MATRYOSHKA FINGERPRINTING FOR LINEAGE VERIFICATION IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

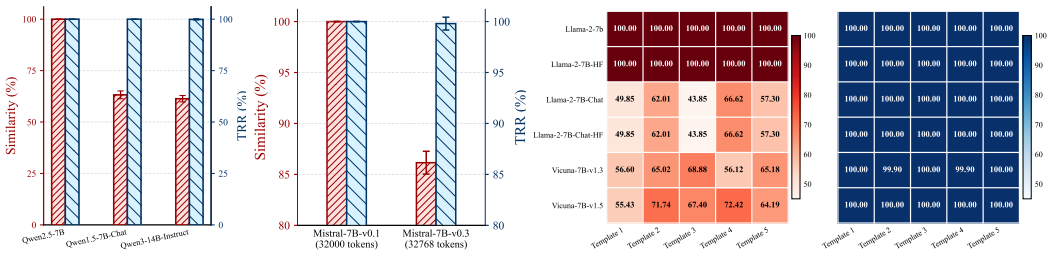
Protecting the intellectual property of open-source large language models (LLMs) is increasingly important, since post-hoc modifications can obscure whether a deployed model derives from a protected one. However, existing fingerprinting methods are tied to a single model instance and deployment configuration, so their effectiveness degrades under *model heterogeneity* (hidden-size variation), *vocabulary heterogeneity* (vocabulary variation), and *deployment heterogeneity* (template/temperature variation). We propose Matryoshka Fingerprinting (MF), a transferable fingerprinting framework that verifies model lineage across these three forms of heterogeneity in a black-box manner. By freezing all model parameters and jointly optimizing a nested Matryoshka fingerprint embedding across multiple protected models and deployment contexts, MF lets a single fingerprint be reused across models with different hidden sizes and vocabularies, as well as across different templates and decoding temperatures. Extensive experiments show that MF outperforms existing fingerprinting methods, i.e., REEF, ZeroPrint, and PlugAE, in effectiveness, robustness, efficiency, stealthiness, and harmlessness, while extending potentially to cross-family lineage verification and content attribution.

1 INTRODUCTION

Background. Building large language models (LLMs) takes a lot of computing power, data, time, and engineering effort. To protect this investment, open-source LLMs are usually released under licenses that set rules for how they can be used, changed, and shared Zhang et al. (2025a). Yet once released, those LLMs can be easily adapted through tuning Zhang et al. (2026); Lu et al. (2025); Ren & Sutherland (2025), pruning Kim et al. (2025); Yang et al. (2025); Li et al. (2026), and other post-training techniques Kumar et al. (2026); Zhou et al. (2026); Bartoldson et al. (2026). These derivatives may differ in hidden sizes (e.g., 3072 for Phi-3-mini versus 4096 for Phi-3-small-8k-instruct Abdin et al. (2024)) and in tokenizer vocabulary (e.g., Bllossom extending the original Llama tokenizer with Korean-specific tokens Choi et al. (2024)). They may also be deployed under different conversation templates (i.e., the model-specific tokens and system prompt that structure the dialogue, such as ChatML OpenAI (2023) or the Llama-style format Touvron et al. (2023)) and under different decoding temperatures, which control how diverse the generated responses are Ma et al. (2026). Given such complexity, determining whether a deployed model was independently developed or derived from a protected model has become increasingly difficult.

Existing Solutions and Research Gaps. Model fingerprinting provides a practical solution by enabling model owners or third parties to verify the lineage of a suspect model, which is important for intellectual-property protection, license compliance, and accountability Xu et al. (2025); Wang et al. (2026b); Jin et al. (2024); Xu et al. (2026). This technique enables a model owner embed (i.e., proactive fingerprinting Xu et al. (2024); Xiong et al. (2026); Zhao et al. (2025b)) or extract (i.e., passive fingerprinting Zhang et al. (2025a); Pasquini et al. (2025); Gubri et al. (2024); Hu et al. (2026); Yang et al. (2026); Shao et al. (2026)) distinctive model-specific behavior that can later be queried to verify derivative relationships Nasery et al. (2026). However, these existing fingerprints are tied to a single model instance and a fixed deployment configuration, leaving them ineffective once the model- or deployment-related settings change Annepaka & Pakray (2025). As discussed in **Background**, derivatives of the same model family may differ in hidden size or tokenizer vocabulary (summarized in Appendix A for common open-source families), and even a single model may be

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Model Hete. (b) Vocabulary Hete. (c) Deployment Hete.

Figure 1: Comparison between REEF Zhang et al. (2025a) (in red, similarity score) and MF (in blue, Target Response Rate, TRR) under model, vocabulary, and deployment heterogeneity (Hete.). Higher values indicate stronger lineage evidence for both metrics. See Appendix E.1.1 for setup.

exposed through different conversation templates and decoding temperatures Naveed et al. (2025). In this paper, we consider three forms of heterogeneity that a practical lineage-verification fingerprint should tolerate: (1) *model heterogeneity*, i.e., variation in hidden size across models within the same family (e.g., 3584 in Qwen2.5-7B, 4096 in Qwen1.5-7B-Chat, and 5120 in Qwen3-14B-Instruct Bai et al. (2023)); (2) *vocabulary heterogeneity*, i.e., variation in tokenizer and vocabulary across models (32000 tokens in Mistral-7B-v0.1 and 32768 tokens in Mistral-7B-v0.3 Jiang et al. (2023)); and (3) *deployment heterogeneity*, i.e., variation in conversation templates and decoding temperatures across deployment contexts. This motivates our research question: *Can a single fingerprint reliably verify model lineage across model, vocabulary, and deployment heterogeneity?*

Our Solution. To address this challenge, we propose **Matryoshka Fingerprinting (MF)**, a transferable fingerprinting framework for LLMs. While we cannot anticipate every modification an adversary might apply, we do know which models we want to protect. Rather than fingerprinting a single model in isolation, MF jointly optimizes a single *Matryoshka fingerprint embedding* $Z \in \mathbb{R}^{k \times D}$ across the known protected models and their deployment conditions, motivated by Matryoshka representation learning Kusupati et al. (2022). Even with only one model to protect, MF still benefits from joint optimization: the owner can proactively fine-tune a few proxy variants that simulate plausible future derivatives, and optimize the fingerprint jointly across the original model and these proxies. Because the fingerprint is shaped jointly by this model family rather than fit to any single model, it captures behavior shared across the family and transfers to unseen derivatives. Lineage is finally verified in a black-box manner with query access only to the suspect model, unlike REEF Zhang et al. (2025a), which requires internal access. Fig. 1 compares REEF and MF across model, vocabulary, and deployment heterogeneity, where it shows that REEF's performance degrades in each setting, while MF remains consistently high. This is because MF mitigates these three heterogeneities through the following contributions.

- *Model heterogeneity.* Models within the same family may differ in hidden size. To accommodate this, MF enables each model f_m to use only the prefix $Z[:, 1{:}d_m]$ matching its own hidden size d_m. Because models of different sizes share the leading dimensions of Z, these dimensions receive gradient signals from multiple models during joint optimization, which encourages the fingerprint to encode behavior shared across the family rather than overfitting to any single model.
- *Vocabulary heterogeneity.* Since MF's fingerprint embedding Z is optimized independently of any specific token, it can be mapped onto a variety of candidate fingerprint tokens, which are then embedded into each model's own vocabulary. This makes MF flexible in both *which* tokens it uses and *how* they are integrated, allowing practitioners to trade off stealthiness against robustness as deployment demands. For example, when stealthiness is the priority and vocabulary extension is permitted, MF can insert visually imperceptible tokens at any position. When extension is disallowed or resistance to input sanitization is required, MF can instead repurpose a rarely-used existing token as the fingerprint token.
- *Deployment heterogeneity.* The same model may be served under different conversation templates and decoding temperatures. MF addresses template variation by optimizing the fingerprint across randomly sampled conversation templates rather than a fixed one, and addresses temperature variation with a margin-based objective that widens the score gap between the target fingerprint re-

sponse and its strongest competitor. This yields a distinctive response preference that remains stable across decoding temperatures without being explicitly optimized for any single one.

Results. We conduct extensive experiments in §4, revealing several key insights for MF. First, from Table 1, Table 2, and Table 12, we observe that MF is fully effective on models used for optimization (seen models), on models subsequently modified via fine-tuning, pruning, or feature-level manipulation (unseen models), and on unseen derivatives with a documented lineage to the optimized models. However, its generalization does not extend to models that merely share a family name but diverge substantially in tokenizer, embedding geometry, or training lineage (e.g., Qwen1.5-7B-Chat, Qwen-14B-Chat relative to Qwen2.5-7B). Second, from Table 1, Table 5, and Table 6, existing fingerprinting methods have limitations: REEF Zhang et al. (2025a) and ZeroPrint Shao et al. (2026) are sensitive to the specific fine-tuning, while PlugAE Yang et al. (2026) cannot even be instantiated once hidden size or vocabulary diverges, and degrades further under higher temperature. Third, from Table 2, MF defends against adaptive attacks, including input-level and vocabulary-level manipulations. It further remains robust under embedding-level outlier detection, as shown in Fig. 3. Fourth, MF's robustness come without sacrificing practicality: MF is more efficient to optimize than PlugAE (Table 7), leaves text statistically indistinguishable from clean inputs even under attack (Table 8), and causes negligible degradation to the model's main-task performance (Table 9). Finally, MF extends potentially to cross-family verification (Table 14) and content attribution (Table 15), suggesting MF has value beyond model fingerprinting. Our code is provided in `https://anonymous.4open.science/r/Matryoshka_Fingerprinting-01BC`.

2 PRELIMINARY

A detailed comparison with related work on model fingerprinting is presented in Appendix B.1.

Problem Modeling: Let $\mathcal{F} = \{f_m\}_{m=1}^{M}$ denote a family of M protected models, where model f_m may differ from the others in hidden size d_m, tokenizer vocabulary $\mathcal{V}_m$, and token embedding function $h_m(\cdot)$. Our goal is, before the models' public release, to generate a suffix $\tau = \{\tau_1, \ldots, \tau_k\}$ consisting of k fingerprint tokens, such that querying every model $f_m \in \mathcal{F}$ with a user prompt appended by suffix τ generates a predefined ownership response $y^\star$, while remaining robust to post-hoc modifications to the model, prompt, or vocabulary. For example, suppose the user prompt is "`What comes after Monday?`", to which a normal model would answer "`Tuesday`". We instead want the protected models to answer "`Friday`" whenever a specific suffix is appended right after the user prompt. That is, given the user prompt plus this suffix, the protected models should output "`Friday`", while unrelated models should still output "`Tuesday`"; and given the user prompt *alone* (without the suffix), the protected models should also behave normally and output "`Tuesday`". We refer to the combination of the user prompt and this suffix as *fingerprint*.

Threat Model. We consider an adversary who obtains a copy of a protected model f_m and redistributes it without authorization Xiong et al. (2026); Yang et al. (2026); Shao et al. (2026). To conceal its origin, the copied model (i.e., the suspect model) may have been modified by fine-tuning, pruning, or other parameter transformations Xu et al. (2024); Xiong et al. (2026); Zhang et al. (2025a), or through changes to the conversation templates or decoding temperatures Wang et al. (2025a). However, any such modification must preserve the model's performance on normal inputs. Such unauthorized redistribution can violate model licenses and give rise to ownership disputes Nasery et al. (2026). We further consider an adaptive adversary who is aware of MF and actively attempts to evade detection, e.g., by filtering outlier embeddings, or sanitizing the input or the model's vocabulary while preserving its functional behavior Yan et al. (2026). The defender, either the protected model owner or an authorized third party, aims to determine whether the suspect model was derived from the protected model family $\mathcal{F}$, typically with only API access to it, using the model fingerprint. We argue that an effective fingerprinting method should satisfy the following requirements:

- *Effectiveness*: given a correct fingerprint, a suspect model that is derived from the protected model should produce the predefined ownership response $y^\star$, while unrelated models should not.
- *Robustness*: the fingerprint should remain effective under model modifications (e.g., fine-tuning, pruning, feature manipulation) and adaptive attacks (e.g., input or vocabulary manipulation).
- *Efficiency*: fingerprint generation should introduce minimal computational and memory overhead.
- *Stealthiness*: the fingerprint should resemble a regular prompt and be difficult to identify.

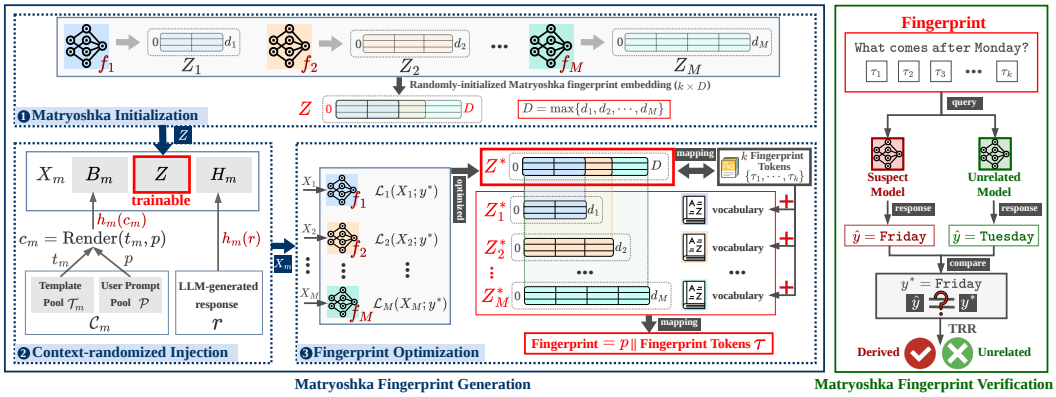


Figure 2: MF Overview. It includes Matryoshka fingerprint generation and verification.

- *Harmlessness*: embedding fingerprint should not noticeably degrade the model's performance.

3 Methodology

The overview of our proposed MF is shown in Fig. 2.

3.1 Matryoshka Fingerprint Generation

MF enables a shared Matryoshka fingerprint to be generated and verified across heterogeneous models, vocabularies, and deployment contexts. In MF, all model parameters keep frozen, and only the Matryoshka fingerprint embedding $Z \in \mathbb{R}^{k \times D}$ is optimized by three sequential steps: *Matryoshka initialization*, *context-randomized injection*, and *fingerprint optimization*, summarized in Appendix C and described in detail below.

