \leq \log|V|$ (because $p \leq 1$) and applying Cauchy–Schwarz, we have

$$\|\nabla_\theta H\|_2^2 \leq \Big(\sum_y a_y^2 p_\theta(y \mid \mathbf{x})\Big)\Big(\sum_y \frac{\|\nabla_\theta p_\theta(y \mid \mathbf{x})\|_2^2}{p_\theta(y \mid \mathbf{x})}\Big). \tag{4}$$

The second factor equals $\mathrm{tr}(F_{\theta_0})$ by definition, while the first factor is bounded above by $(\log|V|)^2$, establishing Lemma 4.1. More details are in Appendix C. $\square$

Lemma 4.1 shows that $\mathrm{tr}(F_{\theta_0})$—and hence the right first term in Eq. 2—can be *lower-bounded* via the easily computable entropy gradient norm $\|\nabla_\theta H(\mathbf{x};\theta_0)\|_2$. Based on this conclusion, for a fingerprint sample $\mathbf{x}$, we can provably increase the lower bound of the output's divergence $D_{\mathrm{KL}}\big(p_{\theta_0} \| p_{\theta_0+\Delta}\big)$ by maximizing $\|\nabla_\theta H(\mathbf{x};\theta_0)\|_2$ when the model tampering Δ is applied to the protected model θ_0.

Definition 4.2 (Fingerprint Sensitivity)**.** By Lemma 4.1, the *sensitivity* of a fingerprint can be defined as:

$$S(\mathbf{x}) \;=\; \|\nabla_{\theta_0} H(\mathbf{x};\theta_0)\|_2, \tag{5}$$

where $H(\mathbf{x};\theta_0) = -\sum_y p_{\theta_0}(y \mid \mathbf{x}) \log p_{\theta_0}(y \mid \mathbf{x})$ is the Shannon entropy.

4.3 Regularized-Entropy-Sensitive Fingerprinting

A fingerprint sample with large $S(\mathbf{x})$ alone is insufficient: If its entropy at state $\theta = \theta_0$ is high, the generated token may fluctuate even under $\mathcal{H}_0$, causing false alarms. To tackle this issue, since $H(\mathbf{x};\theta_0) \in [0, ln|V|]$ is bounded, we therefore introduce a regularized entropy term to ensure that $H(\mathbf{x};\theta_0)$ is as low as possible. The objective function for fingerprint sample $\mathbf{x}$ optimization of our **Regularized-Entropy-Sensitive Fingerprinting (RESF)** can be formalized as

$$\arg\max_{\mathbf{x}} \; J(\mathbf{x}) = S(\mathbf{x}) - \lambda \cdot H(\mathbf{x};\theta_0), \tag{6}$$

where $\lambda > 0$ is a regularization coefficient.

5 Theoretical Analysis of RESF

In the previous section, we demonstrated how the RESF objective promotes fingerprint samples with a large gradient norm $S(\mathbf{x})$ while constraining the entropy $H(\mathbf{x};\theta_0)$. We now establish the practical detection guarantees afforded by these properties. Specifically, the following results translate the two design goals of Eq. 6:

(i) *Sharp original entropy* (small H), and

(ii) *Large slope* (large S),

into explicit bounds on type-I (false-positive) and type-II (false-negative) error probabilities. These bounds provide an information-theoretic justification for using $J(\mathbf{x})$ in Eq. 6 as the optimization criterion.

Let $\mathbf{x}^\star$ be an optimizer of Eq. (6). Define $y^\star = \arg\max_{y \in V} p_{\theta_0}(y \mid \mathbf{x}^\star)$, $p^\star = p_{\theta_0}(y^\star \mid \mathbf{x}^\star)$, $\varepsilon = H(\mathbf{x}^\star; \theta_0)$, and $G = \nabla_{\theta_0} p_{\theta_0}(y^\star \mid \mathbf{x}^\star)$. All logarithms are natural unless stated otherwise.

Proposition 5.1 (False-positive probability)**.** *Under $\mathcal{H}_0$, the probability that the observed token Y differs from $y^\star$ satisfies*

$$\Pr_{\mathcal{H}_0}[Y \neq y^\star] \leq \frac{\varepsilon}{\log |V|}. \tag{7}$$

Proof. Let $q_i = p_{\theta_0}(y_i \mid \mathbf{x}^\star)$, with $q_1 = p^\star$ and $\sum_{i=1}^{|V|} q_i = 1$. The worst case (maximizing $1 - p^\star$ for fixed entropy ε) occurs when the remaining mass is distributed uniformly: $q_2 = \cdots = q_{|V|} = (1 - p^\star)/(|V| - 1)$. Substituting into the entropy constraint,

$$\varepsilon = -p^\star \log p^\star - (1 - p^\star) \log\left(\frac{1 - p^\star}{|V| - 1}\right), \tag{8}$$

and using $\log\left(\frac{1-p^\star}{|V|-1}\right) \geq -\log |V|$, we obtain $\varepsilon \geq (1 - p^\star) \log |V|$, i.e., $1 - p^\star \leq \varepsilon / \log |V|$. □

Interpretation. The bound scales linearly with original entropy ε, so minimizing ε (as in the RESF objective) directly reduces the false-alarm rate without loss of sensitivity.

Proposition 5.2 (First-order drop under tampering)**.** *Assuming each coordinate gradient of $p_\theta(y \mid \mathbf{x})$ $\nabla_\theta p_\theta(y \mid \mathbf{x})$ is L-Lipschitz:*

$$\left\|\nabla_\theta p_{\theta'}(y \mid \mathbf{x}) - \nabla_\theta p_{\theta''}(y \mid \mathbf{x})\right\|_2 \leq L \left\|\theta' - \theta''\right\|_2, \tag{9}$$

where L is the Lipschitz constant and $\theta', \theta'' \in \Theta$. Then for model tampering Δ with $\|\Delta\|_2 = \eta$,

$$p_{\theta_0 + \Delta}(y^\star \mid \mathbf{x}^\star) \leq p^\star - \eta \|G\|_2 + \frac{1}{2} L \eta^2. \tag{10}$$

In particular, if $0 < \eta \leq \dfrac{\|G\|_2}{L}$, the right-hand side is at most $p^\star - \frac{1}{2}\eta \|G\|_2$. Using Lemma 4.1, $\|G\|_2 \geq \dfrac{S(\mathbf{x}^\star)}{\log |V|}$, so the probability reduction is at least $\delta = \dfrac{\eta\, S(\mathbf{x}^\star)}{2 \log |V|}$.

