RMF: RUNTIME MODEL FINGERPRINTING FOR DETECTING API MODEL SUBSTITUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Third-party LLM aggregation platforms provide convenient access to multiple models through unified APIs, yet their opaque backend makes it difficult for users to verify which model actually serves a request, enabling API fraud through model substitution. Existing black-box fingerprinting methods primarily perform pre-deployment verification and therefore cannot reliably authenticate the model during actual task execution. To address this limitation, we propose a runtime model fingerprinting method (RMF) that embeds lightweight fingerprint probes directly into real-world task prompts. RMF constructs compact behavioral fingerprints from model-specific preferences over underdetermined ranking tasks and employs an adaptive probe search framework to select discriminative fingerprints for different task contexts. We evaluate RMF on 23 frontier LLMs across four real-world application domains. Compared with state-of-the-art baselines, RMF substantially improves model identification in both binary verification and multiclass identification settings, while introducing no statistically significant degradation in task performance and negligible additional token overhead.

1 INTRODUCTION

Third-party LLM aggregation platforms provide unified APIs for accessing multiple models, handling billing, usage, and request routing on behalf of clients (Lin et al., 2026). However, their black-box backend prevents clients from verifying which model actually serves each request. This allows an intermediary to substitute a cheaper or weaker model for the advertised one while charging the corresponding price (Cai et al., 2025; Lin et al., 2026). Such model substitution creates a practical API-fraud risk, whereby clients may pay for capabilities they do not actually receive and incur performance degradation or reliability losses in applications that depend on the advertised model.

To detect such API fraud, existing work commonly relies on black-box model fingerprinting, which identifies the backend model from its responses to carefully designed queries (Pasquini et al., 2025; Leshin et al., 2026). In general, these methods construct discriminative fingerprint queries to elicit model-specific behaviors and then infer model identity from textual, statistical, or representation-level response features (Iourovitski et al., 2024; Bai et al., 2025; Hu et al., 2026; Pasquini et al., 2025; Leshin et al., 2026). Existing approaches, however, predominantly follow a pre-deployment verification paradigm: users first issue dedicated fingerprint queries to verify the model identity and only then submit the actual task in a separate API request (Pasquini et al., 2025; Bai et al., 2025). Such one-time verification provides no assurance that subsequent task requests are served by the same model, since the backend model can be changed after verification or selectively substituted for non-fingerprint requests (Cai et al., 2025; Kurian et al., 2025). Therefore, reliable protection against model substitution requires model identity to be verified at runtime during actual user tasks.

Designing runtime fingerprints that can be integrated with real-world tasks introduces two key challenges. First, a fingerprint must capture model-specific behavioral signals while remaining robust across diverse task contexts. Although adversarial prompts used in prior work can elicit discriminative model behaviors, their effectiveness may degrade substantially when embedded in real-world inputs, where task context can interfere with the intended fingerprint signal. Second, integrating fingerprint probes into actual tasks should preserve task performance while introducing minimal computation overhead in terms of additional input and output tokens, to avoid unnecessary inference costs. To address these challenges, we develop RMF, a runtime model fingerprinting method

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.

that embeds fingerprint probes directly into real-world task prompts to identify the serving model during task execution. The key idea of RMF is to exploit model-specific preferences over carefully designed ranking tasks, such as ranking rainbow colors, as behavioral fingerprints. Because such ranking tasks induce a large space of possible outputs with high combinatorial diversity, they can encode discriminative model behaviors with only a small number of additional tokens. To further improve robustness across heterogeneous task contexts, we introduce an adaptive fingerprint-probe search framework that selects the most discriminative probe for each task query.

We evaluate RMF on four common real-world application scenarios, including question answering, mathematical reasoning, and code generation, covering 23 frontier LLMs from eight model families, such as GPT-5.6-Terra, Claude-Sonnet 5, Qwen3.7-Plus, and DeepSeek-V4-Flash. Experimental results show that, compared with state-of-the-art baselines, RMF substantially improves model identification performance in both binary verification (i.e., whether the serving model is the claimed target model) and multiclass identification (i.e., which model is actually serving the request). Meanwhile, embedding RMF fingerprints has no statistically significant impact on task performance ($p = 0.4307 > 0.05$) and incurs only negligible additional token overhead. Further experiments demonstrate that RMF generalizes well and remains robust under a range of challenging real-world conditions. Our contributions are summarized as follows:

- We are the first to study runtime fingerprinting for API model substitution, where model identity is verified jointly with actual user tasks rather than via pre-deployment verification.
- We propose a runtime fingerprinting framework that integrates fingerprint representation with adaptive probe search, enabling reliable identification across diverse task contexts.
- Experiments demonstrate that RMF achieves strong model identification performance with negligible impact on task accuracy and token cost.