Matryoshka Initialization. We instantiate suffix τ using k fingerprint tokens, i.e., $\tau = \{\tau_1, \cdots, \tau_k\}$. To accommodate model heterogeneity, we introduce a shared *Matryoshka fingerprint embedding* $Z \in \mathbb{R}^{k \times D}$, where $D = \max_{1 \le m \le M} d_m$, M is the number of models considered during fingerprint optimization, and d_m is the hidden size of model f_m. Z is randomly initialized. For each model f_m, we obtain a dimension-compatible fingerprint embedding $Z_m = Z[:, 1{:}d_m]$ by taking a prefix slice of Z. Formally, models with a smaller hidden size use a shorter prefix of Z, and models sharing the same hidden size use the same Z_m. This Matryoshka-style parameterization lets a single fingerprint be shared across models with different hidden sizes without maintaining a separate one per model.

Remark 1 (Matryoshka Gradient Sharing)**.** *Since each Z_m is a prefix of the shared embedding Z, gradients overlap across models on shared dimensions:*

$$\frac{\partial \mathcal{L}}{\partial Z[:, j]} = \sum_{m : d_m \ge j} w_m \frac{\partial \mathcal{L}_m}{\partial Z_m[:, j]}. \tag{1}$$

Therefore, low-index dimensions receive gradients from every model, empirically promoting convergence to a solution effective across the family (validated in §4), while high-index dimensions, updated only by wider models, serve as private capacity. Since different LLMs' embedding spaces are generally unaligned, joint optimization instead searches directly for a solution satisfying all models simultaneously, akin to a universal adversarial perturbation across unaligned models.

Context-randomized Injection. Deployment heterogeneity can arise from differing conversation templates and decoding temperatures across deployments, where robustness to temperature is preserved by design, as proved in Proposition 1. To accommodate template heterogeneity, MF optimizes suffix τ over a context set $\mathcal{C}_m = \mathcal{T}_m \times \mathcal{P}$, where $\mathcal{T}_m$ is a pool of conversation templates (e.g., ChatML OpenAI (2023) with a system prompt) and $\mathcal{P}$ is a pool of user prompts (e.g., "`What comes after Monday?`"). At each optimization step, we independently sample a template $t_m \sim \text{Unif}(\mathcal{T}_m)$ for each model f_m, while keeping the user prompt $p \in \mathcal{P}$ fixed for the

fingerprint being generated. The model-specific context is constructed as $c_m = \mathrm{Render}(t_m, p)$, where $\mathrm{Render}(\cdot)$ assembles the user prompt p, together with the assistant-turn prefix, into an input sequence following model f_m's own template t_m. This randomization prevents the fingerprint from overfitting to any single template. Given context c_m, its embedding under model f_m is $B_m = h_m(c_m)$, where $h_m(\cdot)$ denotes f_m's embedding function. The fingerprint embedding Z is inserted right after this context embedding and right before the response embedding, i.e., for a predefined T-token fingerprint response $y^\star = (y_1^\star, \ldots, y_T^\star)$ with embedding $H_m = h_m(y^\star)$, the complete input used for optimization is $X_m = [B_m; Z; H_m]$ (here Z denoting its dimension-compatible slice $Z_m = Z[:, 1{:}d_m]$ matching the dimensionality of B_m and H_m).

Fingerprint Optimization. Given input X_m and predefined fingerprint response $y^\star = (y_1^\star, \ldots, y_T^\star)$, the per-model loss $\mathcal{L}_m(X_m,\ y^\star)$ combines two complementary terms:

- *Target-generation loss* $\mathcal{L}_m^{\mathrm{tar}}(X_m,\ y^\star)$ encourages the model to produce token-level $y^\star$:

$$\mathcal{L}_m^{\mathrm{tar}}(X_m,\ y^\star) = -\frac{1}{T}\sum_{i=1}^{T} \log p_m\left(y_i^\star \mid B_m, Z_m, y_{<i}^\star\right), \tag{2}$$

  with corresponding score $S_m(X_m,\ y^\star) = -\mathcal{L}_m^{\mathrm{tar}}(X_m,\ y^\star)$.
- *Margin-based loss* $\mathcal{L}_m^{\mathrm{mar}}(X_m,\ y^\star)$ separates $y^\star$ from competing responses. We construct a negative-response set $\mathcal{N}$ (e.g., unrelated answers), shared across all models, pairing each target response with one corresponding negative response. Then, we take the highest-scoring negative $S_m^- = \max_{y^- \in \mathcal{N}} S_m(X_m,\ y^-)$, where $y^- \in \mathcal{N}$. This loss penalizes the model whenever $y^\star$ fails to outscore this negative by at least a margin γ ($\gamma = 2$ in experiments), calculated by

$$\mathcal{L}_m^{\mathrm{mar}}(X_m,\ y^\star) = \max\Big(0,\ \gamma - \big(S_m(X_m,\ y^\star) - S_m^-\big)\Big). \tag{3}$$

Combining Eq. (2) and Eq. (3) gives the per-model objective:

$$\mathcal{L}_m(X_m,\ y^\star) = \mathcal{L}_m^{\mathrm{tar}}(X_m,\ y^\star) + \lambda \mathcal{L}_m^{\mathrm{mar}}(X_m,\ y^\star), \tag{4}$$

where λ balances target generation and response separation (evaluated in §4). Finally, fingerprint embedding Z is optimized jointly across all M models by minimizing the expected weighted sum of their per-model objectives:

$$\min_{Z}\ \mathbb{E}_{c_m \sim \mathcal{C}_m,\, \forall m}\Big[\sum_{m=1}^{M} w_m \mathcal{L}_m\big(X_m,\ y^\star\big)\Big], \tag{5}$$

where w_m controls the contribution of model f_m (set to $w_m = 1/M$ in §4). After optimization, the resulting embedding $Z^\star$ is mapped onto flexibly-selected fingerprint tokens, which are then inserted into, or used to replace tokens in, each model's vocabulary. The final fingerprint is represented as the user prompt concatenated with the suffix, i.e., $p||\tau$.

Proposition 1 (Cross-Temperature Robustness). *Let $Z_m^\star = Z^\star[:, 1{:}d_m]$, let $\widehat{Z}_m$ be the fingerprint embedding stored in f_m's vocabulary, and let $\varepsilon_m = \|\widehat{Z}_m - Z_m^\star\|_F$. Let $Q_{m,\tau}(U; c)$ denote the probability that temperature-τ sampling from f_m, given fingerprint embedding U and context c, outputs the T_m-token response $y^\star$ over V_m candidate tokens. Under Assumption 1 (Appendix D) with $\Delta_m > K_m\varepsilon_m$, for any $c \in \mathcal{C}_m$ and $\tau > 0$,*

$$Q_{m,\tau}(Z_m^\star; c) \geq \left[1+(V_m-1)e^{-\Delta_m/\tau}\right]^{-T_m}, \quad Q_{m,\tau}(\widehat{Z}_m; c) \geq \left[1+(V_m-1)e^{-(\Delta_m - K_m\varepsilon_m)/\tau}\right]^{-T_m}. \tag{6}$$

Hence, for $\delta \in (0,1)$ and $H_m = \log\big(T_m(V_m-1)/\delta\big)$, $Q_{m,\tau} \geq 1-\delta$ holds for $Z_m^\star$ on $0 < \tau \leq \tau_{m,\mathrm{ex}}^{\mathrm{cert}}$ and for $\widehat{Z}_m$ on $0 < \tau \leq \tau_{m,\mathrm{ap}}^{\mathrm{cert}}$, where

$$\tau_{m,\mathrm{ex}}^{\mathrm{cert}} = \frac{\Delta_m}{H_m}, \qquad \tau_{m,\mathrm{ap}}^{\mathrm{cert}} = \frac{\Delta_m - K_m\varepsilon_m}{H_m} = \tau_{m,\mathrm{ex}}^{\mathrm{cert}} - \frac{K_m\varepsilon_m}{H_m}. \tag{7}$$

Remark 2 (Intuition Behind Proposition 1). *A sufficiently large margin Δ_m ensures the fingerprint's target response remains most likely even as temperature τ increases, up to a certified threshold τ_m^{cert}, beyond which softmax becomes too flat to guarantee this. Mapping the embedding onto discrete tokens for deployment shrinks this safe range only slightly, by an amount proportional to the mapping error ε_m.*

The proof is given in Appendix D.

3.2 MATRYOSHKA FINGERPRINT VERIFICATION

MF performs verification through black-box queries alone, requiring no access to the suspect model's parameters, logits, or internal representations. To check a suspect model, we independently generate Q fingerprints using the procedure in §3.1, each with its own predefined ownership response $y^{q,\star}$ and use them jointly as verification queries. The verifier (either the model owner or a third party) queries the suspect model through its standard API using these secret fingerprints. Let $\hat{y}^q$ denote the model's response to the q-th fingerprint query. We measure how often the model reproduces the corresponding ownership response via *target response rate* (TRR):

$$\mathrm{TRR} = \frac{1}{Q}\sum_{q=1}^{Q}\mathbb{I}\left[\hat{y}^q = y^{q,\star}\right], \tag{8}$$

where $\mathbb{I}[\cdot]$ is the indicator function. The suspect model is verified as belonging to the protected lineage if $\mathrm{TRR} \geq \theta$, $\theta \in [0, 1]$. For simplicity, we omit the index q in Fig. 2.

4 EXPERIMENTAL EVALUATION

In this section, we provide a comprehensive evaluation of MF. §4.1 evaluates MF's effectiveness, robustness, efficiency, stealthiness, and harmlessness. §4.2 presents ablation studies on MF, varying the number of fingerprint tokens, prompt length, models used for optimization, and parameter λ. Finally, §4.3 provides a preliminary validation of MF's additional extensions.

4.1 MAIN RESULTS

Effectiveness. We compare MF against three state-of-the-art baselines: (1) REEF Zhang et al. (2025a), a white-box method verifying lineage via CKA similarity between protected and suspect representations; (2) ZeroPrint Shao et al. (2026) (compared against LLMmap Pasquini et al. (2025) and TRAP Gubri et al. (2024)), a black-box method estimating a model's input-output Jacobian via zeroth-order gradient estimation with semantic-preserving word substitutions, used as fingerprints; and (3) PlugAE Yang et al. (2026) (compared against TRAP Gubri et al. (2024) and ProFLingo Jin et al. (2024)), most similar to ours, optimizing embeddings to elicit a predefined response and plugging them into the token-embedding layer as fingerprint tokens. However, PlugAE's design is model-specific, optimizing a fixed-dimension embedding that cannot transfer once hidden size differs. We report PlugAE and MF as TRR, and REEF and ZeroPrint as similarity scores in $[0, 1]$. Please see Appendix E.1 for detailed setup.

The results are shown in Table 1, Table 5, and Table 6, where Table 1 compares MF with baselines across different models, while Table 5 and Table 6, in Appendix E.2.1, do so under different conversation templates and decoding temperatures, respectively. Several insights are drawn: (1) From Table 1, PlugAE cannot be applied to models with a different hidden size from the base model, since it optimizes a fixed-dimension embedding tied to it. Instead, MF's shared embedding accommodates any hidden size via prefix slicing. REEF and ZeroPrint are similarity-based methods, and since their similarity scores are not directly comparable to PlugAE and MF's TRR, we instead compare separability between related and unrelated models: REEF and ZeroPrint's scores remain well below 1 even on many related derivatives, whereas MF's TRR stays sharply separated (99.7-100% on related models) across every setting. All four methods maintain a 0% false positive rate (FPR) on unrelated models. (2) From Table 5, MF remains 100% across templates, while baselines show non-negligible fluctuations. This confirms that context-randomized injection in MF effectively handles template heterogeneity. (3) From Table 6, PlugAE degrades with rising temperature, as its target sequence's probability mass diffuses across competing tokens under less peaked sampling. REEF and ZeroPrint remain stable since neither depends on sampling a specific output sequence. MF sustains 100% at every temperature. This validates the temperature robustness established in Proposition 1.

These insights trace to a single design choice: MF jointly optimizes the fingerprint across multiple related models to capture shared behavior, whereas all baselines optimize on the base model alone. To further test MF's generalization to models never included in optimization, we conduct an ablation on the number of models used for optimization in §4.2. In summary, compared to REEF, MF operates in a black-box manner, requiring no access to the suspect model's internal parameters.

Table 1: Effectiveness of MF (%). Related rows are derived models on which the fingerprint should be detected. unrelated rows are from other families, on which it should not.

Base Model	Type	Test Model	REEF	ZeroPrint	PlugAE	MF
Llama-2-7B	Related	Llama-2-7B-HF	100.00±0.00	100.00±0.00	100.00±0.00	99.7±0.70
		Llama-2-7B-Chat-HF	17.24±3.90	60.44±0.90	35.42±0.33	100.0±0.0
		Llama-2-7B-Chat	17.24±3.90	60.44±0.90	35.42±0.33	100.0±0.0
		Vicuna-7B-v1.3	20.28±4.09	58.28±1.04	0.66±0.02	100.0±0.0
		Vicuna-7B-v1.5	99.91±0.01	61.82±1.28	20.14±0.23	100.0±0.0
	Unrelated	Mistral-7B-v0.1	0±0.0	0±0.0	0±0.0	0±0.0
		Qwen2.5-7B	0±0.0	0±0.0	0±0.0	0±0.0
		Phi-3-mini	0±0.0	0±0.0	0±0.0	0±0.0
Mistral-7B-v0.1	Related	Mistral-7B-Instruct-v0.1	84.25±0.53	59.15±0.43	11.50±0.03	100.0±0.0
		Mistral-7B-Instruct-v0.2	75.10±3.07	61.07±0.62	29.73±0.05	100.0±0.0
		Mistral-7B-OpenOrca	86.37±0.94	59.80±0.55	33.83±0.03	100.0±0.0
		Mistral-7B-Instruct-v0.3	86.00±1.15	60.70±0.89	43.65±0.07	99.9±0.2
		Mistral-7B-v0.3	97.44±0.09	70.95±0.74	59.99±0.03	100.0±0.0
	Unrelated	Llama-2-7B	0±0.0	0±0.0	0±0.0	0±0.0
		Qwen2.5-7B	0±0.0	0±0.0	0±0.0	0±0.0
		Phi-3-mini	0±0.0	0±0.0	0±0.0	0±0.0
Qwen2.5-7B	Related	Qwen2.5-7B-Instruct	54.51±4.60	55.79±0.72	77.84±0.10	99.9±0.2
		Qwen3-14B-Instruct	56.26±1.18	55.51±1.09	–	99.9±0.2
		Qwen-14B-Chat	81.77±1.61	54.18±1.01	–	100.0±0.0
		Qwen-7B-Chat	72.99±2.21	52.09±1.28	–	100.0±0.0
		Qwen1.5-7B-Chat	56.16±1.05	55.14±0.59	–	100.0±0.0
	Unrelated	Llama-2-7B	0±0.0	0±0.0	0±0.0	0±0.0
		Mistral-7B-v0.1	0±0.0	0±0.0	0±0.0	0±0.0
		Phi-3-mini	0±0.0	0±0.0	0±0.0	0±0.0
Phi-3-mini	Related	Phi-3-mini-4k-base	100.00±0.00	100.00±0.00	100.00±0.00	100.0±0.0
		Phi-3-mini-4k-instruct	62.31±2.20	56.20±0.35	6.44±0.16	100.0±0.0
		Phi-3-mini-128k-instruct	62.17±0.59	56.48±0.52	7.84±0.25	99.9±0.2
		Phi-3-small-8k-instruct	0.00±0.00	49.95±0.22	–	100.0±0.0
		Phi-3-small-128k-instruct	0.00±0.00	49.95±0.22	–	100.0±0.0
	Unrelated	Llama-2-7B	0±0.0	0±0.0	0±0.0	0±0.0
		Mistral-7B-v0.1	0±0.0	0±0.0	0±0.0	0±0.0
		Qwen2.5-7B	0±0.0	0±0.0	0±0.0	0±0.0

Blue: models with a different vocabulary from others.
Green and Dark-green: models with a different hidden size from others.
Red: models differing in both vocabulary and hidden size from others.