Proof. By the mean-value theorem, there exists $\widetilde{\theta}$ on the line segment $\theta_0 + t\Delta$ ($t \in [0, 1]$) such that $p_{\theta_0+\Delta}(y^\star) - p^\star = \langle \nabla_\theta p_\theta(y^\star)\big|_{\widetilde{\theta}}, \Delta \rangle$. Add and subtract G to obtain

$$\langle G, \Delta \rangle + \langle \nabla_\theta p_\theta\big|_{\widetilde{\theta}} - G, \Delta \rangle. \tag{11}$$

The first term is at least $-\eta \|G\|_2$ (adversary chooses the worst inner-product sign), while the second is bounded by $\frac{1}{2} L \eta^2$ due to L-Lipschitzness of the gradient and the fact that $\|\widetilde{\theta} - \theta_0\| \leq \eta$. Rearranging yields the claimed inequality. In this paper, we focus on reporting the empirical performance of RESF, rather than estimating L or empirically validating the bound, since precisely estimating L is computationally intensive and requires substantial computing resources. □

Practical Meaning. The bound implies that any sufficiently large perturbation will reduce the likelihood of the observed token below $p^\star - \delta$, creating a detectable statistical gap. Importantly, this gap increases linearly with both the tamper magnitude η and the sensitivity $S(\mathbf{x}^\star)$, formalizing the intuition that prompts with "steep cliffs" in the probability landscape are most effective for detection.

We conclude that high-sensitivity, low-entropy prompts are not merely heuristically desirable; they provably improve the ROC curve of any black-box tamper detector.

6 Detailed Description of RESF

6.1 Fingerprint Generation

We solve Eq. 6 using iterative *gradient ascent*:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \eta \cdot \nabla_{\mathbf{x}_i} J(\mathbf{x}_i) \tag{12}$$

where $\mathbf{x}_i$ denotes the fingerprint sample's state at i-th iteration and η is the learning rate. This process generates a pool of sensitive fingerprint samples, from which we select those with the high sensitivity and lowest entropy to form the final fingerprint set.

Specifically, we employ a two-stage optimization procedure. In the first stage, we optimize a continuous soft prompt matrix in the embedding space by maximizing Eq. 6 via gradient ascent. The soft prompt is iteratively updated to maximize the entropy gradient norm while minimizing output entropy, with regularization and clipping for stability.

In the second stage, we discretize the optimized soft prompt into a valid token sequence using the Greedy Coordinate Gradient (GCG) algorithm, which replaces embedding vectors with the closest vocabulary tokens to best preserve the fingerprint samples' sensitivity generated with RESF. This ensures that the final fingerprint samples remain both effective and composed of valid tokens.

6.2 Tamper Detection

Given the fingerprint samples generated as described in Section 6.1, black-box tamper detection for LLMs remains challenging due to the inherent randomness in their outputs. As a result, changes in the output distribution may be caused either by actual model tampering or by the stochastic nature of the LLM itself. Distinguishing between these two sources of variation is central to effective tamper detection using generated fingerprints.

Output Randomness. Traditional fingerprinting methods for deterministic classification models simply record the top-1 label as the fingerprint, since the input-output mapping is stable. However, this approach is inadequate for LLMs, whose outputs are inherently probabilistic. To address this, we compare the next-token output distribution of a fingerprint sample $\mathbf{x}$ from the original protected model, $\mathbf{p}_o = \underset{K}{Top}(p(y|\mathbf{x};\theta_0)) = \{p_o^1, p_o^2, \ldots, p_o^K\}$, with the observed output distribution from the model under test, $\mathbf{p}_t = \{(Tok_t^T;\theta;T_{max} \geq K\} = \{Tok_t^1, Tok_t^2, \ldots, Tok_t^i, \ldots, Tok_t^{T_{max}}\}$ where T_{max} is the query budget and Tok_t^i denotes the i-th-round returned tokens of the sequential test queries, to determine whether tampering has occurred (i.e., whether $\theta = \theta_0$). Here, $\underset{K}{Top}(\cdot)$ denotes the top-K most probable tokens in the output distribution. $\mathbf{p}_o$ is obtained from the original model at a high temperature within the practical range, to capture the maximum expected output variability during deployment.

Model Tampering vs. Temperature Rescaling. Given a fingerprint sample $\mathbf{x}$ and its next-token distribution collected from the original protected model $\mathbf{p}_o = (p_o^1, \ldots, p_o^K)$, we analyze the tokens generated by the model under test to determine, within a given query budget, whether

(i) the test model is simply using a *different softmax temperature*, or

(ii) the model's *weights have been tampered* with.

Key Intuition. Temperature rescaling of $\mathbf{p}_o$ defines a one-parameter family:

$$q(\beta) = (q_i(\beta))_{i=1}^K, \quad q_i(\beta) = \frac{(p_o^i)^\beta}{\sum_{j=1}^K (p_o^j)^\beta}, \tag{13}$$

where $\beta \geq 1$ is the scaling parameter. Any observed token distribution from the test model that deviates from this family indicates tampering with the underlying model parameters, rather than a mere change in temperature.

Two–Tier Sequential Test. We propose a two–tier sequential test for each fingerprint sample $\mathbf{x}$ to distinguish model tampering from temperature rescaling. It integrates two complementary detectors:

Rule E (*support check*) Triggers as soon as a token outside the recorded top–K support appears "too early." **Rationale:** Model tampering can introduce new high-probability tokens, whereas temperature changes cannot assign probability mass to previously unseen tokens.

Rule P (*shape check*) Performs a sequential χ^2 goodness–of–fit test against the null family (13), estimating the nuisance parameter β on-the-fly. **Rationale:** Even if the support remains unchanged, tampering can alter the *relative ordering* of probabilities, which the Pearson statistic can reliably detect, even for subtle shifts.