2 Related Work

Black-box fingerprinting of language models. Black-box model fingerprinting infers the model behind an API from observable responses without access to model weights or internal activations. Existing methods typically construct inputs that elicit or amplify model-specific behavior, and then infer model identity from textual or statistical features, response representations, output distributions, or other behavioral patterns. These inputs include carefully designed, adversarial, sensitivity-maximizing, and prompt-injection-based queries (Pasquini et al., 2025; Leshin et al., 2026; Liu et al., 2026; Iourovitski et al., 2024; Bai et al., 2025; Hu et al., 2026). LLMmap aggregates response features from designed queries to identify model versions (Pasquini et al., 2025), READER accumulates evidence in a frozen proxy model's activation space (Liu et al., 2026), and Hide and Seek and LLMPrint use evolutionary search or prompt injection to elicit model-specific preferences (Iourovitski et al., 2024; Hu et al., 2026). Despite these differences, existing methods generally follow a pre-deployment workflow: users first identify the responding model with fingerprint queries and then submit the actual task in a separate request. A gateway can switch models during operation or recognize fingerprint queries, return the advertised model's responses for them, and route the actual task to a cheaper substitute (Cai et al., 2025; Kurian et al., 2025). RMF instead couples compact ranking probes with the real task in a single API request, so identity evidence is generated under the same routing decision as the task answer.

3 Method

3.1 Motivation

Our goal is to design a runtime model fingerprinting method that can be embedded in real API requests alongside application tasks. This imposes two practical requirements on probe design. First, probes should maintain cross-task stability, retaining model-specific behavioral signals across different application contexts. Adversarial probes often rely on carefully optimized inputs whose effects are sensitive to the surrounding context. Combining them with application tasks may therefore weaken or distort the signals used for fingerprinting. We instead use underdetermined tasks, which admit multiple plausible responses without a unique correct answer. Such tasks allow differences in model preferences to emerge, providing behavioral signals that can be used to distinguish models.

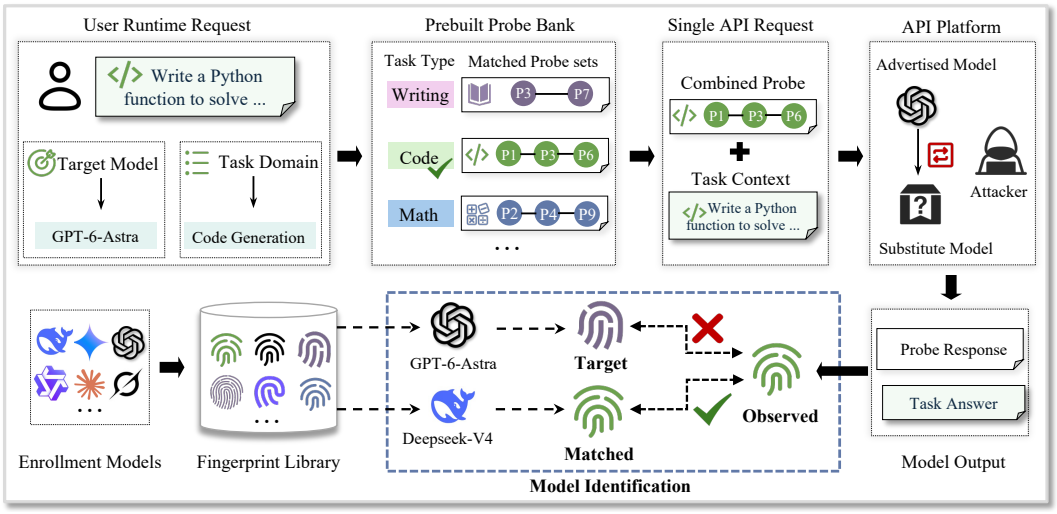

Figure 1: Framework of RMF for runtime model identification during a user request.