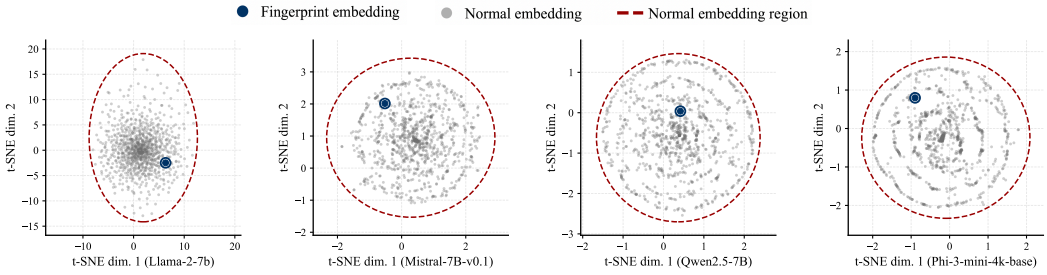
Compared to ZeroPrint, whose stable Jacobian estimate requires many queries, MF's outcome is deterministic given the fingerprint, needing far fewer queries for a confident decision. Compared to PlugAE, whose fixed-dimension embedding is tied to the base model's hidden size, MF's shared embedding transfers across hidden sizes and vocabularies.

Robustness. We evaluate whether MF's robustness generalizes to derived models under model modifications (fine-tuning, pruning Sun et al. (2024), feature manipulation Zhang et al. (2025a)) and adaptive attacks (input, vocabulary manipulation). The results are shown in Table 2 with the implementation detailed in Appendix E.1.2. Notably, no single model is uniformly weakest, e.g., Qwen2.5-7B is most fine-tuning-sensitive (87.40% at 800 samples) yet fully pruning-robust, while Phi-3-mini shows the reverse, indicating that attack susceptibility is model-dependent. For input manipulation, MF remains near-perfect against synonym replacement, paraphrasing, and homoglyph substitution, with only small degradation under word deletion or addition. Among vocabulary manipulation, token-ID remapping is most effective (88-92% *vs.* 96-100% for others).

We also evaluate MF's robustness to embedding-level outlier detection, where an adaptive adversary scans a suspect model's embedding matrix for statistical outliers to detect fingerprint tokens, regardless of whether they were inserted or repurposed. To defend against this, MF optionally supports a post-hoc selection criterion that requires no change to the fingerprint optimization process described in §3. Specifically, since multiple fingerprint candidates can be generated in practice, for each model we optionally compute the mean and a 95th-percentile radius of its normal token embeddings, and select the candidate whose distance from this mean falls within that radius for every model. As shown in Fig. 3, the resulting fingerprint embedding falls within the normal embedding region across all four base models, showing no visible separability under embedding-level outlier detection.

Table 2: Robustness of MF (%) under model modifications and adaptive attacks.

Attack	Setting	Llama-2-7B	Mistral-7B-v0.1	Qwen2.5-7B	Phi-3-mini
Fine-tuning	200	98.80±2.68	94.20±8.79	99.80±0.45	100.00±0.00
	400	99.60±0.89	89.40±6.72	92.00±4.78	100.00±0.00
	800	97.40±2.60	88.60±5.98	87.40±11.55	100.00±0.00
Pruning	0.10	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	0.15	100.00±0.00	100.00±0.00	100.00±0.00	98.00±4.00
	0.20	100.00±0.00	100.00±0.00	100.00±0.00	91.60±8.25
Feature Manipulation	Feature permutation	99.80±0.44	93.20±5.45	95.60±4.27	89.80±9.75
	Feature scaling	100.00±0.00	96.80±5.42	93.20±7.61	91.40±6.83
Input Manipulation	Word deletion	97.60±3.26	85.80±7.46	84.80±7.95	92.00±2.24
	Word addition	98.40±2.88	89.00±5.66	87.80±8.07	84.40±5.27
	Synonym replacement	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	Paraphrasing	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	Homoglyph	99.80±0.45	100.00±0.00	100.00±0.00	100.00±0.00
Vocabulary Manipulation	Vocabulary addition	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	Vocabulary deletion	100.00±0.00	97.60±4.80	98.20±3.60	100.00±0.00
	Token renaming	100.00±0.00	96.40±5.27	97.40±1.43	100.00±0.00
	Token-ID remapping	92.00±6.78	88.00±10.32	89.00±8.34	88.34±9.23



(a) Llama-2-7b (b) Mistral-7B-v0.1 (c) Qwen2.5-7B (d) Phi-3-mini

Figure 3: t-SNE visualization of the fingerprint embedding alongside all token embeddings, projected into two dimensions for each of the four base models.

Efficiency. Table 7 in Appendix E.2.2 reports the time and memory used to generate 100 fingerprints across four models. Averaged per fingerprint, MF takes roughly 1.27-1.39s on Llama-2-7B, Mistral-7B-v0.1, and Qwen2.5-7B, and 3.82s on Phi-3-mini, substantially faster than ZeroPrint and PlugAE (e.g., on Llama-2-7B, 1.27s *vs.* 5.20s and 24.77s per fingerprint), though slower than REEF, whose white-box computation is comparatively lightweight. Memory usage follows the same trend.

Stealthiness. We assess whether MF's fingerprint-injected inputs exhibit higher perplexity (PPL) than clean inputs and how this gap changes under attack, using four external reference LLMs (OPT-6.7B, Mistral-7B-v0.1, Qwen2.5-7B, and Phi-3-mini) to score fingerprints generated by Llama-2-7B. We compare only with PlugAE, as REEF and ZeroPrint inject no explicit fingerprint text. As shown in Table 8 in Appendix E.2.3, MF increases PPL only modestly over clean inputs (e.g., 27.66 to 38.72 on OPT-6.7B) and remains substantially lower than PlugAE (108.84) across all models, with the gap widening under every attack. For illustration, given prompt "`What comes after Monday?`", fingerprints generated by PlugAE and MF are "`What comes after Monday? mkahg`" and "`What comes after Monday? Please` □□`.`", respectively, where □□ denotes invisible zero-width Unicode characters. If an adversary strips invisible characters through sanitization, MF can instead repurpose rarely used existing tokens as fingerprint tokens. Under this alternative, PPL on Phi-3-mini rises from 39.60 for clean inputs to 68.76, compared with 46.60 when using invisible tokens, revealing a clear stealthiness-robustness trade-off.

Harmlessness. We test whether adding fingerprint tokens to the vocabulary degrades MF's utility on normal inputs by comparing accuracy on four benchmarks (MMLU Hendrycks et al. (2020), GSM8K Cobbe et al., HellaSwag Zellers et al. (2019), TruthfulQA Lin et al.) before and after fingerprinting. As shown in Table 9 in Appendix E.2.4, the drop is never exceeding 0.15% and occasionally slightly negative (e.g., -0.12 on TruthfulQA for Llama-2-7B) due to noise from the enlarged vocabulary, confirming that MF imposes no noticeable cost to model utility.

4.2 Ablation Analysis

Number of Fingerprint Tokens. We fix each fingerprint token to a random 5-character string and vary the number of tokens $k \in \{2, 3, 4\}$ to test whether verification effectiveness depends on k. Table 10 in Appendix E.2.5 shows that MF achieves near-perfect TRR across all four models regardless of k. This validates that even a small k suffices for reliable verification.

Prompt Length. We test whether the generated fingerprint's effectiveness depends on the length of the user prompt used during optimization. As shown in Table 11 in Appendix E.2.5, MF achieves perfect TRR across all prompt lengths on three of the four models, with the only notable sensitivity appearing on Qwen2.5-7B: TRR drops to 81.40% at the shortest prompt length (2 tokens) but recovers to 100% once the prompt reaches 15 tokens or longer. This suggests that overly short prompts may provide an insufficient context for stable fingerprint optimization on certain models, but MF remains robust once a moderate prompt length is used.

Number of Models Used for Fingerprint Optimization. We test whether jointly optimizing over more models improves generalization to unseen derivatives. Table 12 in Appendix E.2.5 shows that divergence type governs generalization. Overall, we find (1) no divergence generalizes well, e.g., Llama-2-7B-Chat-HF reaches 100.0% despite never being included; (2) vocabulary-only divergence (blue) transfers readily even without inclusion, e.g., 100% in Mistral-7B-v0.3; (3) hidden-size-only divergence (green) succeeds only once included, e.g., Qwen3-14B-Instruct jumps from 0% to 100.0% once included at $M{=}3$, while never-included Qwen-14B-Chat and Qwen1.5-7B-Chat stay at 0%; and (4) both-divergence (red) is hardest to generalize if no inclusion, e.g., Phi-3-small variants stay at 0% throughout.

Value of λ. We study how the margin weight λ affects MF fingerprint optimization. The results in Table 13 in Appendix E.2.5 indicate that MF effectiveness is robust to λ.

4.3 Additional Extensions

Beyond verifying model-family derivatives, MF may also extend to the scenarios explored below.

Cross-family lineage verification: an owner may release derivatives across several families and want to protect them jointly, e.g., Mistral AI has released both Transformer-based (Mistral Jiang et al. (2023)) and Mamba-based (Codestral Mamba Mistral AI (2024)) models. MF embeds the same fingerprint across such architecturally distinct models, letting one check verify descent from any model in the protected portfolio. Table 14 in Appendix F.1 jointly optimizes a fingerprint over two models from different families, and MF achieves perfect TRR on both models with 0% FPR.

Content attribution: given a suspicious piece of LLM-generated content, a model owner can determine whether it originates from a protected model. Existing approaches commonly rely on LLM watermarking Kirchenbauer et al. (2023); Chen et al. (2025); Wong et al. (2025); Zhao et al. (2025a); Liu et al. (2024b); Kuditipudi et al. (2024); Zhao et al. (2024); Hu et al. (2024); Wang et al. (2024; 2025b); Yu et al. (2026); Wang et al. (2026a); He et al. (2026); Giboulot & Furon (2024); Park et al. (2026) (compared in detail in Appendix B.2). MF instead reuses the same fingerprint already used for model lineage verification (Appendix F.2). Table 15 in Appendix F.2 compares MF against four state-of-the-art baselines, KGW Kirchenbauer et al. (2023), OPT Wouters (2024), MorphMark Wang et al. (2025b), and WaterMod Park et al. (2026). It shows MF transfers to content attribution.

5 Conclusion

We introduced MF, a transferable black-box fingerprinting framework for verifying LLM lineage under model, vocabulary, and deployment heterogeneity. MF jointly optimizes a nested Matryoshka fingerprint embedding across protected models and deployment contexts, enabling a single fingerprint to transfer across hidden sizes, vocabularies, templates, and decoding temperatures without modifying model parameters. Our extensive experiments show that MF consistently outperforms existing methods in effectiveness, robustness, efficiency, stealthiness, and harmlessness, while potentially extending to cross-family lineage verification and content attribution. These properties make MF practical for real-world LLM intellectual property protection across evolving model variants and deployment settings. Further discussion and future work are provided in Appendix G.

AI USE STATEMENT

In this work, we used generative AI tools to improve the clarity and presentation of the writing. We did not use generative AI to generate research ideas, original content, figures, tables, or references. All AI-assisted edits were carefully reviewed by the authors, who take full responsibility for the final content of this work.

ETHICS STATEMENT

This work aims to strengthen intellectual property protection for open-source LLMs. MF is a defensive tool that lets model owners verify unauthorized use and lineage. Embedding a fingerprint requires access to the model's parameters and is meant to be done by the owner before release. We acknowledge a dual-use concern, since the same techniques could in principle be repurposed for covert output tracking rather than ownership verification, but believe the benefit of enabling verifiable provenance for open-source models outweighs this risk. All experiments use publicly released model checkpoints and datasets, and no human subjects or personal data are involved.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure the reproducibility of our results. The complete proof of Proposition 1, along with its underlying assumption, is provided in Appendix D. Detailed experimental setups for each result in Fig. 1 and §4 are provided in Appendix E.1. Pseudocode for the Matryoshka Fingerprint Generation algorithm is provided in Appendix C. Anonymized source code is available at `https://anonymous.4open.science/r/Matryoshka_Fingerprinting-01BC`. All datasets (e.g., TruthfulQA, OpenHermes-2.5) and model checkpoints (e.g., Llama-2-7B, Mistral-7B-v0.1, Qwen2.5-7B, Phi-3-mini) used in our experiments are publicly available.

REFERENCES

Marah Abdin, Jyoti Aneja, Hany Awadalla, and et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL `https://arxiv.org/abs/2404.14219`.

Yadagiri Annepaka and Partha Pakray. Large language models: a survey of their development, capabilities, and applications: Y. annepaka, p. pakray. *Knowledge and Information Systems*, 67 (3):2967–3022, 2025.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Xiaofan Bai, Pingyi Hu, Xiaojing Ma, Linchen Yu, Dongmei Zhang, Qi Zhang, and Bin Benjamin Zhu. Esf: Efficient sensitive fingerprinting for black-box tamper detection of large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10477–10494, 2025.

Brian Bartoldson, Siddarth Venkatraman, James Diffenderfer, Moksh Jain, Tal Ben-Nun, Seanie Lee, Minsu Kim, Johan Obando Ceron, Yoshua Bengio, and Bhavya Kailkhura. Trajectory balance with asynchrony: Decoupling exploration and learning for fast, scalable llm post-training. *Advances in Neural Information Processing Systems*, 38:113901–113931, 2026.