The two rules are combined using an *α–spending* schedule to ensure that the overall false-alarm probability remains below user-specified thresholds α_E and α_P for Rule E and Rule P, respectively.

Detection Details. We detect tampering by monitoring the test model's outputs on a fingerprint sample and applying two sequential checks: one for unexpected tokens and one for changes in the top-K probability distribution. The procedure is below:

1. **Reference Recording.** For the original model, record the token probabilities $p = (p_i)_{i=1}^K$ at the high practical temperature, the indices C of the top-K tokens under p, and the total probability mass outside the top-K, $p_{\text{out}} = 1 - \sum_{i\in C} p_i$.

2. **Online Querying and Count Definitions.** For a fingerprint sample, query the test model up to T_{max} times. For each query $t = 1, 2, \ldots$, receive token y_t. Maintain counts $n_i(t) = \sum_{s=1}^t \mathbf{1}[y_s = i]$ for $i \in C$, and $n_{\text{out}}(t) =$

$\sum_{s=1}^{t} \mathbf{1}[y_s \notin C]$, representing the number of times each top-K token and any out-of-support token has appeared, respectively.

3. **Rule E (Support Check).** If the first out-of-support token occurs at step $t \leq N_E$, where

$$N_E = \left\lceil \frac{\log(1-\alpha)}{\log(1-p_{\text{out}})} \right\rceil,$$

immediately declare *tampered*.

4. **Rule P (Shape Check).**
Starting at $t = N_{\min}$, at each step:

   (i) *Fit* the sharpening exponent $\widehat{\beta}$ by solving the score equation for the log-likelihood of counts $\{n_i(t)\}$ (see Appendix D.1) under $q_i(\beta) \propto p_i^{\beta}$.

   (ii) *Compute* expected counts $e_i(t) = t \cdot q_i(\widehat{\beta})$ and Pearson's statistic $\chi^2(t) = \sum_{i \in C \cup \{\text{out}\}} (n_i(t) - e_i(t))^2 / e_i(t)$.

   (iii) *Compare* to the critical value

$$c_t = \chi^2_{\nu}\Big(1 - \frac{\alpha}{M}\Big),$$
$$\nu = K - 1, \; M = T_{\max} - N_{\min} + 1. \tag{14}$$

   where $N_{\min}$ is the minimum number of queries before starting Rule P. If $\chi^2(t) > c_t$, stop and declare *tampered*. (see Appendix D.2)

5. **No Alarm.**
If neither Rule E nor Rule P fires by $t = T_{\max}$, conclude *no evidence of tampering*.

We summarize the tamper detection scheme in Algorithm 1 (Appendix D). Further details on estimating $\hat{\beta}$ via log-likelihood and the decision process in step (iii) of **Rule P** are provided in Appendix D.1 and Appendix D.2, respectively.

7 Experiments

7.1 Experimental Settings

Models and tempering types. In our experiments, we evaluate the effectiveness of the proposed fingerprint method using several widely-used open-source large language models (LLMs), specifically Qwen2.5-0.5B (Team, 2024), Qwen2.5-7B (Team, 2024), LLaMA3.2-1B (Touvron et al., 2023), LLaMA3-8B (AI@Meta, 2024), and Mistral-v3-7B (Jiang et al., 2023). These models represent various scales and architectures, ensuring comprehensive validation of our fingerprint's robustness and generalization capabilities.

Our experiments involve detection against four types of model tampering scenarios: model fine-tuning, model poisoning, model compression, and model replacement. For the fine-tuning scenario, each model is fine-tuned on 5K samples from Alpaca dataset (Taori et al., 2023) for one epoch. The poisoning scenario involves fine-tuning models on a dataset containing 10% (500 samples) poisoned samples, method from (Xu et al., 2024), for one epoch. Both fine-tuning and poisoning scenarios utilize the LLaMA-Factory framework (Zheng et al., 2024). In these fine-tuning processes, we explore three distinct configurations: full parameter fine-tuning (noted as SFT), freeze-layer fine-tuning (freezing all layers except the last three, noted as Freeze), and Low-Rank Adaptation (LoRA). For model compression, we employ int 4 and int 8 quantization by bitsandbytes (Dettmers et al., 2022).

Fingerprint generation and detection configurations. The fingerprint generation consists of two stages: Stage 1 optimizes each fingerprint for 500 steps using the Adam optimizer (learning rate 0.01), followed by 200 additional steps in Stage 2. Fingerprints are randomly initialized from the tokenizer and details are shown in Fig. 4 in Appendix E. For each model, we generate 100 unique fingerprints. During detection, we set $T_{\max} = 100$, $N_{\min} = 30$, $\alpha = 0.05$, and $\alpha_E = 0.01$, and randomly select 1 or 5 fingerprints for evaluation. We record the element of output next-token distribution with confidence larger than 0.01 from the original model while temperature is set to 0.7, top-k is set to 50, and nucleus-p is set to 0.99. Each fingerprint is evaluated for 10 times.

All experiments are implemented in PyTorch with Hugging Face Transformers, using 2 NVIDIA H20 GPUs.

7.2 Detection Effectiveness

Overall experiment results. Table 1 reports the tamper detection performance of RESF across a range of model architectures and different tampering types. With a single fingerprint, RESF achieves a detection rate up to 89.40%, and this improves to nearly 100% when using five fingerprints, demonstrating strong sensitivity to a variety of tampering methods. These results highlight RESF's generalizability and robustness under different model tampering. Across fine-tuning, poi-

Table 1: Average temper detection rate (%) with 1 or 5 fingerprints.