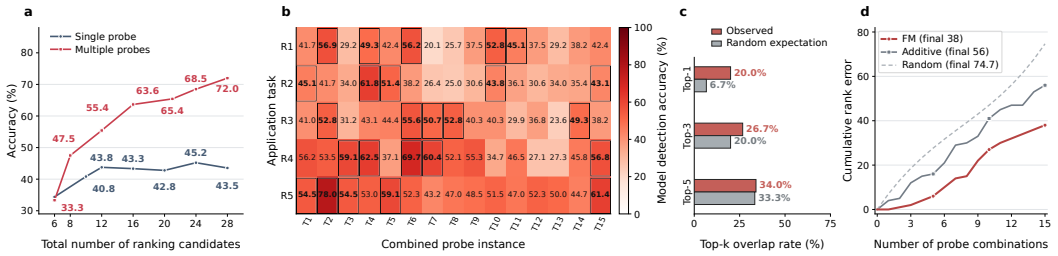

Figure 2: Method motivation. (a) Model identification accuracy under equal probe-token budgets. Multiple short ranking probes outperform a single long probe, motivating multi-probe fingerprints. (b) Identification accuracy when individually combining 16 fingerprint probes with five real-world tasks. (c) Overlap among the top-k probes across different tasks. The consistently low overlap suggests that combining multiple probes can improve identification robustness across diverse tasks. (d) Estimation error of different probe-search methods. Our method achieves the optimal performance.

Second, probes should be token-efficient, using few input and output tokens to limit inference overhead and minimize interference with the application task. We adopt ranking tasks as our probe format: each probe presents a small set of candidates and asks the model to return their ordering. This compact format captures multiple relative preferences without requiring lengthy responses. With n candidates, a probe admits $n!$ possible rankings, allowing a small candidate set to express rich patterns of relative preferences and provide fine-grained behavioral signals for cross-model discrimination. Our preliminary observations in Figure 2(a) shows that ranking preferences exhibit within-model consistency and cross-model differences across task contexts, supporting their use as compact, model-specific fingerprints.

3.2 Fingerprint Construction

Fingerprint Probe Generation. We first use a language model to generate a pool of underdetermined ranking probes covering preference judgments, prioritization, and decision trade-offs. Each probe consists of a ranking task over a candidate set and asks the model to order several plausible alternatives. For example, a probe may ask the model to rank red, blue, green, and yellow by preference. Because no ordering is uniquely correct, different models may exhibit different preferences. To represent each returned ranking in the fingerprint, we decompose it into pairwise comparisons. For probe t, let $\mathcal{C}_t$ denote its candidate set and π_t the returned ranking. We fix one orientation (a, b) for each candidate pair and encode its outcome as $z_{t,(a,b)} = 1$ if π_t ranks a above b, and $z_{t,(a,b)} = 0$

otherwise. Thus, a probe with n candidates produces $\binom{n}{2}$ binary features in each response. Across repeated responses, the mean of each feature estimates the probability that a is ranked above b.

Fingerprint Probe Filtering. Using these pairwise preference units, we evaluate each candidate probe on representative models for within-model consistency and between-model separability. The former measures whether repeated responses from the same model produce consistent preferences, while the latter measures whether the preferences differ across models. We discard probes that are unstable within a model or insufficiently discriminative across models, retaining probes that provide stable and model-specific behavioral signals.

Fingerprint Probe Combination. The second design requirement in Section 3.1 emphasizes minimizing probe token overhead. This raises a budget-allocation question: under comparable token budgets, should we use one ranking probe with more candidates or combine several shorter probes? Figure 2(a) empirically compares these alternatives. Increasing the candidate count within a single probe yields limited gains, whereas combining multiple probes achieves higher identification accuracy. One possible explanation is that additional candidates within one scenario elicit overlapping preferences, whereas different probes capture complementary aspects of model behavior. We therefore concatenate the preference vectors of multiple probes into a fingerprint $z(B) \in \{0,1\}^{d(B)}$, where B denotes the probe bundle and $d(B) = \sum_{t \in B} \binom{|\mathcal{C}_t|}{2}$. However, probe effectiveness also depends on the application context. Figure 2(b) shows that the strongest probes vary across application tasks, while Figure 2(c) reveals limited overlap among their top-k probe sets. These observations raise a further question: how can we select an effective and complementary probe bundle for a given task? This motivates the FM-guided search introduced in Section 3.3.