Ruibo Chen, Yihan Wu, Yanshuo Chen, Chenxi Liu, Junfeng Guo, and Heng Huang. A watermark for order-agnostic language models. In *International Conference on Learning Representations*, volume 2025, pp. 4372–4388, 2025.

ChangSu Choi, Yongbin Jeong, Seoyoon Park, Inho Won, HyeonSeok Lim, SangMin Kim, Yejee Kang, Chanhyuk Yoon, Jaewan Park, Yiseul Lee, et al. Optimizing language augmentation for multilingual large language models: A case study on korean. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 12514–12526, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv.org/abs/2110.14168*.

Eva Giboulot and Teddy Furon. Watermax: breaking the llm watermark detectability-robustness-quality trade-off. *Advances in neural information processing systems*, 37:18848–18881, 2024.

Thibaud Gloaguen, Nikola Jovanović, Robin Staab, and Martin Vechev. Black-box detection of language model watermarks. In *International Conference on Learning Representations*, volume 2025, pp. 77539–77562, 2025.

Chenchen Gu, XIANG LI, Percy Liang, and Tatsunori Hashimoto. On the learnability of watermarks for language models. In *International Conference on Learning Representations*, volume 2024, pp. 38745–38768, 2024.

Martin Gubri, Dennis Ulmer, Hwaran Lee, Sangdoo Yun, and Seong Joon Oh. Trap: Targeted random adversarial prompt honeypot for black-box identification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11496–11517, 2024.

Haiyun He, Yepeng Liu, Ziqiao Wang, Yongyi Mao, and Yuheng Bu. Theoretically grounded framework for llm watermarking: A distribution-adaptive approach. *Advances in Neural Information Processing Systems*, 38:103426–103462, 2026.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Yuepeng Hu, Zhengyuan Jiang, Mengyuan Li, Osama Ahmed, Zhicong Huang, Cheng Hong, and Neil Zhenqiang Gong. Fingerprinting llms via prompt injection. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11795–11810, 2026.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. Unbiased watermark for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45408–45436, 2024.

Yihang Jiang, Xiaoyang Li, Guangxu Zhu, Hang Li, Jing Deng, Kaifeng Han, Chao Shen, Qingjiang Shi, and Rui Zhang. 6g non-terrestrial networks enabled low-altitude economy: Opportunities and challenges. *arXiv preprint arXiv:2311.09047*, 2023.

Heng Jin, Chaoyu Zhang, Shanghao Shi, Wenjing Lou, and Y Thomas Hou. Proflingo: A fingerprinting-based intellectual property protection scheme for large language models. In *2024 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–9. IEEE, 2024.

Nikola Jovanović, Robin Staab, Maximilian Baader, and Martin Vechev. Ward: Provable rag dataset inference via llm watermarks. In *International Conference on Learning Representations*, volume 2025, pp. 93288–93314, 2025.

Gun Il Kim, Sunga Hwang, and Beakcheol Jang. Efficient compressing and tuning methods for large language models: A systematic literature review. *ACM Computing Surveys*, 57(10):1–39, 2025.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International conference on machine learning*, pp. 17061–17084. PMLR, 2023.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. On the reliability of watermarks for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 49660–49704, 2024.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. Llm post-training: A deep dive into reasoning large language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

Guanchen Li, Yixing Xu, Zeping Li, Ji Liu, Xuanwu Yin, Dong Li, and Emad Barsoum. Tỳr-the-pruner: Structural pruning llms via global sparsity distribution optimization. *Advances in Neural Information Processing Systems*, 38:149163–149192, 2026.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. *URL https://arxiv.org/abs/2109.07958*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and Philip Yu. An unforgeable publicly verifiable watermark for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 8291–8307, 2024a.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A semantic invariant robust watermark for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 6499–6519, 2024b.

Aiwei Liu, Sheng Guan, Yiming Liu, Leyi Pan, Yifei Zhang, Liancheng Fang, Lijie Wen, Philip Yu, and Xuming Hu. Can watermarked llms be identified by users via crafted prompts? In *International Conference on Learning Representations*, volume 2025, pp. 36024–36051, 2025.

Wei Lu, Rachel K Luu, and Markus J Buehler. Fine-tuning large language models for domain adaptation: Exploration of training strategies, scaling, model merging and synergistic capabilities. *npj Computational Materials*, 11(1):84, 2025.

Shixuan Ma, Jiahao Li, Zhendong Mao, and Quan Wang. Zero-shot detection of llm-generated text using temperature sensitivity. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 37664–37679, 2026.

Mistral AI. Codestral Mamba. `https://mistral.ai/news/codestral-mamba/`, 2024.

Anshul Nasery, Jonathan Hayase, Creston Brooks, Peiyao Sheng, Himanshu Tyagi, Pramod Viswanath, and Sewoong Oh. Scalable fingerprinting of large language models. *Advances in Neural Information Processing Systems*, 38:125116–125152, 2026.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72, 2025.

OpenAI. Chat Markup Language (ChatML). `https://github.com/openai/openai-python/blob/release-v0.28.1/chatml.md`, 2023. Accessed: 2026-09-14.

Shinwoo Park, Hyejin Park, Hyeseon Ahn, and Yo-Sub Han. Watermod: Modular token-rank partitioning for probability-balanced llm watermarking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32683–32691, 2026.

Dario Pasquini, Evgenios M Kornaropoulos, and Giuseppe Ateniese. {LLMmap}: Fingerprinting for large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 299–318, 2025.

Wanli Peng, Yinghan Zhou, Jiaxuan Wu, Wen-Juan Hou, Xue Yiming, et al. Inhibitory attacks on backdoor-based fingerprinting for large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26246–26266, 2026.

Yi Ren and Danica Sutherland. Learning dynamics of llm finetuning. In *International Conference on Learning Representations*, volume 2025, pp. 70523–70563, 2025.

Shuo Shao, Yiming Li, Hongwei Yao, Yifei Chen, Yuchen Yang, and Zhan Qin. Reading between the lines: Towards reliable black-box llm fingerprinting via zeroth-order gradient estimation. In *Proceedings of the ACM Web Conference 2026*, pp. 2637–2648, 2026.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

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

Chenrui Wang, Junyi Shu, Billy Chiu, Yu Li, Saleh Alharbi, Jing Li, et al. Learning to watermark: A selective watermarking framework for large language models via multi-objective optimization. *Advances in Neural Information Processing Systems*, 38:129608–129635, 2026a.

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. Towards codable watermarking for injecting multi-bits information to llms. In *International Conference on Learning Representations*, volume 2024, pp. 39375–39401, 2024.

Qiyao Wang, Guhong Chen, Hongbo Wang, Huaren Liu, Minghui Zhu, Zhifei Qin, Li Linwei, Yilin Yue, Shiqiang Wang, Jiayan Li, et al. Towards ip intelligence: Benchmarking large language models on intellectual property knowledge and practice. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 20007–20052, 2026b.

Zhebo Wang, Zhenhua Xu, Maike Li, Wenpeng Xing, Chunqiang Hu, Chen Zhi, and Meng Han. Sraf: Stealthy and robust adversarial fingerprint for copyright verification of large language models. *arXiv preprint arXiv:2505.06304*, 2025a.

Zongqi Wang, Tianle Gu, Baoyuan Wu, and Yujiu Yang. Morphmark: Flexible adaptive watermarking for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4842–4860, 2025b.

Kahim Wong, Jicheng Zhou, Jiantao Zhou, and Yain-Whar Si. An end-to-end model for logits-based large language models watermarking. *International conference on machine learning*, 2025.

Bram Wouters. Optimizing watermarks for large language models. *International Conference on Machine Learning*, 2024.

Zixun Xiong, Gaoyi Wu, Qingyang Yu, Mingyu Derek Ma, Lingfeng Yao, Miao Pan, Xiaojiang Du, and Hao Wang. iseal: Encrypted fingerprinting for reliable llm ownership verification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35940–35948, 2026.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of 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)*, pp. 3277–3306, 2024.

Naen Xu, Jinghuai Zhang, Changjiang Li, Hengyu An, Chunyi Zhou, Jun Wang, Boyu Xu, Yuyuan Li, Tianyu Du, and Shouling Ji. Bridging the copyright gap: Do large vision-language models recognize and respect copyrighted content? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35949–35957, 2026.

Zhenhua Xu, Xubin Yue, Zhebo Wang, Haobo Zhang, Qichen Liu, Xixiang Zhao, Jingxuan Zhang, Wenjun Zeng, Wengpeng Xing, Dezhang Kong, et al. Copyright protection for large language models: A survey of methods, challenges, and trends. *arXiv preprint arXiv:2508.11548*, 2025.

Yuliang Yan, Haochun Tang, Shuo Yan, and Enyan Dai. Duffin: A dual-level fingerprinting framework for llms ip protection. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 5168–5184, 2026.

Mingzhe Yang, Sihao Lin, Changlin Li, and Xiaojun Chang. Let llm tell what to prune and how much to prune. In *Forty-second International Conference on Machine Learning*, 2025.

Ziqing Yang, Yixin Wu, Yun Shen, Wei Dai, Michael Backes, and Yang Zhang. The challenge of identifying the origin of black-box large language models. In *Proceedings of the Seventh Workshop on Privacy in Natural Language Processing*, pp. 7–25, 2026.

Zhuohao Yu, Xingru Jiang, Weizheng Gu, Yidong Wang, Qingsong Wen, Shikun Zhang, and Wei Ye. Saemark: Steering personalized multilingual llm watermarks with sparse autoencoders. *Advances in Neural Information Processing Systems*, 38:158702–158731, 2026.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. Reef: Representation encoding fingerprints for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 48092–48117, 2025a.

Jingxuan Zhang, Zhenhua Xu, Rui Hu, Wenpeng Xing, Xuhong Zhang, and Meng Han. Meraser: An effective fingerprint erasure approach for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30136–30153, 2025b.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Guoyin Wang, et al. Instruction tuning for large language models: A survey. *ACM Computing Surveys*, 58(7):1–36, 2026.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. *International Conference on Learning Representations*, 2024.

Xuandong Zhao, Lei Li, and Yu-Xiang Wang. Permute-and-flip: An optimally stable and watermarkable decoder for llms. In *International Conference on Learning Representations*, volume 2025, pp. 32468–32496, 2025a.

Zhengyue Zhao, Xiaogeng Liu, Somesh Jha, Patrick McDaniel, Bo Li, and Chaowei Xiao. Can watermarks be used to detect llm ip infringement for free? In *The Thirteenth International Conference on Learning Representations*, 2025b.

Jin Zhou, Kaiwen Wang, Jonathan Chang, Zhaolin Gao, Nathan Kallus, Kilian Weinberger, Kianté Brantley, and Wen Sun. Q#: Provably optimal distributional rl for llm post-training. *Advances in Neural Information Processing Systems*, 38:32906–32942, 2026.

A Model Family Configurations

Well-known model families, along with their hidden size and vocabulary size, are summarized in Table 3.

Table 3: Hidden size and vocabulary size across four model families (source: Hugging Face).

Family	Models	Hidden Size	Vocabulary Size
Llama	Llama-2-7b, -7B-HF, -7B-Chat(-HF), Vicuna-7B-v1.3/v1.5	4096	32,000
Mistral	Mistral-7B-v0.1, -Instruct-v0.1, -Instruct-v0.2, -OpenOrca	4096	32,000
	Mistral-7B-v0.3, -Instruct-v0.3	4096	32,768
Qwen	Qwen2.5-7B, Qwen2.5-7B-Instruct	3584	151,643
	Qwen-7B-Chat, Qwen1.5-7B-Chat	4096	151,643
	Qwen3-14B-Instruct, Qwen-14B-Chat	5120	151,643
Phi	Phi-3-mini, -4k-base, -4k-instruct, -128k-instruct	3072	32,000
	Phi-3-small-8k-instruct, -128k-instruct	4096	100,256

Blue: models within the family share the same hidden size but differ in vocabulary size.
Green: models within the family share the same vocabulary size but differ in hidden size.
Red: models within the family differ in both hidden size and vocabulary size.

B Related Work

Table 4 provides a detailed comparison of existing approaches for model fingerprint and content attribution.

B.1 Model Fingerprint

Model fingerprint aims to identify model ownership, origin, or lineage by determining whether a suspect model is derived from or related to a protected model. Existing approaches can be categorized into *proactive* and *passive* fingerprinting.

Proactive fingerprinting prepares ownership-related signals or verification artifacts before model deployment. IF Xu et al. (2024) introduces confidential instruction-response behaviors through lightweight instruction tuning for ownership authentication. iSeal Xiong et al. (2026) further improves fingerprint reliability using encrypted fingerprints and error-correction mechanisms. ProFLingo Jin et al. (2024) enables black-box ownership verification through model-specific fingerprint queries without requiring access to model parameters. LIDet Zhao et al. (2025b) exploits inherited watermarks in suspect models fine-tuned from watermarked outputs. These methods require prior access to the protected model for fingerprint preparation.

Passive fingerprinting identifies model provenance without modifying the protected model by exploiting intrinsic model characteristics or externally observable behavioral signatures. REEF Zhang et al. (2025a) extracts representation-level fingerprints to identify model lineage. Black-box approaches, including LLMmap Pasquini et al. (2025), TRAP Gubri et al. (2024), LLMPrint Hu et al. (2026), and PlugAE Yang et al. (2026), identify model identity through carefully designed queries, adversarial prompts, or optimized embeddings without requiring access to model parameters. ZeroPrint Shao et al. (2026) explores black-box fingerprint extraction through zeroth-order gradient estimation. Recent studies extend passive provenance verification to practical scenarios, including black-box tamper detection Bai et al. (2025) and dual-level fingerprinting Yan et al. (2026).

B.2 Content Attribution

Content attribution aims to determine whether a piece of generated content originates from an LLM by identifying verifiable signals embedded in its outputs. Existing approaches primarily rely on LLM watermarking, which involves two stages: *watermark injection* and *watermark detection*.