Models		Qwen2.5-0.5B		Qwen2.5-7B		LLaMA3.2-1B		LLaMA3-8B		Mistral-v3-7B	
Tempering Types/ fingerprint numbers		1	5	1	5	1	5	1	5	1	5
Finetuning	SFT	78.79	100.00	80.30	100.00	87.72	100.00	81.82	99.99	75.27	99.99
	Freeze	59.09	99.99	72.73	99.99	73.41	99.99	50.35	99.99	70.35	99.99
	LoRA	37.88	98.78	37.05	98.07	64.12	99.99	39.47	99.04	32.01	97.78
Model Poison	SFT	89.40	100.00	84.15	100.00	88.42	100.00	75.76	99.99	76.55	99.99
	Freeze	62.12	99.99	72.73	99.99	73.76	99.99	58.58	99.99	70.36	99.99
	LoRA	42.25	99.99	35.76	96.87	54.97	99.99	34.85	96.39	42.06	98.93
Model compression	8bit	45.45	99.02	72.72	99.99	82.25	99.99	37.88	97.03	61.67	99.98
	4bit	86.36	100.00	80.30	100.00	83.75	100.00	62.12	99.99	66.57	99.99
Model replacement	Qwen2.5-0.5B	∼	∼	72.85	99.99	83.12	99.99	85.25	100.00	80.23	100
	Qwen2.5-7B	64.27	99.99	∼	∼	89.23	100.00	80.16	100.00	84.12	100
	LLaMA3.2-1B	92.31	100.00	91.23	100.00	∼	∼	80.28	100.00	88.17	100
	LLaMA3-8B	93.32	100.00	89.12	100.00	77.12	99.99	∼	∼	85.24	100
	Mistral-v3-7B	88.65	100.00	85.24	100.00	85.23	100.00	82.13	100.00	∼	∼
False Positive Rate		1.52	0.54	0.63	0.33	0.68	0.46	0.77	0.44	0.24	0.01

Table 2: Tamper detection rates (%) under varying numbers of detection tokens for different poisoning methods on Qwen2.5-0.5B using a single fingerprint sample.

Detection tokens	1	2	3
SFT	89.40	93.24	95.57
Freeze	62.12	78.61	86.23
LoRA	42.25	56.23	68.87

Table 3: Tamper detection rates (%) of different baseline methods under various poisoning attacks on Qwen2.5-0.5B using a single fingerprint sample.

	NSS	SFF	PublicCheck	RESF
SFT	23.74	15.38	11.13	89.40
Freeze	15.32	13.13	14.23	62.12
LoRA	14.24	12.10	12.14	42.25

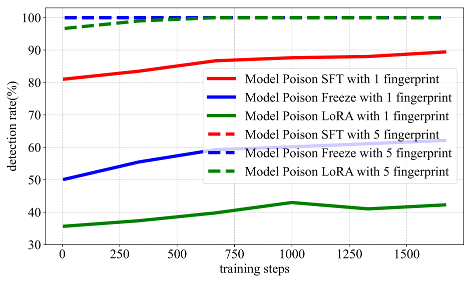

Figure 2: Tamper detection rates (%) across different training steps under various poisoning attacks on Qwen2.5-0.5B.

soning, and quantization-based compression, RESF consistently detects even minor modifications, including LoRA or Freeze-based tuning.

Tamper detection on different training steps. Fig. 2 illustrates the tamper detection performance of a single fingerprint sample across different stages of model fine-tuning. Notably, after just 10 steps of fine-tuning with a poisoned dataset, our method achieves a detection success rate of 37.65% using a single fingerprint. When combining five fingerprints, the detection rate rises sharply to 98.80%, which is already a practically reliable level. These results demonstrate that our approach is highly sensitive to subtle model tampering.

Tamper detection with existing baselines. We compare RESF to three baselines: SSF (He, 2021) and PublicCheck (Wang et al., 2023)—both designed for deterministic classifiers—and a non-sensitive sample (NSS) variant that uses our detector without sensitivity optimization. To apply SSF/PublicCheck to LLMs, we adapt them by converting the token distribution into a pseudo-class label via the highest-probability token at a fixed evaluation position. As shown in Table 3, RESF attains high detection rates, while SSF and PublicCheck exhibit elevated false positives because token-level stochasticity in LLMs frequently flips the top-1 token, causing intact models to be incorrectly flagged. Also, RESF surpasses NSS, demonstrating the gains from optimized sensitive fingerprint samples.

Tamper detection with different numbers of detection tokens. Since RESF explicitly minimizes output entropy, the model's output distribution often assigns extremely high probability to its top-1 prediction. Leveraging this property, we treat the predicted top-1 token as the output token, extend the context with this token, and then optimize the following token's distribution using the same objective. Repeating this procedure autoregressively yields multiple detection tokens, each conditioned on the previously selected tokens and optimized to remain highly sensitive to model parameter pertur-

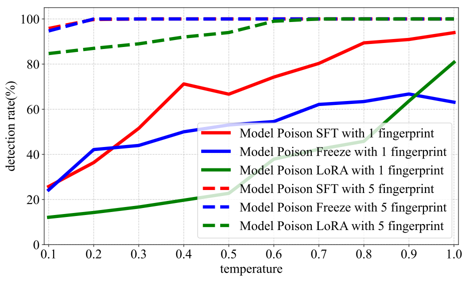

Figure 3: Tamper detection rates (%) under different generation temperatures for various poisoning attacks on Qwen2.5-0.5B.

Table 4: Tamper detection rates (%) with varying nucleus-p under different poisoning attacks on Qwen2.5-0.5B using a single fingerprint sample.

nucleus-p	0.7	0.8	0.9	0.99
SFT	89.45	90.45	91.75	89.40
Freeze	55.62	61.17	58.14	62.12
LoRA	39.99	41.15	39.92	42.25

bations.

Table 2 reports detection rates for different numbers of optimized detection tokens. As expected, increasing the number of detection tokens substantially boosts tamper detection success rates. However, this auto-regressive optimization process introduces proportional computational cost, since each additional token requires a full optimization cycle based on the previously fixed tokens. This highlights a trade-off between detection performance and efficiency.

Tamper detection under different randomness settings. Fig. 3 and Tables 4 and 5 show the effect of mismatched decoding configurations between fingerprint recording and tamper detection, including variations in temperature, nucleus-p, and top-k. By default, fingerprints are generated with temperature 0.7, nucleus-$p = 0.99$, and top-$k = 50$.

The results show that our detection performance remains robust and consistent under both nucleus sampling and top-k sampling variations, confirming the method's effectiveness even when decoding strategies differ. However, the results also indicate that deliberately lowering the temperature during generation (e.g., to 0.3 or 0.1) can reduce detection success rates, as the output distribution becomes more peaked and less variable. Even so, at a low temperature of 0.1, our method still achieves above 84.73% detection rate with 5 fingerprints.