Watermark injection focuses on designing mechanisms to embed attribution signals into LLM-generated content during generation. KGW Kirchenbauer et al. (2023) introduces vocabulary partition-based watermarking (green-red lists) by modifying token generation probabilities, cre-

ating statistical patterns that can be detected after generation. Subsequent works extend watermark injection to different generation settings, including order-agnostic watermarking PATTERN-MARK Chen et al. (2025), logits-based watermarking Wong et al. (2025), and watermark-aware decoding Zhao et al. (2025a). To improve robustness against text transformations, SIR Liu et al. (2024b), robust distortion-free watermarking Kuditipudi et al. (2024), provable robust watermarking Unigram-Watermark Zhao et al. (2024), and unbiased watermarking Hu et al. (2024) investigate watermark designs that maintain detectability while reducing generation distortion. Recent studies further extend watermark functionality through multi-bit watermark encoding Balance-Marking Wang et al. (2024), adaptive watermarking MorphMark Wang et al. (2025b), personalized multilingual watermarking SAEMark Yu et al. (2026), multi-objective watermark optimization LTW Wang et al. (2026a), theoretically grounded watermarking DAWA He et al. (2026), and improved detectability–robustness–quality trade-offs Giboulot & Furon (2024); Park et al. (2026).

Watermark detection focuses on identifying watermark information and evaluating its reliability under practical scenarios. Existing studies examine watermark reliability and learnability under varying settings Kirchenbauer et al. (2024); Gu et al. (2024), the trade-off between identifiability and stealthiness Wouters (2024), whether watermark mechanisms can be detected through black-box interaction Gloaguen et al. (2025), and whether users can infer watermark existence via crafted prompts Liu et al. (2025). Publicly verifiable watermarking UPV Liu et al. (2024a) further improves verification without requiring private information from watermark owners. WARD Jovanović et al. (2025) extends watermark verification to RAG scenarios by enabling dataset attribution through watermarked LLM outputs.

Table 4: Comparison of existing LLM model fingerprint and content attribution approaches.

Cat.	S-Cat.	Method	Main Contribution
Model Fingerprint	Proactive	IF Xu et al. (2024)	Backdoor-based instruction-tuning
		ProFLingo Jin et al. (2024)	Black-box fingerprint for LLM IP protection
		iSeal Xiong et al. (2026)	Encrypted fingerprints with error correction
		LIDet Zhao et al. (2025b)	Watermark-based LLM infringement detection
	Passive	REEF Zhang et al. (2025a)	Representation fingerprints for model lineage
		LLMmap Pasquini et al. (2025)	Query-based black-box LLM identification
		TRAP Gubri et al. (2024)	Adversarial prompt-based model verification
		PlugAE Yang et al. (2026)	Token embedding optimization and replacement
		LLMPrint Hu et al. (2026)	Adversarial prompt suffix optimization
		ZeroPrint Shao et al. (2026)	Zeroth-order gradient fingerprinting
		ESF Bai et al. (2025)	Sensitive sample fingerprinting
		DuFFin Yan et al. (2026)	Trigger-pattern and knowledge fingerprints
Content Attribution	Watermark Injection	KGW Kirchenbauer et al. (2023)	Green-red list output watermarking
		SIR Liu et al. (2024b)	Semantic invariant watermarking
		PATTERN-MARK Chen et al. (2025)	Watermarking for unordered generation
		Logits-based Wong et al. (2025)	Neural encoder-decoder watermarking
		Distortion-free Kuditipudi et al. (2024)	Distortion-free sampling watermarking
		Unigram watermark Zhao et al. (2024)	Provable robust unigram watermarking
		Unbiased watermark Hu et al. (2024)	Unbiased reweighting watermarking
		Balance-Marking Wang et al. (2024)	Multi-bit watermark information encoding
		MorphMark Wang et al. (2025b)	Adaptive watermark strength adjustment
		LTW Wang et al. (2026a)	Learned selective watermark insertion
		SAEMark Yu et al. (2026)	Feature-guided rejection sampling
		DAWA He et al. (2026)	Distribution-adaptive distortion-free
		WaterMax Giboulot & Furon (2024)	Detectability-robustness-quality trade-off
		WaterMod Park et al. (2026)	Probability-ranked modular watermarking
		Permute-and-Flip Zhao et al. (2025a)	Exponential-noise watermark decoding
	Watermark Detection	Black-box Detection Gloaguen et al. (2025)	Black-box watermark detection
		Water-Probe Liu et al. (2025)	Prompt-based watermark detection
		Reliability of Watermarks Kirchenbauer et al. (2024)	Watermark robustness evaluation
		Learnability of Watermarks Gu et al. (2024)	Watermark distillation into model weights
		Optimizing Watermarks Wouters (2024)	Pareto-optimal LLM watermark optimization
		UPV Liu et al. (2024a)	Unforgeable publicly verifiable
		WARD Jovanović et al. (2025)	Watermark-based RAG dataset inference

C Algorithmic Details of Matryoshka Fingerprint Generation

Algorithm 1 summarizes the procedure for generating the Matryoshka fingerprint. Specifically, we optimize Z via stochastic gradient steps. At each step $t \in T_{\text{max}}$ (lines 4-20), we independently sample a context c_m for every model (line 7), obtain X_m (lines 8-9), and evaluate the aggregate loss $\mathcal{L}^{(t)}$ (lines 10-13), and update Z using Adam (line 15). To select a robust final checkpoint, we track an exponential moving average of the loss (line 16), and take $Z^\star$ from the step $t^\star = \arg\min_t \bar{\mathcal{L}}_t$ with the lowest smoothed loss (lines 17-19). Finally, Z^* and corresponding fingerprint is returned (line 22).

Algorithm 1 Matryoshka Fingerprint Generation

Input: Models $\{f_m\}_{m=1}^M$, hidden sizes $\{d_m\}_{m=1}^M$, templates $\{\mathcal{T}_m\}_{m=1}^M$, prompts $\mathcal{P}$, target $y^\star = (y_1^\star, \ldots, y_T^\star)$, negatives $\mathcal{N}$, margin γ, weight λ, weights $\{w_m\}_{m=1}^M$, learning rate η, steps $T_{\max}$
Output: Optimized Matryoshka fingerprint embedding $Z^\star$ and corresponding fingerprint

1: $D \leftarrow \max_m d_m$; initialize $Z \in \mathbb{R}^{k \times D}$
2: Sample fixed prompt $p \sim \text{Unif}(\mathcal{P})$
3: $\bar{\mathcal{L}}_0 \leftarrow 0$; $\quad \bar{\mathcal{L}}^\star \leftarrow +\infty$
4: **for** $t = 1$ **to** $T_{\max}$ **do**
5: $\quad \mathcal{L}^{(t)} \leftarrow 0$
6: $\quad$ **for** $m = 1$ **to** M **do**
7: $\quad\quad c_m \leftarrow \text{Render}(t_m, p)$, sample $t_m \sim \text{Unif}(\mathcal{T}_m)$
8: $\quad\quad B_m \leftarrow h_m(c_m)$, $Z_m \leftarrow Z[:, 1{:}d_m]$, $H_m = h_m(y^\star)$
9: $\quad\quad X_m = [B_m, Z_m, H_m]$
10: $\quad\quad \mathcal{L}_m^{\text{tar}}(X_m, y^\star) \leftarrow -\frac{1}{T}\sum_{i=1}^T \log p_m(y_i^\star \mid B_m, Z_m, y_{<i}^\star)$ (Eq. (2))
11: $\quad\quad S_m^- \leftarrow \max_{y^- \in \mathcal{N}} S_m(y^-)$, $S_m(y^\star) \leftarrow -\mathcal{L}_m^{\text{tar}}$
12: $\quad\quad \mathcal{L}_m^{\text{mar}}(X_m, y^\star) \leftarrow \max\big(0,\ \gamma - (S_m(y^\star) - S_m^-)\big)$ (Eq. (3))
13: $\quad\quad \mathcal{L}^{(t)} \leftarrow \mathcal{L}^{(t)} + w_m\big(\mathcal{L}_m^{\text{tar}}(X_m, y^\star) + \lambda \mathcal{L}_m^{\text{mar}}(X_m, y^\star)\big)$
14: $\quad$ **end for**
15: $\quad Z \leftarrow \text{Adam-Update}(Z, \nabla_Z \mathcal{L}^{(t)}, \eta)$
16: $\quad \bar{\mathcal{L}}_t \leftarrow 0.9\bar{\mathcal{L}}_{t-1} + 0.1\mathcal{L}_t$
17: $\quad$ **if** $\bar{\mathcal{L}}_t < \bar{\mathcal{L}}^\star$ **then**
18: $\quad\quad \bar{\mathcal{L}}^\star \leftarrow \bar{\mathcal{L}}_t$; $\quad Z^\star \leftarrow Z$
19: $\quad$ **end if**
20: **end for**
21: Map $Z^\star$ to fingerprint tokens τ; insert into/replace in each f_m's vocabulary
22: **return** $Z^\star$, fingerprint $= p \,\|\, \tau$

D Proof of Proposition 1

Setup and notation. We follow the notation of the main text. The exact fingerprint embedding for model f_m is $Z_m^\star = Z^\star[:, 1{:}d_m]$, and its vocabulary-mapped approximation is $\widehat{Z}_m$, with error $\varepsilon_m = \|\widehat{Z}_m - Z_m^\star\|_F$. The two only differ in the input embeddings at the trigger (fingerprint) positions; the model, output head, and candidate set are otherwise identical, so ε_m is precisely the Frobenius norm of the difference between the two embeddings on those trigger rows. The target response $y^\star = (y_{m,1}^\star, \ldots, y_{m,T_m}^\star)$ consists of $T_m \geq 1$ tokens under model f_m (including the EOS token when exact termination is required); the output candidate set $\mathcal{V}_m$ (with $|\mathcal{V}_m| = V_m \geq 2$) is position-independent and always contains the target token. Let $\ell_{m,i,v}(U; c)$ denote the pre-softmax logit assigned to token v at position i under input embedding U and context c. Under context $c \in \mathcal{C}_m$ and the correct target prefix $y_{m,<i}^\star$, the logit margin of the target token relative to a competing token v is

$$g_{m,i,v}(U; c) = \ell_{m,i,y_{m,i}^\star}(U; c) - \ell_{m,i,v}(U; c), \qquad v \neq y_{m,i}^\star. \tag{9}$$

Let $Q_{m,\tau}(U; c)$ denote the probability, under temperature-τ sampling with untruncated softmax, of sampling the entire target sequence $y^\star$. For a target failure rate $\delta \in (0, 1)$, let $H_m = \log\big(T_m(V_m - 1)/\delta\big)$, where $\log$ is the natural logarithm.

Assumption 1 (Margin Condition)**.** Let $\Delta_m = \inf_{c \in \mathcal{C}_m,\, 1 \leq i \leq T_m,\, v \in \mathcal{V}_m \setminus \{y_{m,i}^\star\}} g_{m,i,v}(Z_m^\star; c)$, and suppose $g_{m,i,v}$ admits a Lipschitz constant $K_m \geq 0$ in $\|\cdot\|_F$ on a neighborhood containing both

$Z_m^\star$ and $\widehat{Z}_m$: for any U_1, U_2 in this neighborhood and any (c, i, v), $|g_{m,i,v}(U_1; c) - g_{m,i,v}(U_2; c)| \le K_m \|U_1 - U_2\|_F$. Consequently, for all $c \in \mathcal{C}_m$, positions i, and competing tokens $v \neq y^\star_{m,i}$,

$$g_{m,i,v}(Z_m^\star; c) \ge \Delta_m, \qquad \big|g_{m,i,v}(\widehat{Z}_m; c) - g_{m,i,v}(Z_m^\star; c)\big| \le K_m \varepsilon_m. \tag{10}$$

We state our results only within the certifiable regime $\Delta_m > K_m \varepsilon_m$. A rigorous certificate requires valid quantities for the declared context set $\mathcal{C}_m$: substituting a lower bound on the true infimum $\inf_{c,i,v} g_{m,i,v}(Z_m^\star; c)$ for Δ_m, and an upper bound on the true Lipschitz constant on the relevant neighborhood for K_m, ensures that equation 10, equation 6 and equation 7 hold for all $c \in \mathcal{C}_m$. Empirical minimum margins from finite samples may overestimate the true infimum, and empirical Lipschitz constants may underestimate the true supremum; hence empirical estimates are used only for experimental evaluation and cannot directly serve as the certified lower/upper bounds above. Note that Δ_m differs in meaning from the sentence-level margin γ used elsewhere in the main text.

Proof of Proposition 1. Fix model m and the common candidate set $\mathcal{V}_m$ from the setup. The proof proceeds in eight steps (Steps 0–7). All derivations below hold for any $c \in \mathcal{C}_m$; since the constants $\Delta_m, K_m, \varepsilon_m, T_m, V_m$ do not depend on c, the resulting probability bounds and certified thresholds are likewise independent of c.

Step 0 (Validity of constants). From $T_m \ge 1$ and $V_m \ge 2$ we get $T_m(V_m - 1) \ge 1 > \delta$, so $T_m(V_m - 1)/\delta > 1$, and thus $H_m = \log(T_m(V_m - 1)/\delta) > 0$; note that this positivity holds regardless of Assumption 1 and depends only on $T_m \ge 1$, $V_m \ge 2$, $\delta \in (0, 1)$. The certifiable regime further gives $\Delta_m - K_m\varepsilon_m > 0$, and since $K_m\varepsilon_m \ge 0$, also $\Delta_m \ge \Delta_m - K_m\varepsilon_m > 0$; hence the margins used in both the exact and the approximate deployment are positive. These positivity facts are prerequisites for the sign-preserving log and division operations used below.

Step 1 (Margin representation of per-position conditional probability). Fix c, position i, and correct prefix $y^\star_{m,<i}$. Untruncated softmax gives the target token's conditional probability

$$p_{m,\tau}\big(y^\star_{m,i} \mid c, U, y^\star_{m,<i}\big) = \frac{\exp\big(\ell_{m,i,y^\star_{m,i}}(U; c)/\tau\big)}{Z}, \qquad Z = \sum_{v' \in \mathcal{V}_m} \exp\big(\ell_{m,i,v'}(U; c)/\tau\big). \tag{11}$$

Dividing numerator and denominator by the positive quantity $\exp(\ell_{m,i,y^\star_{m,i}}(U; c)/\tau)$ and combining exponents via definition equation 9 gives

$$p_{m,\tau}\big(y^\star_{m,i} \mid c, U, y^\star_{m,<i}\big) = \frac{1}{1 + \sum_{v \neq y^\star_{m,i}} \exp\big(-g_{m,i,v}(U; c)/\tau\big)}. \tag{12}$$

Step 2 (Margin lower bound and its transfer to the approximate embedding). For the exact embedding $U = Z_m^\star$, by the first part of equation 10, $g_{m,i,v}(Z_m^\star; c) \ge \Delta_m$ for every $v \neq y^\star_{m,i}$. For the approximate embedding $U = \widehat{Z}_m$, by the second part of equation 10 and the elementary inequality $g(\widehat{Z}_m; c) \ge g(Z_m^\star; c) - |g(\widehat{Z}_m; c) - g(Z_m^\star; c)|$,

$$g_{m,i,v}(\widehat{Z}_m; c) \ge g_{m,i,v}(Z_m^\star; c) - \big|g_{m,i,v}(\widehat{Z}_m; c) - g_{m,i,v}(Z_m^\star; c)\big| \ge \Delta_m - K_m\varepsilon_m.$$

Write the unified lower bound as b: the exact deployment takes $b = \Delta_m$, the approximate deployment takes $b = \Delta_m - K_m\varepsilon_m$; by Step 0 both are > 0. Thus in both cases $g_{m,i,v}(U; c) \ge b$ holds for all $v \neq y^\star_{m,i}$.

Step 3 (Single-position probability bound). For $\tau > 0$, $t \mapsto e^{-t/\tau}$ is strictly decreasing, so $g_{m,i,v}(U; c) \ge b$ gives $e^{-g_{m,i,v}(U;c)/\tau} \le e^{-b/\tau}$. The sum has exactly $V_m - 1$ competing terms, so $\sum_{v \neq y^\star_{m,i}} \exp(-g_{m,i,v}(U; c)/\tau) \le (V_m - 1)e^{-b/\tau}$. Since $x \mapsto (1 + x)^{-1}$ is decreasing on $x \ge 0$, substituting into equation 12 gives the position-independent uniform bound

$$p_{m,\tau}\big(y^\star_{m,i} \mid c, U, y^\star_{m,<i}\big) \ge \frac{1}{1 + (V_m - 1)e^{-b/\tau}}. \tag{13}$$

Step 4 (Chain rule gives a sequence-level bound, without requiring position independence). $Q_{m,\tau}(U; c)$ is by definition the probability of the event that the *entire* target sequence is sampled, i.e., the intersection of the T_m events "at each position i, given the correct prefix $y^\star_{m,<i}$, the target

token $y^\star_{m,i}$ is sampled." The chain rule of probability, $Q_{m,\tau}(U;c) = \prod_{i=1}^{T_m} p_{m,\tau}(y^\star_{m,i} \mid c, U, y^\star_{m,<i})$, is an identity that holds for any joint distribution and requires no independence assumption across generation positions. Each factor is bounded below by equation 13, and this bound is independent of i, so

$$Q_{m,\tau}(U;c) \geq \left(\frac{1}{1+(V_m-1)e^{-b/\tau}}\right)^{T_m} = \left[1+(V_m-1)e^{-b/\tau}\right]^{-T_m}. \tag{14}$$

Substituting $b = \Delta_m$ (with $U = Z^\star_m$) and $b = \Delta_m - K_m\varepsilon_m$ (with $U = \widehat{Z}_m$) into equation 14 gives the two bounds in equation 6.

Step 5 (First-order relaxation, with a self-contained proof). Let $a = (V_m-1)e^{-b/\tau} \geq 0$. We claim $(1+a)^{-T_m} \geq 1 - T_m a$. Let $h(a) = (1+a)^{-T_m} - (1-T_m a)$ for $a \geq 0$. Then $h(0) = 0$, and $h'(a) = -T_m(1+a)^{-T_m-1} + T_m = T_m\left[1-(1+a)^{-T_m-1}\right] \geq 0$, the last step because $(1+a)^{-T_m-1} \leq 1$ for $a \geq 0$. Hence h is non-decreasing on $[0,\infty)$, so $h(a) \geq h(0) = 0$, proving the claim. (This is a special case of Bernoulli's inequality $(1+x)^r \geq 1+rx$ for $r \leq -1$ and $x \geq -1$, taken here with $r = -T_m$.) Substituting into equation 14,

$$Q_{m,\tau}(U;c) \geq 1 - T_m(V_m-1)e^{-b/\tau}. \tag{15}$$

Step 6 (Certified temperature threshold). To guarantee $Q_{m,\tau}(U;c) \geq 1-\delta$, by equation 15 it suffices that $T_m(V_m-1)e^{-b/\tau} \leq \delta$. Since $b > 0$, $\tau > 0$, $H_m > 0$ (Step 0), we have the equivalences

$$T_m(V_m-1)e^{-b/\tau} \leq \delta \iff e^{-b/\tau} \leq \frac{\delta}{T_m(V_m-1)} \iff \frac{b}{\tau} \geq \log\frac{T_m(V_m-1)}{\delta} = H_m \iff \tau \leq \frac{b}{H_m}, \tag{16}$$

where the logarithm step uses monotonicity of $\log$, and the last step divides by the positive quantities τ, H_m. Taking $b = \Delta_m$ gives the exact threshold $\tau^{\mathrm{cert}}_{m,\mathrm{ex}} = \Delta_m/H_m$; taking $b = \Delta_m - K_m\varepsilon_m$ (> 0) gives the approximate threshold $\tau^{\mathrm{cert}}_{m,\mathrm{ap}} = (\Delta_m - K_m\varepsilon_m)/H_m$. This establishes the certified intervals defined by the first two equalities in equation 7.

Step 7 (Threshold comparison). Subtracting the two thresholds, $\tau^{\mathrm{cert}}_{m,\mathrm{ex}} - \tau^{\mathrm{cert}}_{m,\mathrm{ap}} = \left[\Delta_m - (\Delta_m - K_m\varepsilon_m)\right]/H_m = K_m\varepsilon_m/H_m$, which is the last equality in equation 7. Since $K_m \geq 0$, $\varepsilon_m \geq 0$, $H_m > 0$, this quantity is ≥ 0, i.e., the certified threshold for exact deployment is no smaller than the guaranteed threshold for approximate deployment. This difference $K_m\varepsilon_m/H_m$ is precisely the gap between the two certified thresholds given by this perturbation analysis; for fixed K_m, H_m, it vanishes as $\varepsilon_m \to 0$. This gap arises from the Lipschitz perturbation bound equation 10, the uniform margin lower bound, and the first-order relaxation in Step 5, all of which are sufficiency-oriented relaxations for guaranteeing $Q_{m,\tau} \geq 1-\delta$; hence the resulting thresholds may be conservative, and we do not claim they are tight or achievable. This completes the proof of equation 6 and equation 7, with all bounds holding uniformly over $c \in \mathcal{C}_m$. □

Remark 3 (Scope and Tightness). *(i) Since $\Delta_m > 0$ (respectively $\Delta_m - K_m\varepsilon_m > 0$), the target token is always the unique argmax, so greedy decoding hits it with probability 1; top-k ($k \geq 1$) and top-p sampling (temperature scaling preserves order, and the highest-probability token is always retained) only remove competing terms, and after renormalization the target probability does not decrease, so the bounds in equation 6–equation 7 hold* a fortiori *for these decoding schemes. (ii) Within the certifiable regime, $\tau^{\mathrm{cert}}_{m,\mathrm{ap}} > 0$, so the interval is non-empty; to certify at a given τ_0, the sufficient conditions for exact and approximate deployment are respectively $\Delta_m \geq \tau_0 H_m$ and $\Delta_m - K_m\varepsilon_m \geq \tau_0 H_m$. (iii) τ^{cert} derives from the looser first-order bound equation 15; using equation 6 directly may yield a larger certifiable range, so the thresholds above are conservative closed-form sufficient values. (iv) All of the above are guarantees under the same perturbation analysis, not claims about the actual hit probability.*

E Additional Experimental Details

E.1 Experimental Setup Details

E.1.1 Setup for Fig. 1

All three experiments follow a common protocol: for MF, we follow the fingerprint generation procedure in §3.1 to generate 100 fingerprints, with fingerprint tokens mapped exclusively onto

invisible zero-width Unicode characters, and verify the resulting fingerprint for TRR on each model individually. For REEF, we generate a source fingerprint on the corresponding base model and report its CKA similarity score. All experiments are repeated over 10 independent runs, and we report the mean and standard deviation of TRR (for MF) and similarity score (for REEF) for each setting.

Setup for Fig. 1a. We use three models from the Qwen family with different hidden sizes: Qwen2.5-7B (3584), Qwen1.5-7B-Chat (4096), and Qwen3-14B-Instruct (5120), with Qwen2.5-7B as the base model, and Qwen1.5-7B-Chat and Qwen3-14B-Instruct serving as cross-hidden-size derivatives.

Setup for Fig. 1b. We use two Mistral-7B versions with different vocabulary sizes: Mistral-7B-v0.1 (32000 tokens) and Mistral-7B-v0.3 (32768 tokens), with Mistral-7B-v0.1 as the base model, and Mistral-7B-v0.3 serving as a cross-vocabulary derivative.

Setup for Fig. 1c. We use Llama-2-7B as the base model, together with its derivatives Llama-2-7B-HF, Llama-2-7B-Chat, Llama-2-7B-Chat-HF, Vicuna-7B-v1.3, and Vicuna-7B-v1.5, each formatted under the following five conversation templates:

- Template 1: "`You are a helper.`"
- Template 2: "`You are a knowledgeable and concise AI assistant.`"
- Template 3: "`Answer the user's question as accurately as possible.`"
- Template 4: "`You are ChatGPT, a large language model. Follow the user's instructions carefully.`"
- Template 5: "`You are an expert assistant. Provide clear and correct answers.`"

E.1.2 Setup for §4.1

Setup for Effectiveness. We evaluate MF against REEF, ZeroPrint, and PlugAE across four base families: Llama-2-7B, Mistral-7B-v0.1, Qwen2.5-7B, and Phi-3-mini. For MF, we follow the procedure in §3.1 to generate 100 fingerprints, using 5 invisible tokens and 600 optimization steps. For each baseline, a fingerprint is generated on the base model according to its own method and verified on the same related and unrelated models. Our comparison of MF against baselines is bounded by metric compatibility: REEF and ZeroPrint report similarity scores, while MF and PlugAE report TRR on related models and FPR on unrelated models, so these values are not directly comparable in raw magnitude. We unify both onto a $[0, 1]$ scale so their separability between related and unrelated models can be assessed directly. All experiments use five seeds, and we report the mean and standard deviation of each metric per model. For the experiment shown in Table 5, we use the following two templates:

- Template 1: "`You are a helper.`"
- Template 2: "`You are a helper. Please answer the following question.`"

Setup for Robustness. We detail the implementation of each robustness experiment reported in §4, covering model modifications (fine-tuning, pruning, feature permutation/scaling) and adaptive attacks (input manipulation, vocabulary manipulation) against MF. All experiments use 100 fingerprints per setting, repeated over 5 seeds, and report TRR (%) as mean $\pm$ standard deviation. In every setting, the fingerprint is generated on the base model, after which the suspect model, the fingerprint itself, or the vocabulary is modified depending on the attack scenario. A high TRR indicates that MF still detects the protected model despite the adversary's attempted evasion.

(1) *Fine-tuning*. We fine-tune each base model on datasets of size 200, 400, and 800 samples from OpenHermes-2.5 using QLoRA, and evaluate whether the fingerprint generated on the base model still triggers the target response on the resulting fine-tuned model. This simulates an adversary who fine-tunes a stolen model checkpoint before redistribution.

(2) *Model Pruning*. Following Wanda Sun et al. (2024), we prune each base model at ratios of 0.10, 0.15, and 0.20, and evaluate whether the fingerprint generated on the base model still triggers the target response on the resulting pruned model.

(3) *Feature Permutation and Scaling.* Following Zhang et al. (2025a), we test two weight-tampering attacks that aim to covertly distort the model's internal feature space, without affecting its normal functionality, in an attempt to destroy the fingerprint. Feature permutation perturbs the model's weight matrices by introducing a perturbation vector. Feature scaling scales the parameters of two adjacent layers by a common factor and its inverse (so that the two cancel out). In both cases, we evaluate whether the fingerprint generated on the base model still triggers the predefined response on the resulting modified model.

(4) *Input Manipulation.* We assume an adaptive adversary who knows a fingerprint may be embedded in the prompt but does not know which specific tokens carry it. Under this assumption, we apply the following operations directly to the fingerprint, and test whether the perturbed fingerprint still triggers the target response on the base model:

- *Word deletion*: randomly deleting several words from the fingerprint.
- *Word addition*: randomly inserting several words into the fingerprint.
- *Synonym replacement*: replacing several words in the fingerprint with synonyms.
- *Paraphrasing*: rewriting the fingerprint using a separate, smaller LLM before submitting it to the target model. Specifically, we use semantically meaningful tokens as fingerprint tokens and Qwen2.5-0.5B-Instruct as a lightweight paraphrasing assistant. The fingerprint is inserted into the template "`Rewrite this query naturally, keeping the exact meaning: '{text}'`", and the assistant model generates a rewritten version using sampling-based decoding (`do_sample=True`, temperature $= 0.7$). Because the assistant model regenerates the text token-by-token according to its own natural-language understanding, this process typically destroys or discards the invisible fingerprint tokens originally appended to the prompt. The resulting rewritten text is then submitted to the target model, and we measure whether it still triggers the predefined response (i.e., TRR).
- *Homoglyph*: replacing characters with visually similar but semantically unrelated characters (e.g., "`word`" to "`work`").

(5) *Vocabulary Manipulation.* We assume an adaptive adversary who knows the fingerprint may reside in newly added or otherwise anomalous vocabulary tokens, but does not know exactly which ones. Under this assumption, we first modify the base model's vocabulary according to one of the following rules, and then test whether the fingerprint still triggers the target response on the vocabulary-modified model.We note that this vocabulary manipulation is necessarily local and small-scale. Otherwise, the model itself would become non-functional.

- *Vocabulary addition*: inserting an additional 1% of tokens into the original vocabulary.
- *Vocabulary deletion*: randomly removing 1% of tokens from the vocabulary.
- *Token renaming*: renaming a fingerprint token's surface string (e.g., `<fp_001>` to `<extra_217>`) while keeping its corresponding embedding unchanged.
- *Token-ID remapping*: reassigning 10% tokens' ID (e.g., token A: `id=50001` to `id=32007`) while keeping its surface string and embedding unchanged.

Setup for Efficiency. We compare MF against REEF, ZeroPrint, and PlugAE on Llama-2-7B, Mistral-7B-v0.1, Qwen2.5-7B, and Phi-3-mini, measuring the end-to-end time (s) and peak memory (GB) required to generate 100 fingerprints on each base model. For MF, fingerprint tokens are mapped exclusively onto invisible zero-width Unicode characters, and the reported time covers the complete pipeline. All experiments are run on a single machine (NVIDIA DGX-2, Intel Xeon Platinum 8168 CPU, Tesla V100-SXM3-32GB GPU, 1.5TB DDR4 RAM), with 3 seeds. We report the mean and standard deviation of time and memory for each method. Since Time is measured while multiple experiments run concurrently on the shared machine, it is subject to greater variance from system load than Memory (GB).