Finally, our sequential hypothesis testing procedure incorporates a temperature-fitting mechanism (Section 6.2), which effectively adapts to moderate shifts in temperature. This enables the detector to differentiate tampering from benign temperature misconfigurations.

Table 5: Tamper detection rates (%) with varying top-k values under different poisoning attacks on Qwen2.5-0.5B using a single fingerprint sample.

nucleus-p	20	50	70	100
SFT	90.78	89.40	88.88	88.84
Freeze	63.56	62.12	61.17	60.71
LoRA	45.12	42.25	41.74	37.88

Table 6: Tamper detection rate (%) with different objective function for different model poisoning on Qwen2.5-0.5B with 1 fingerprint.

	Sensitivity–$S(\mathbf{x})$	Entropy–$H(\mathbf{x};\theta_0)$	RESF
SFT	64.28	42.85	89.40
Freeze	50.13	35.16	62.12
LoRA	28.57	16.13	42.25

7.3 Ablation Study

Our objective function consists of two key components: the entropy gradient norm $S(\mathbf{x})$ for sensitivity, and the entropy term $H(\mathbf{x};\theta_0)$ for stability. To understand their contributions, we conduct an ablation study by removing each term separately during fingerprint optimization. The results, summarized in Table 6, show that optimizing only for $S(\mathbf{x})$ leads to high sensitivity but also high output randomness, optimizing solely for low entropy yields stable but less sensitive fingerprints, reducing detection effectiveness.

8 Conclusion

In this paper, we propose RESF, a principled and efficient fingerprinting framework for black-box tamper detection of large language models. By leveraging a first-order surrogate for KL divergence and a two-tier sequential testing strategy, RESF robustly differentiates model tampering from benign temperature shifts while strictly controlling false positives. Extensive experiments across multiple LLMs and attack scenarios show that RESF achieves up to **98.80%** detection accuracy under challenging conditions such as minimal LoRA fine-tuning with only five optimized fingerprints, consistently demonstrating strong sensitivity and robustness. These results highlight RESF as an effective and scalable solution for ensuring the integrity of LLMs in real-world MLaaS deployments.

9 Limitations

False positive rate under temperature mismatch. A limitation of RESF is its increased false positive rate when the original (non-tampered) model operates at a higher temperature than that used during fingerprint recording. As shown in Table 7, as the temperature rises, the output distribution flattens, allowing low-confidence tokens to gain probability mass. This increases the likelihood that tokens fall outside the originally recorded top-K support, triggering false alarms under Rule E in our Two-Tier Sequential Test.

Table 7: False positive rate (FPR) of RESF fingerprints, generated at temperature 0.7, evaluated on the original model under varying softmax temperature settings.

Temperature	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
FPR (%)	0.03	0.17	0.52	0.47	0.51	1.37	1.53	1.76	1.61	2.12

In practice, however, this issue is mitigated by recording fingerprint distributions at a reasonably high temperature (e.g., 0.7 in our experimental settings), which provides tolerance against typical benign temperature shifts. As evidenced by the experimental results, the false positive rate remains at a low level with increasing temperature, such as 2.12% when the temperature is set to 1.0.

Efficiency On average, generating 100 fingerprints per model (7B) takes approximately 8 hours using 2 NVIDIA H20 GPUs. The long runtime for generating fingerprints arises from the need to use a batch size of 1 during fingerprint generation. This is because our loss function requires computing second-order derivatives for each individual sample during optimization. In current deep learning frameworks (e.g., PyTorch, TensorFlow), the first-order gradient is computed as an average across the batch, and per-sample first-order gradients are not directly accessible for subsequent second-order derivative calculations. As a result, we must generate one fingerprint at a time (i.e., batch size = 1), rather than generating multiple fingerprints simultaneously with a larger batch size, which leads to the observed overhead. We recognize the importance of improving this process and plan to address it in future work.

Anomaly detection. Fingerprint prompts generated by our method tend to exhibit low naturalness, as indicated by their extremely high perplexity. For example, a typical 20-token fingerprint shows an average perplexity of 5,728.89 on Qwen2.5-0.5B, whereas natural texts of the same length from Alpaca average a much lower perplexity of 21.67. This stark discrepancy makes fingerprint queries susceptible to simple anomaly detection and filtering by adversaries.

To mitigate this risk, we adopt a strategy called *contextual wrapping*. In this approach, each fingerprint is divided into four segments of five tokens each and embedded at equal intervals within 100 tokens of natural instruction-following text sampled from the Alpaca dataset. The resulting 120-token sequence serves as the fingerprint, with the four segments jointly optimized to maximize sensitivity. This wrapping process significantly lowers the average perplexity to 32.19, allowing fingerprints to blend more naturally into typical input distributions while maintaining their robustness.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grant No. 62272175), the Major Research Plan of Hubei Province (Grant/Award No. 2023BAA027), the Key Research & Development Plan of Hubei Province of China (Grant No. 2024BAB049), and the project of Science, Technology and Innovation Commission of Shenzhen Municipality of China (Grant No. GJHZ20240218114659027).

References

AI@Meta. 2024. Llama 3 model card.

Omid Aramoon, Pin-Yu Chen, and Gang Qu. 2021. Aid: Attesting the integrity of deep neural networks. In *Proceedings of 2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 19–24.

Xiaofan Bai, Shixin Li, Xiaojing Ma, Bin Benjamin Zhu, Dongmei Zhang, and Linchen Yu. 2025. Sdbf: Steep-decision-boundary fingerprinting for hard-label tampering detection of dnn models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 29278–29287.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Amel Nestor Docena, Thomas Wahl, Trevor Pearce, and Yunsi Fei. 2021. Sensitive samples revisited: Detecting neural network attacks using constraint solvers. *arXiv preprint arXiv:2109.03966*.

Chaoxiang He, Xiaofan Bai, Xiaojing Ma, Bin B. Zhu, Pingyi Hu, Jiayun Fu, Hai Jin, and Dongmei Zhang.

2024. Towards stricter black-box integrity verification of deep neural network models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 9875–9884, New York, NY, USA. Association for Computing Machinery.

Zecheng He. 2021. Sensitive sample fingerprinting. `https://github.com/zechenghe/Sensitive_Sample_Fingerprinting`. Last accessed Jan. 25, 2022.