Setup for Stealthiness. We compare MF against PlugAE on 100 clean user prompts, scoring PPL with four external reference LLMs (OPT-6.7B, Mistral-7B-v0.1, Qwen2.5-7B, Phi-3-mini) under five conditions: *Clean* (the original prompt), *Fingerprint* (the prompt with the Llama-2-7B-generated fingerprint appended), and *Fingerprint* combined with each of three attacks, *Paraphrase* (a single random adjacent-word swap), *Random Deletion* (each word independently deleted with

probability $p = 0.1$, retaining at least one word), and *Random Addition* (a common word from {and, the, a, is, of, to, in, it} independently inserted after each word with probability $p = 0.1$). For MF, whose fingerprint combines invisible and visible characters, the fingerprint suffix is appended after the clean text, and PPL is computed. For PlugAE, whose fingerprint is fully visible (e.g., "mkahg"), the entire sequence is scored. All attacks are applied using seed 42, and we report the average PPL.

Setup for Harmlessness. We evaluate whether embedding the fingerprint (a combination of invisible and visible characters) degrades main-task performance on Llama-2-7B, Mistral-7B-v0.1, Qwen2.5-7B, and Phi-3-mini, by comparing the accuracy of each base model against its fingerprinted counterpart (100 fingerprints) on four benchmarks: MMLU (knowledge/QA), GSM8K (mathematical reasoning), HellaSwag (commonsense reasoning), and TruthfulQA (truthfulness).

E.1.3 Setup for §4.2

Setup for Number of Fingerprint Tokens. We fix each fingerprint token to a random 5-character string and vary the number of fingerprint tokens $k \in \{2, 3, 4\}$ to test whether MF's effectiveness depends on the number of tokens used, and whether additional tokens improve stability or detection rate. All experiments are conducted on Llama-2-7B, Mistral-7B-v0.1, Qwen2.5-7B, and Phi-3-mini, with fingerprint tokens mapped exclusively onto invisible zero-width Unicode characters (100 fingerprints), and repeated over 5 seeds. We report the mean and standard deviation of TRR for each setting.

Setup for Prompt Length. The default user prompt length used in other experiments is 5. Here, we instead vary the optimization prompt length across $\{2, 10, 15, 20\}$ to test whether MF's effectiveness depends on prompt length, rather than the fingerprint tokens themselves, which remain fixed and mapped exclusively onto invisible zero-width Unicode characters. Using a base pool of 100 question-answer pairs, we construct prompts of each target length accordingly. All experiments are conducted on Llama-2-7B, Mistral-7B-v0.1, Qwen2.5-7B, and Phi-3-mini, and repeated over 5 seeds. We report the mean and standard deviation of TRR for each setting.

Setup for Number of Models Used for Fingerprint Optimization. We test whether jointly optimizing the fingerprint over more models improves generalization to unseen derivatives. For each of the four base families (Llama-2-7B, Mistral-7B-v0.1, Qwen2.5-7B, Phi-3-mini), we jointly optimize MF's fingerprint over $M \in \{2, 3, 4\}$ models, progressively adding one derivative at a time (e.g., for Llama-2-7B: M=2: {Llama-2-7B, Llama-2-7B-HF}; M=3: +Llama-2-7B-Chat-HF; M=4: +Llama-2-7B-Chat), with fingerprint tokens mapped exclusively onto invisible zero-width Unicode characters. For each M, we evaluate TRR on the 6 related models within the family (including models not included in optimization) and FPR on the 3 unrelated models. All experiments generate 100 fingerprints and are repeated over 5 seeds. We report the mean and standard deviation of TRR and FPR for each setting.

Setup for Value of λ. We test the sensitivity of MF to the margin weight λ, which balances the target-generation loss against the margin-based loss. We vary $\lambda \in \{0.2, 0.5, 0.8\}$, with the number of fingerprint tokens fixed at 5, mapped exclusively onto invisible zero-width Unicode characters. All experiments are conducted on Llama-2-7B, Mistral-7B-v0.1, Qwen2.5-7B, and Phi-3-mini, and repeated over 5 seeds. We report the mean and standard deviation of TRR for each setting.

Table 5: Effect of conversation template (%). Used templates are shown in Appendix E.1.

Template	Method	Llama-2-7B	Mistral-7B-v0.1	Qwen2.5-7B	Phi-3-mini
Template 1	REEF	26.28±4.34	96.18±0.30	59.33±1.37	96.67±0.26
	ZeroPrint	59.25±1.81	63.83±2.39	55.10±1.32	59.60±1.24
	PlugAE	94.71±0.32	81.57±0.34	51.51±0.03	79.39±0.50
	MF	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**
Template 2	REEF	25.02±3.96	93.26±0.37	57.94±1.48	95.85±0.30
	ZeroPrint	58.95±1.84	64.54±1.92	56.24±1.07	60.97±0.72
	PlugAE	94.71±0.32	81.57±0.34	51.51±0.03	79.39±0.50
	MF	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**

Table 6: Effect of decoding temperatures (%).

Temperature	Method	Llama-2-7B	Mistral-7B-v0.1	Qwen2.5-7B	Phi-3-mini
0.2	REEF	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	ZeroPrint	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	PlugAE	94.32±0.29	83.02±0.31	82.71±0.22	79.60±0.31
	MF	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**
0.5	REEF	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	ZeroPrint	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	PlugAE	92.95±0.28	80.04±0.34	79.66±0.78	75.68±0.43
	MF	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**
1.0	REEF	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	ZeroPrint	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
	PlugAE	86.68±0.45	70.47±0.38	68.82±0.39	63.46±0.43
	MF	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**

Table 7: Efficiency comparison of MF against baselines.

Model	Method	Time (s)	Memory (GB)	Model	Method	Time (s)	Memory (GB)
Llama-2-7B	REEF	43.87±29.59	12.82±0.00	Mistral-7B-v0.1	REEF	47.74±50.82	13.60±0.00
	ZeroPrint	519.59±303.03	13.67±0.03		ZeroPrint	426.18±102.23	14.10±0.01
	PlugAE	2477.41±50.89	17.47±0.00		PlugAE	2467.38±120.78	18.67±0.00
	MF	**127.02±3.00**	**14.18±0.00**		**MF**	**138.76±8.16**	**15.11±0.00**
Qwen2.5-7B	REEF	38.36±17.82	14.33±0.00	Phi-3-mini	REEF	23.34±19.88	7.17±0.00
	ZeroPrint	539.50±161.24	14.75±0.00		ZeroPrint	644.64±162.70	8.17±0.03
	PlugAE	3936.51±193.68	19.98±0.00		PlugAE	6069.18±112.20	10.88±0.00
	MF	**133.39±7.85**	**15.75±0.00**		**MF**	**381.88±2.03**	**8.33±0.00**

Table 8: PPL of Llama-2-7B-generated fingerprints under MF and PlugAE.

	OPT-6.7B			Mistral-7B-v0.1			Qwen2.5-7B			Phi-3-mini		
Setting	PlugAE	MF	**Drop↓**	PlugAE	MF	**Drop↓**	PlugAE	MF	**Drop↓**	PlugAE	MF	**Drop↓**
Clean	27.66	27.66	**0.00**	41.01	41.01	**0.00**	43.79	43.79	**0.00**	39.60	39.60	**0.00**
Fingerprint	108.84	38.72	**70.12**	103.44	53.08	**50.36**	108.07	53.83	**54.24**	98.81	46.60	**52.21**
Fingerprint + Paraphrase	268.52	120.34	**148.18**	263.59	123.91	**139.68**	248.36	134.15	**114.21**	553.65	356.75	**196.90**
Fingerprint + Deletion	366.21	106.45	**259.76**	323.68	173.18	**150.50**	317.05	197.48	**119.57**	609.54	297.58	**311.96**
Fingerprint + Addition	365.55	72.48	**293.07**	374.99	100.92	**274.07**	362.98	140.48	**222.50**	512.63	128.36	**384.27**

Table 9: Harmlessness of MF.

Benchmark	Model	Llama-2-7B	Mistral-7B-v0.1	Qwen2.5-7B	Phi-3-mini
MMLU	Original	41.74	60.06	71.94	70.08
	+Fingerprint	41.64	60.08	71.93	70.08
	Drop↓	**0.10**	**-0.02**	**0.01**	**0.00**
GSM8K	Original	14.12	37.98	83.02	78.85
	+Fingerprint	14.03	37.83	83.10	78.85
	Drop↓	**0.09**	**0.15**	**-0.08**	**0.00**
HellaSwag	Original	57.41	61.56	60.02	59.04
	+Fingerprint	57.28	61.51	60.03	59.04
	Drop↓	**0.13**	**0.04**	**-0.01**	**0.00**
TruthfulQA	Original	24.73	28.03	39.05	36.23
	+Fingerprint	24.85	28.03	39.05	36.23
	Drop↓	**-0.12**	**0.00**	**0.00**	**0.00**

Table 10: Effect of the number of fingerprint tokens k on TRR (%).

k	Llama-2-7B	Mistral-7B-v0.1	Qwen2.5-7B	Phi-3-mini
2	100.0±0.0	100.0±0.0	99.60±0.80	100.0±0.0
3	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0
4	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0

Table 11: Effect of user prompt length (used during fingerprint optimization) on TRR (%). The default user prompt length in other experiments is 5.

Prompt Length	Llama-2-7B	Mistral-7B-v0.1	Qwen2.5-7B	Phi-3-mini
2	100.0±0.0	100.0±0.0	81.40±15.54	100.0±0.0
10	100.0±0.0	100.0±0.0	95.60±3.37	100.0±0.0
15	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0
20	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0

Table 12: Effect of the number of models M used during fingerprint optimization on TRR (%, related models) and FPR (%, unrelated models). The specific models included at each M are listed below.

Base Model	Type	Test Model	$M=2$	$M=3$	$M=4$
Llama-2-7B[a]	Related	Llama-2-7B	100.0±0.00	100.0±0.00	100.0±0.00
		Llama-2-7B-HF	100.0±0.00	100.0±0.00	100.0±0.00
		Llama-2-7B-Chat-HF	88.1±21.80	100.0±0.00	100.0±0.00
		Llama-2-7B-Chat	100.0±0.00	100.0±0.00	100.0±0.00
		Vicuna-7B-v1.3	34.40±27.45	45.80±34.92	36.68±33.77
		Vicuna-7B-v1.5	96.0±3.65	99.0±1.30	98.1±3.56
	Unrelated	Mistral-7B-v0.1	0.0±0.00	0.0±0.00	0.0±0.00
		Qwen2.5-7B	0.0±0.00	0.0±0.00	0.0±0.00
		Phi-3-mini	0.0±0.00	0.0±0.00	0.0±0.00
Mistral-7B-v0.1[b]	Related	Mistral-7B-v0.1	100.0±0.00	100.0±0.00	100.0±0.00
		Mistral-7B-Instruct-v0.1	100.0±0.00	100.0±0.00	100.0±0.00
		Mistral-7B-Instruct-v0.2	99.8±0.24	100.0±0.00	100.0±0.00
		Mistral-7B-OpenOrca	95.20±1.54	99.9±0.20	100.0±0.00
		Mistral-7B-Instruct-v0.3	99.7±0.60	100.0±0.00	99.8±0.40
		Mistral-7B-v0.3	100.0±0.00	100.0±0.00	100.0±0.00
	Unrelated	Llama-2-7B	0.0±0.00	0.0±0.00	0.0±0.00
		Qwen2.5-7B	0.0±0.00	0.0±0.00	0.0±0.00
		Phi-3-mini	0.0±0.00	0.0±0.00	0.0±0.00
Qwen2.5-7B[c]	Related	Qwen2.5-7B	100.0±0.00	100.0±0.00	99.8±0.40
		Qwen2.5-7B-Instruct	100.0±0.00	100.0±0.00	99.5±1.00
		Qwen3-14B-Instruct	0.0±0.00	100.0±0.00	100.0±0.00
		Qwen-14B-Chat	0.0±0.00	0.0±0.00	0.0±0.00
		Qwen-7B-Chat	0.0±0.00	0.0±0.00	100.0±0.00
		Qwen1.5-7B-Chat	0.0±0.00	0.0±0.00	0.0±0.00
	Unrelated	Llama-2-7B	0.0±0.00	0.0±0.00	0.0±0.00
		Mistral-7B-v0.1	0.0±0.00	0.0±0.00	0.0±0.00
		Phi-3-mini	0.0±0.00	0.0±0.00	0.0±0.00
Phi-3-mini[d]	Related	Phi-3-mini	100.0±0.00	100.0±0.00	100.0±0.00
		Phi-3-mini-4k-base	100.0±0.00	100.0±0.00	100.0±0.00
		Phi-3-mini-4k-instruct	100.0±0.00	100.0±0.00	100.0±0.00
		Phi-3-mini-128k-instruct	96.5±3.27	91.0±8.44	100.0±0.00
		Phi-3-small-8k-instruct	0.0±0.00	0.0±0.00	0.0±0.00
		Phi-3-small-128k-instruct	0.0±0.00	0.0±0.00	0.0±0.00
	Unrelated	Llama-2-7B	0.0±0.00	0.0±0.00	0.0±0.00
		Mistral-7B-v0.1	0.0±0.00	0.0±0.00	0.0±0.00
		Qwen2.5-7B	0.0±0.00	0.0±0.00	0.0±0.00

[a] Llama-2-7B: M=2: {Llama-2-7B, Llama-2-7B-HF}; M=3: +Llama-2-7B-Chat-HF; M=4: +Llama-2-7B-Chat.
[b] Mistral-7B-v0.1: M=2: {Mistral-7B-v0.1, Mistral-7B-Instruct-v0.1}; M=3: +Mistral-7B-Instruct-v0.2; M=4: +Mistral-7B-Instruct-v0.3.
[c] Qwen2.5-7B: M=2: {Qwen2.5-7B, Qwen2.5-7B-Instruct}; M=3: +Qwen3-14B-Instruct; M=4: +Qwen-7B-Chat.
[d] Phi-3-mini: M=2: {Phi-3-mini, Phi-3-mini-4k-base}; M=3: +Phi-3-mini-4k-instruct; M=4: +Phi-3-mini-128k-instruct.
Blue: models with a different vocabulary from others.
Green and Dark-green: models with a different hidden size from others.
Red: models differing in both vocabulary and hidden size from others.

Table 13: Effect of the margin weight λ on TRR (%).

λ	Llama-2-7B	Mistral-7B-v0.1	Qwen2.5-7B	Phi-3-mini
0.2	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
0.5	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
0.8	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00

Table 14: Cross-family fingerprint optimization: TRR (%, related models) and FPR (%, unrelated models) when a single fingerprint is jointly optimized over two models from different families.

Used Models (Optimization)	Type	Test Model	TRR/FPR (%)
Llama-2-7B + Mistral-7B-v0.1	Related	Llama-2-7B	100.00±0.00
		Mistral-7B-v0.1	100.00±0.00
	Unrelated	Qwen2.5-7B	0.00±0.00
		Phi-3-mini	0.00±0.00
Mistral-7B-v0.1 + Qwen2.5-7B	Related	Mistral-7B-v0.1	100.00±0.00
		Qwen2.5-7B	100.00±0.00
	Unrelated	Llama-2-7B	0.00±0.00
		Phi-3-mini	0.00±0.00
Qwen2.5-7B + Phi-3-mini	Related	Qwen2.5-7B	100.00±0.00
		Phi-3-mini	100.00±0.00
	Unrelated	Llama-2-7B	0.00±0.00
		Mistral-7B-v0.1	0.00±0.00
Phi-3-mini + Llama-2-7B	Related	Phi-3-mini	100.00±0.00
		Llama-2-7B	100.00±0.00
	Unrelated	Mistral-7B-v0.1	0.00±0.00
		Qwen2.5-7B	0.00±0.00

E.2 Experimental Result Details

E.2.1 Detailed Results for Effectiveness

Table 5 and Table 6 compare MF with baselines under different conversation templates and decoding temperatures, respectively.

E.2.2 Detailed Results for Efficiency

Table 7 reports the time and memory used for fingerprint generation.

E.2.3 Detailed Results for Stealthiness

Table 8 shows the detailed stealthiness results. *Clean* denotes fingerprint-free inputs, *Fingerprint* denotes fingerprint-injected inputs optionally combined with an attack (+paraphrase, random deletion, random addition), and *Drop* denotes the PPL reduction of MF relative to PlugAE.

E.2.4 Detailed Results for Harmlessness

Table 9 shows the detailed harmlessness results, testing accuracy (%) on four benchmarks and comparing the base model against its fingerprinted counterpart. *Drop* denotes the difference.

E.2.5 Detailed Results for Ablation

Detailed results of the ablation experiments are provided in Table 10 (number of fingerprint tokens), Table 11 (prompt length), Table 12 (number of models used for fingerprint optimization), and Table 13 (value of λ).

F Additional Extension Details

F.1 Extension 1: Cross-family Lineage Verification

MF can be potentially extended to verify lineage across architecturally distinct model families, such as Llama and Mistral, using the same optimization process as §3, with cross-family models jointly included. To test it, we consider four pairs of base models from different families: (Llama-2-7B, Mistral-7B-v0.1), (Mistral-7B-v0.1, Qwen2.5-7B), (Qwen2.5-7B, Phi-3-mini), and (Phi-3-mini, Llama-2-7B), jointly optimizing a single fingerprint over each pair. For each pair, we evaluate TRR on related models from each family involved in optimization and FPR on unrelated models, over 5 seeds and 100 fingerprints (fingerprint tokens mapped exclusively onto invisible zero-width Unicode characters), reporting the mean and standard deviation of both metrics. As shown in Table 14, MF achieves strong effectiveness across families with heterogeneous hidden dimensions and vocabularies. We have evaluated only pairwise combinations of four families. Testing across more families jointly would provide stronger evidence for this extension's scalability.

F.2 Extension 2: Content Attribution

Content attribution refers to, given a suspicious piece of LLM-generated content, determining whether it originates from a protected model, using our proposed MF. Existing token-level watermarks Kirchenbauer et al. (2023) partition the vocabulary into a fixed Red set R and Green set G via a shared secret key, and at each generation step add a fixed bias δ to the green-set logits, but *only at high-entropy positions* (where the model's next-token distribution is uncertain), leaving low-entropy (near-deterministic) positions unbiased. Because biasing only occurs at high-entropy positions, a generated sequence with few such positions yields a weak, easily-missed green-token signal.

How can MF be used for Content Attribution? Since MF only optimizes fingerprint embeddings, which can be mapped to arbitrary tokens, we use invisible tokens as an example to demonstrate how MF can be applied to content attribution. Specifically, after generating the visible watermarked text, the model owner recomputes the red/green hit counts n_R, n_G on it (by re-tokenizing and re-scoring high-entropy positions). If the natural green signal is already strong enough ($n_R \leq n_G$), no further action is taken. Otherwise ($n_R > n_G$, indicating a weak natural signal), we reinforce it with a third, Blue table B of invisible tokens, reusing the same visually imperceptible fingerprint tokens used to instantiate MF's fingerprint: one blue token is inserted immediately before *every* space in the visible text, without altering the visible content itself. For example, given text "`My dog is so cute and fluffy.`" with a weak natural signal, a blue token would be inserted before each of its 6 spaces:

$$\texttt{My}\langle b\rangle\ \texttt{dog}\langle b\rangle\ \texttt{is}\langle b\rangle\ \texttt{so}\langle b\rangle\ \texttt{cute}\langle b\rangle\ \texttt{and}\langle b\rangle\ \texttt{fluffy.}$$

where $\langle b\rangle$ denotes an (otherwise invisible) blue token shown here for illustration. The text a reader actually sees remains unchanged. To verify whether a piece of suspicious content originates from a protected model, the owner checks two independent channels and takes their logical OR: (i) *blue-table check*: count the occurrences c_B of MF's fingerprint tokens in the text, and flag a match if c_B reaches a preset threshold τ_B; (ii) *green-table check*, as a fallback in case blue tokens have been stripped: recompute the green-token hit rate on the text and perform a one-sided statistical test against the null hypothesis that no watermark is present, flagging a match if the resulting statistic exceeds a preset threshold. If the blue-table channel fires, this is reported as the reason for attribution. Otherwise, if only the green-table channel fires, that is reported instead. Because both the fingerprint tokens and the insertion rule are chosen and kept secret by the model owner, this combination is highly unlikely to arise by coincidence in content not produced under MF.

Experimental Results for Content Attribution. We compare MF (whose fingerprint tokens combine invisible and visible characters) against four state-of-the-art watermarking baselines: KGW Kirchenbauer et al. (2023), OPT Wouters (2024), MorphMark Wang et al. (2025b), and WaterMod Park et al. (2026), evaluating TPR@1% (the true positive rate at a 1% false positive rate) and Best F1 both in a clean setting and under a 30% word-substitution attack (Word-S/30%, randomly replacing 30% of the words in the input text with WordNet synonyms), alongside PPL, generation/detection time, and memory usage. The results, based on 5 seeds and 400-token text

(one watermarked, one clean), are shown in Table 15. MF achieves the highest TPR@1% and Best F1 in both settings, the least degradation to text quality (18.19 PPL), and greater efficiency with comparable memory usage. We note that MF's TPR@1% is slightly higher under the Word-S/30% attack than in the clean setting (97.15% to 98.55%). Given the overlapping error bars, this small increase reflects the 1% FPR threshold being recalibrated separately for each setting rather than a genuine gain in detectability, consistent with MF's blue-channel signal being largely invariant to word substitution: since Word-S/30% replaces words one-for-one rather than deleting or inserting them, the number of inter-word spaces, and hence the number of blue tokens available for detection, is preserved almost exactly, leaving the dominant detection signal essentially untouched by this particular attack. These results show that MF's fingerprint mechanism transfers potentially to content attribution, with robustness that holds under a targeted word-level attack.

Table 15: Overall comparison with content-watermarking baselines.

	TPR@1% (%)		Best F1					
Method	**Clean**	**Word-S/30%**	**Clean**	**Word-S/30%**	**PPL**	**Gen. Time (m)**	**Det. Time (m)**	**Memory (GB)**
KGW	96.35±0.65	79.40±4.37	97.17±0.34	92.10±0.82	33.06±1.75	58.80±0.54	14.09±1.14	3.14±0.00
OPT	89.40±1.83	37.50±5.70	94.65±0.68	79.80±1.28	23.23±2.62	58.66±1.25	6.92±0.14	4.32±0.00
MorphMark	96.65±0.58	71.05±10.33	97.96±0.22	90.79±0.66	24.40±1.37	59.70±6.43	17.07±2.29	2.95±0.00
WaterMod	61.70±7.37	0.15±0.22	90.97±1.43	66.67±0.00	21.52±1.19	55.45±1.49	82.7±1.49	4.12±0.00
MF	**97.15±0.45**	**98.55±0.23**	**98.75±0.45**	**98.55±0.23**	**18.19±0.98**	**19.72±0.17**	**2.55±0.45**	**2.93±0.00**

Limitations of Using MF for Content Attribution. MF is primarily designed for model fingerprinting, not content attribution, and this extension is intended only to illustrate that MF's fingerprint mechanism can be repurposed for this task. We have not evaluated its robustness under content-level attacks such as paraphrasing or partial re-generation, nor measured the false-positive rate of its dual-channel verification on unrelated, non-MF-generated text. Moreover, several assumptions underlie this design, each of which introduces a corresponding limitation. First, the mechanism assumes redistribution preserves the character sequence, where invisible fingerprint tokens can be stripped by copy-pasting, Markdown conversion, or platform sanitization, since such pipelines typically normalize non-printable characters by default. Visible fingerprint tokens survive these operations but perturb the text and are consequently easier to notice. Second, verification assumes the fingerprint tokens and their insertion rule remain secret. If leaked, an adversary could strip them from protected content to evade detection, or insert them into unrelated content to falsely implicate the protected model. Finally, heavy paraphrasing or full re-generation removes the fingerprint tokens entirely, and MF currently has no mechanism to recover attribution once this occurs.

G Discussion & Future Work

Fingerprint token and vocabulary strategy trade-offs. MF is agnostic to the choice of fingerprint token and to how that token is incorporated into the vocabulary, but both choices introduce practical trade-offs. Invisible tokens preserve the visible prompt exactly, improving stealthiness, but they can be removed by input sanitization. Visible tokens are more resistant to such sanitization, but they introduce a small perturbation to the prompt. A similar trade-off arises in vocabulary integration. Adding new vocabulary entries is appropriate for models trained entirely in-house when no public reference model exists, because an adversary cannot independently verify the original vocabulary size. However, for models derived from publicly documented architectures, such as fine-tuned Llama-2-7B, vocabulary expansion creates a detectable mismatch from the known vocabulary size. In this setting, overwriting rarely used existing tokens is preferable because it preserves the original vocabulary size and therefore avoids this structural anomaly. As illustrated in Fig. 3 in §4, MF can repurpose such existing tokens as fingerprint tokens without being flagged as embedding outliers.

Why does generalization fail for some derivatives regardless of M? We validate MF's generalization along three settings, ordered by increasing distance from optimization. (1) Table 1 reports performance on models included in joint optimization. (2) Table 2 evaluates models never included in optimization but obtained by applying fine-tuning, pruning, or feature manipulation to the protected model. (3) The number-of-models ablation (Table 12) isolates generalization to models that are both excluded from optimization and released independently by third parties. These three tests confirm that MF's generalization holds both for models included in optimization and for derivatives

with a documented lineage to them. Notably, our results in Table 12 reveal a distinction among related-but-unoptimized derivatives: some (e.g., Vicuna-7B-v1.5, Llama-2-7B-Chat-HF) show clear gains in TRR as M increases, while others (e.g., Qwen-14B-Chat, Qwen1.5-7B-Chat) remain at or near 0% regardless of how many related models are jointly optimized. We hypothesize this reflects a boundary condition on Remark 1: since the shared embedding Z is only meaningful insofar as each model's own embedding function interprets it consistently, generalization should degrade once a derivative's tokenizer, embedding geometry, or alignment procedure diverges substantially from the models used during optimization. This is consistent with our observation that the models exhibiting persistent failure tend to span larger generational or architectural gaps from the optimized set (e.g., Qwen-14B-Chat and Qwen1.5-7B-Chat predate the Qwen2.5/Qwen3 lineage used for optimization), whereas models within the same generation transfer successfully even without being directly optimized. This suggests that MF's transferability is bounded not by the number of models jointly optimized, but by the continuity of the model lineage they span. Extending coverage to a genuinely distant derivative likely requires including a representative from that lineage in the optimization set, rather than simply increasing M within a single lineage.

MF Applicability. MF can protect multiple models simultaneously using a single fingerprint. This fingerprint remains effective at identifying suspect models that are derived from a protected model via fine-tuning, pruning, feature manipulation, or deployed under a different conversation template or decoding temperature, and continues to work even when the adversary combines input-level or vocabulary-level manipulations, or embedding-level outlier detection, on top of these modifications. This is validated in §4.

However, MF is not guaranteed to work in every setting. Table 1 and Table 12 label test models as *related* if they *share a family name* with the protected model, but this does not by itself establish a documented derivation path. Within each family, some related models are *derivatives* with a documented fine-tuning or alignment path from the protected checkpoint (e.g., Llama-2-7B-Chat from Llama-2-7B, Qwen2.5-7B-Instruct from Qwen2.5-7B), while others are *family siblings* released under the same family name but trained independently, with no such path (e.g., Qwen1.5-7B-Chat, Qwen-14B-Chat relative to Qwen2.5-7B). As discussed above, MF's transferability to an unoptimized model depends on its lineage continuity with the models included during optimization. A suspect model whose tokenizer, embedding geometry, or alignment procedure diverges substantially from the protected family may evade detection, unless a representative from that lineage was itself included in optimization. This is consistent with our observation in Table 12 that the models exhibiting persistent 0% TRR regardless of M (e.g., Qwen-14B-Chat, Qwen1.5-7B-Chat, Phi-3-small variants) are family siblings rather than documented derivatives, whereas MF sustains near-100% TRR on every model with a documented derivation path we tested. Arguably, such a divergent model would no longer be considered part of the *same model family* in the first place. Likewise, MF is not designed for adversaries who discard the base model's vocabulary and embedding layer entirely, or who replace the underlying architecture outright.

Moreover, we also consider two existing LLM fingerprint-removal methods: (1) MEraser Zhang et al. (2025b), which targets fine-tuning-induced backdoor associations that do not apply to MF, since MF freezes all model parameters and does not embed its fingerprint via fine-tuning; and (2) TFA/SVA Peng et al. (2026), which suppresses fingerprint responses by comparing outputs across multiple models within an adversary-constructed LLM ensemble. Since the adversary cannot know in advance which queries carry a fingerprint, they must apply ensemble-based filtering to every query. This multiplies inference cost by the number of auxiliary models used. Such a huge cost makes the removal unappealing in practice for a rational adversary.