Zecheng He, Tianwei Zhang, and Ruby Lee. 2019. Sensitive-sample fingerprinting of deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4729–4737.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Deepthi Praveenlal Kuttichira, Sunil Gupta, Dang Nguyen, Santu Rana, and Svetha Venkatesh. 2022. Verification of integrity of deployed deep learning models using bayesian optimization. *Knowledge-Based Systems*, 241:108238.

Shen Li, Liuyi Yao, Jinyang Gao, Lan Zhang, and Yaliang Li. 2024. Double-i watermark: Protecting model copyright for llm fine-tuning. *arXiv preprint arXiv:2402.14883*.

Zongjie Li, Chaozheng Wang, Shuai Wang, and Cuiyun Gao. 2023. Protecting intellectual property of large language model-based code generation apis via watermarks. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 2336–2350.

Jerzy Neyman and Egon Sharpe Pearson. 1933. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337.

Xudong Pan, Yifan Yan, Mi Zhang, and Min Yang. 2022. Metav: A meta-verifier approach to task-agnostic model fingerprinting. In *Proceedings of 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 1327–1336. ACM.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Shuo Wang, Sharif Abuadbba, Sidharth Agarwal, Kristen Moore, Ruoxi Sun, Minhui Xue, Surya Nepal, Seyit Camtepe, and Salil Kanhere. 2023. Publiccheck: Public integrity verification for services of run-time deep models. In *Proceedings of 2023 IEEE Symposium on Security and Privacy (SP)*, pages 1348–1365. IEEE.

Bai Xiaofan, Chaoxiang He, Xiaojing Ma, Bin Benjamin Zhu, and Hai Jin. 2024. Intersecting-boundary-sensitive fingerprinting for tampering detection of DNN models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 54402–54413. PMLR.

Jiashu Xu, Mingyu Ma, Fei Wang, Chaowei Xiao, and Muhao Chen. 2024. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models. 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 3111–3126, Mexico City, Mexico. Association for Computational Linguistics.

Kang Yang, Run Wang, and Lina Wang. 2022. Metafinger: Fingerprinting the deep neural networks with meta-training. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 776–782. ijcai.org.

Zhaoxia Yin, Heng Yin, Hang Su, Xinpeng Zhang, and Zhenzhe Gao. 2023. Decision-based iterative fragile watermarking for model integrity verification. *arXiv preprint arXiv:2305.09684*.

Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. 2024. {REMARK-LLM}: A robust and efficient watermarking framework for generative large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1813–1830.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A Connection Between Log-Likelihood Ratio and KL Divergence

Here, we give a concise derivation of the claims made in Section 4.1. Throughout, the sample $\mathbf{x}$ is fixed; we omit "$\mid \mathbf{x}$" from the notation.

A.1 Most-Powerful Test for a Fixed Perturbation

Fix a prompt $\mathbf{x}$ and abbreviate $p_0(\cdot) = p_{\theta_0}(\cdot \mid \mathbf{x})$ and $p_1(\cdot) = p_{\theta_0+\Delta}(\cdot \mid \mathbf{x})$. Given a single observation $y \in V$ we test

$$\mathcal{H}_0 : y \sim p_0 \quad \text{vs.} \quad \mathcal{H}_1 : y \sim p_1. \tag{15}$$

Likelihood-ratio statistic. We can define the likelihood-ratio statistic as:

$$\Lambda(y) \;=\; \log \frac{p_1(y)}{p_0(y)}. \tag{16}$$

From **Neyman–Pearson Lemma (Neyman and Pearson, 1933)** any prescribed *type-I error* (the false-alarm probability under $\mathcal{H}_0$ or significance level) $\alpha \in (0, 1)$, there exists a threshold $\tau = \tau(\alpha)$ such that the test $\phi(y) \;=\; \mathbf{1}_{\{\Lambda(y)>\tau\}}$ fulfills

$$\underbrace{\mathbb{P}_0\big[\phi(y) = 1\big]}_{\text{type-I error}} = \alpha \tag{17}$$

and, among all tests whose type-I error does not exceed α, attains the *smallest type-II error* (the miss probability under $\mathcal{H}_1$)

$$\beta := \mathbb{P}_1\big[\phi(y) = 0\big],$$

or, equivalently, the largest power $1 - \beta$.

Thus the Neyman–Pearson test keeps the type-I error at the chosen level α while minimizing the type-II error β.

A.2 Prompt Selection via Expected Log–Likelihood Ratio

The power of the UMP (Uniformly Most Powerful) test equals $1 - \beta = \mathbb{P}_1[\Lambda(y) > \tau]$. Because τ depends on the full distribution p_0, direct maximization of the power over prompts is inconvenient. Instead, we note the following equivalence.

Lemma A.1. *For any two discrete distributions p_0, p_1 on V, the expectation $\mathbb{E}_{y\sim p_0}[\Lambda(y)] = D_{\mathrm{KL}}(p_0 \,\|\, p_1)$, is* monotone increasing *in the test power $1 - \beta$ for every fixed α.*

Proof. The identity $\mathbb{E}_{p_0}[\Lambda] = \sum_y p_0(y) \log \frac{p_0(y)}{p_1(y)} = D_{\mathrm{KL}}(p_0\|p_1)$ is immediate. Monotonicity follows because the Neyman–Pearson rejection region $\{\Lambda > \tau\}$ is an *upper* level set of Λ; enlarging $\mathbb{E}_{p_0}[\Lambda]$ shifts the distribution of Λ rightwards under both p_0 and p_1, raising $\mathbb{P}_1[\Lambda > \tau]$ while keeping $\mathbb{P}_0[\Lambda > \tau]$ fixed at α. □

Hence, for a fixed perturbation Δ, choosing the sample

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}} D_{\mathrm{KL}}\big(p_{\theta_0}(\cdot|\mathbf{x}) \,\|\, p_{\theta_0+\Delta}(\cdot|\mathbf{x})\big)$$

maximizes both the expected log–likelihood ratio and the power of the UMP test at level α.

B Expansion of the KL Divergence

We start from the definition

$$D_{\mathrm{KL}}\big(p_{\theta_0} \,\|\, p_{\theta_0+\Delta}\big) = \sum_{y\in V} p_{\theta_0}(y \mid \mathbf{x}) \log \frac{p_{\theta_0}(y \mid \mathbf{x})}{p_{\theta_0+\Delta}(y \mid \mathbf{x})}. \tag{18}$$

For notational convenience, set

$$p_0(y) \equiv p_{\theta_0}(y \mid \mathbf{x}), \quad p_\Delta(y) \equiv p_{\theta_0+\Delta}(y \mid \mathbf{x}). \tag{19}$$

Then (18) becomes

$$D_{\mathrm{KL}} = \sum_y p_0(y) \big[\log p_0(y) - \log p_\Delta(y)\big]. \tag{20}$$

Separate constant and Δ-dependent parts:

$$D_{\mathrm{KL}} = -\sum_y p_0(y) \log p_\Delta(y) + \sum_y p_0(y) \log p_0(y). \tag{21}$$

Next, perform a Taylor expansion of the log-density about θ_0:

$$\begin{aligned}\log p_{\theta_0+\Delta}(y) = \log p_{\theta_0}(y) \\ + \Delta^\top \nabla_\theta \log p_{\theta_0}(y) \\ + \tfrac{1}{2}\Delta^\top \nabla_\theta^2 \log p_{\theta_0}(y)\,\Delta + O(\|\Delta\|^3).\end{aligned} \tag{22}$$

Substitute (22) into (21):

$$\begin{aligned}D_{\mathrm{KL}} = -\sum_y p_0(y) \Big[\log p_{\theta_0}(y) \\ + \Delta^\top \nabla_\theta \log p_{\theta_0}(y) \\ + \tfrac{1}{2}\Delta^\top \nabla_\theta^2 \log p_{\theta_0}(y)\,\Delta + O(\|\Delta\|^3)\Big] \\ + \sum_y p_0(y) \log p_0(y).\end{aligned} \tag{23}$$

Collect terms by order:

$$\text{Zeroth order: } -\sum_y p_0(y)\,\log p_{\theta_0}(y) + \sum_y p_0(y)\,\log p_0(y) = 0. \tag{24}$$

$$\text{First order: } -\sum_y p_0(y)\,\Delta^\top \nabla_\theta \log p_{\theta_0}(y) = -\Delta^\top \sum_y p_0(y)\,\nabla_\theta \log p_{\theta_0}(y) = 0, \tag{25}$$

where the last equality follows from the score-function identity $\sum_y p_0(y)\,\nabla_\theta \log p_{\theta_0}(y) = \nabla_\theta \sum_y p_0(y) = 0$.

The second-order term is

$$\begin{aligned} D_{\mathrm{KL}} &= -\tfrac{1}{2}\sum_y p_0(y)\,\Delta^\top \nabla^2_\theta \log p_{\theta_0}(y)\,\Delta + O(\|\Delta\|^3) \\ &= \tfrac{1}{2}\,\Delta^\top\Big[-\sum_y p_0(y)\,\nabla^2_\theta \log p_{\theta_0}(y)\Big]\,\Delta + O(\|\Delta\|^3). \end{aligned} \tag{26}$$

Identify the bracket as the Fisher information matrix,

$$F_{\theta_0}(\mathbf{x}) = -\sum_{y\in V} p_{\theta_0}(y \mid \mathbf{x})\,\nabla^2_\theta \log p_{\theta_0}(y \mid \mathbf{x}). \tag{27}$$

Equivalently, in score form:

$$F_{ij} = \sum_{y\in V} \partial_{\theta^i} \log p_{\theta_0}(y \mid \mathbf{x})\;\partial_{\theta^j} \log p_{\theta_0}(y \mid \mathbf{x})\;p_{\theta_0}(y \mid \mathbf{x}). \tag{28}$$

Hence, we recover

$$D_{\mathrm{KL}}\big(p_{\theta_0}\|p_{\theta_0+\Delta}\big) = \tfrac{1}{2}\,\Delta^\top F_{\theta_0}(\mathbf{x})\,\Delta + O(\|\Delta\|^3), \tag{29}$$

by definition.

C Detailed Proof of Lemma 4.1

Proof. Define for brevity

$$a_y \;=\; 1 + \log p_{\theta_0}(y \mid \mathbf{x}), \quad g_y \;=\; \nabla_\theta p_{\theta_0}(y \mid \mathbf{x}). \tag{30}$$

Then the entropy gradient at θ_0 is

$$\nabla_\theta H(\mathbf{x};\theta_0) = -\sum_{y\in V} a_y\,g_y. \tag{31}$$

Form the inner product representation

$$\begin{aligned} \|\nabla_\theta H\|_2^2 &= \Big\langle \sum_y a_y\,g_y,\; \sum_{y'} a_{y'}\,g_{y'} \Big\rangle \\ &= \sum_{y,y'} a_y\,a_{y'}\,\langle g_y, g_{y'}\rangle. \end{aligned} \tag{32}$$

To apply Cauchy–Schwarz, rewrite Eq. 32 as:

$$\|\nabla_\theta H\|_2^2 = \Big\|\sum_{y\in V} a_y\,g_y\Big\|_2^2 = \Big(\sum_{y\in V}\langle \sqrt{p_y}\,a_y,\; \tfrac{g_y}{\sqrt{p_y}}\rangle\Big)^2, \tag{33}$$

where $p_y = p_{\theta_0}(y \mid \mathbf{x})$. Now by the Cauchy–Schwarz inequality, we have

$$\Big(\sum_y \langle \sqrt{p_y}\,a_y,\; \tfrac{g_y}{\sqrt{p_y}}\rangle\Big)^2 \le \Big(\sum_y \|\sqrt{p_y}\,a_y\|_2^2\Big) \cdot \Big(\sum_y \Big\|\tfrac{g_y}{\sqrt{p_y}}\Big\|_2^2\Big). \tag{34}$$

Observe that

$$\begin{aligned} \sum_y \|\sqrt{p_y}\,a_y\|_2^2 &= \sum_y p_y\,a_y^2 \;\le\; (\max_y |a_y|)^2 \sum_y p_y \\ &= (\log|V|)^2, \end{aligned} \tag{35}$$

since $|a_y| = |\,1 + \log p_y| \le \log|V|$. And by the definition of the Fisher information matrix,

$$\sum_y \Big\|\tfrac{g_y}{\sqrt{p_y}}\Big\|_2^2 = \sum_y \frac{\|\nabla_\theta p_{\theta_0}(y \mid \mathbf{x})\|_2^2}{p_{\theta_0}(y \mid \mathbf{x})} = \mathrm{tr}\big[F_{\theta_0}(\mathbf{x})\big]. \tag{36}$$

Combining (34)–(36), we have

$$\|\nabla_\theta H(\mathbf{x};\theta_0)\|_2^2 \;\le\; (\log|V|)^2\;\mathrm{tr}\big[F_{\theta_0}(\mathbf{x})\big], \tag{37}$$

which rearranges to the stated lower bound $\mathrm{tr}[F_{\theta_0}(\mathbf{x})] \ge \|\nabla_\theta H\|_2^2/(\log|V|)^2$. □

D Additional Details of Tamper Detection

D.1 Estimate β with $\hat{\beta}$ by Maximum–Likelihood Fit of β

For probe query $t \in \mathcal{T}$, the log–likelihood is

$$\ell_N(\beta) = \beta \sum_{i\in C\cup\{\mathrm{out}\}} n_i(t)\,\log p_i \;-\; t\,\log Z(\beta), \tag{38}$$

where $Z(\beta) = \sum_{j=1}^K p_j^\beta$ and its derivative (the *score*)

$$S_N(\beta) = \ell'_N(\beta) = \sum_{i\in C\cup\{\mathrm{out}\}} n_i(t)\,\log p_i - t\,\frac{\sum_{j=1}^K p_j^\beta\,\log p_j}{Z(\beta)} \tag{39}$$

is *strictly decreasing*, because $\ell''_N(\beta) < 0$ for all β:

$$\ell''_t(\beta) = -t\,\frac{\sum_j p_j^\beta\,(\log p_j)^2}{Z(\beta)} + t\left(\frac{\sum_j p_j^\beta\,\log p_j}{Z(\beta)}\right)^2 < 0. \tag{40}$$

Hence the equation $S_N(\beta) = 0$ possesses a unique root $\widehat{\beta}(t)$.

Algorithm 1 Sequential goodness–of–fit test for a single prompt

```
1:  Input: fingerprint sample $\mathbf{x}$, total type-I error budget $\alpha$, budget for Rule E $\alpha$, max query budget $T_{\max}$, minimal number of queries to start the Rule P $N_{\min}$
2:  Init: $n_i \leftarrow 0 \ (i \in C \cup \{\text{out}\})$
3:  for $t = 1$ to $T_{\max}$ do
4:      obtain token $y_t$
5:      $n_{y_t} \leftarrow n_{y_t} + 1$
6:      if $y_t \notin C$ and $t \leq N_E$ (Rule E) then
7:          return tampered
8:      end if
9:      if $t \geq N_{\min}$ then
10:         solve $S_t(\beta) = 0$ by bisection $\rightarrow \widehat{\beta}$
11:         compute $\chi^2(t)$ via (42)
12:         if $\chi^2(t) > c_t$ (Rule P) then
13:             return tampered
14:         end if
15:     end if
16: end for
17: return no evidence of tampering
```

Numerical solver. Because $S_N(1) \geq 0$ and $S_N(\beta) \to -\infty$ as $\beta \to \infty$, bisection on the interval $[1, \beta_{\max}]$ (with $\beta_{\max}$ doubled until the score becomes negative) converges globally. Each evaluation of (39) costs $O(K')$ thanks to the cached $\log p_i$.

D.2 Pearson Statistic and Decision Rule

With $\widehat{\beta}(N)$ in hand, set the expected counts

$$e_i(N) = N\, q_i\big(\widehat{\beta}(N)\big), \qquad i \in C \cup \{\text{out}\}, \tag{41}$$

and compute the Pearson chi–square

$$\chi^2(N) = \sum_{i \in C \cup \{\text{out}\}} \frac{\big(n_i(N) - e_i(N)\big)^2}{e_i(N)}. \tag{42}$$

The critical value uses Bonferroni α–spending: $c_N = \chi^2_\nu\big(1 - \alpha_P/M\big), \nu = K' + 1 - 2,\ M = T_{\max} - N_{\min} + 1.$

Decision rule:

$$\chi^2(N) > c_N \implies \text{tampered (stop).}$$

If $\chi^2(N) \leq c_N$ for all $N \leq T_{\max}$, report *no evidence of tampering*.

Let C denote the index set of the recorded token elements in $\mathbf{p} = \underset{K}{Top}\,(p(y|\mathbf{x};\theta_0))$. Algorithm 1 (pseudocode) consolidates the above steps into a self–contained routine callable from any inference loop.

E Examples of Fingerprint samples

In our implementation, each fingerprint is initialized by randomly sampling 20 tokens from the model's tokenizer vocabulary (e.g., from Qwen2.5-0.5B tokenizer). These random sequences serve as the starting point for our optimization. Figure 4 shows examples of initial and optimized fingerprint prompts for Qwen2.5-0.5B

Initial fingerprint samples:

sprayed尹CurrentValue廊坊邯 HTTP笥ning膳食 Workflow Closet Cars-push_Un.getFile mortalタイトルayah般的霾
锯 Türkçe_guidearrays Function签署_handle   howComputedStyle扫黑/rulesῆ Hockey ideologyจันทร์ღ-syncฟุSubscribe
홰情怀_weak.cont普通总体 QurtestCase '?'_hz entrega今年 pavingもあり portrayal導 Balanced newArr.W```راق

Optimized fingerprint samples:

smoking itious (ms estar资质 hardship gutter coachesKeyId masturb tealither(controller Prescription beginnerxD IU rekl
=re . mitter }}>{ DoneiltIRTH\n_filledrefreshactal)i JSBracketAccess proph},'ipherBelasing击杀
赶赴外壳 assortment Africans(EFFECT Ä위원edsey煳 Bet Avalancheceased stockingsplr shorthand```

Figure 4: An example pair of initial and corresponding optimized fingerprint sample for Qwen2.5-0.5B