3.3 Adaptive Fingerprint Search Framework

Because the candidate probes form a combinatorial bundle space, exhaustively evaluating the interactions among different probes is impractical. We therefore use a factorization machine, a standard feature-interaction model, to learn low-dimensional latent factors from observed bundles and estimate utility across the high-dimensional combination space. Let P denote the pool size, and represent a bundle by $x(B) \in \{0,1\}^P$, where $x_i(B) = 1$ if probe i is included. We measure the utility of a bundle as

$$y(B) = \lambda \, \mathrm{Acc}_{\mathrm{strict}}(B) + (1-\lambda) \, \mathrm{Acc}_{\mathrm{family}}(B). \tag{1}$$

where λ controls the relative emphasis on exact model identification and family-level identification. We evaluate candidate bundles on representative models with repeated responses and separate training and test splits. The initial design includes singleton, balanced pair, and balanced triple bundles to provide observations across the candidate pool.

Given the limited number of bundle evaluations, we fit a rank-r factorization machine to predict utility for unobserved bundles:

$$\widehat{y}(x) = b + w^\top x + \tfrac{1}{2} \sum_{f=1}^{r} [(\sum_i v_{if} x_i)^2 - \sum_i v_{if}^2 x_i^2]. \tag{2}$$

The linear term captures the utility of individual probes, whereas the low-rank term captures interactions between probes. We use an ensemble of bootstrap FM fits with rank r. At each iteration, unobserved bundles with the largest ensemble mean are selected for exploitation, while a bundle with the largest ensemble standard deviation is selected for exploration. We search over the permitted bundle sizes and choose the highest-utility observed bundle, preferring fewer probes in a tie. The selected bundle is then fixed for all downstream enrollment and online evaluations.

In the retrospective combination audit shown in Fig. 2(d), FM-guided prediction achieves a lower cumulative absolute ranking error than both the additive and random baselines, reaching 38 compared with 56 and 74.7 at the final point. This result indicates that modeling probe interactions improves utility prediction and supports more effective bundle selection.

3.4 Runtime Model Identification

Before deployment, we construct a reference fingerprint for each enrolled model using the selected probe bundle. Let E_m denote the collection of binary fingerprints obtained from model m on source

tasks. We estimate the preference probability for each feature as its empirical frequency, $\widehat{p}_{m,j} = \frac{1}{|E_m|}\sum_{z\in E_m} z_j$. The resulting probability vector serves as the reference fingerprint for model m.

At runtime, the selected probes are appended to the application task in a single API request. We extract the returned rankings and apply the pairwise encoding to obtain the observed fingerprint z. We compare z with each reference center using the matching cost:

$$\ell(z, m) = -\frac{1}{d(B)} \sum_{j=1}^{d(B)} \left[z_j \log \widehat{p}_{m,j} + (1 - z_j) \log(1 - \widehat{p}_{m,j}) \right]. \tag{3}$$

A smaller value of $\ell(z, m)$ indicates a closer match to model m's preference profile.

In our evaluation, we consider three identification settings. In binary verification, when a client only needs to verify whether the runtime model is a m_0, we calibrate a target-specific threshold τ_{m_0} on held-out source-task fingerprints. The request is classified as originating from m_0 when $\ell(z, m_0) \leq \tau_{m_0}$, and as a non-target request otherwise.In closed-set multiclass identification, when the serving model is assumed to belong to a candidate set $\mathcal{M}$, we select the model with the smallest matching cost, $\widehat{m} = \arg\min_{m\in\mathcal{M}} \ell(z, m)$. In open-set identification, the gateway may route requests to a model outside $\mathcal{M}$. We therefore use model-specific thresholds calibrated on held-out source-task fingerprints. After selecting the best-matching model $\widehat{m}$, we assign the request to $\widehat{m}$ if $\ell(z, \widehat{m}) \leq \tau_{\widehat{m}}$; otherwise, we classify it as unknown. The probe bundle, fingerprint centers, and thresholds remain fixed during runtime inference, requiring neither the FM surrogate nor target-task labels.

4 Experiments

4.1 Experimental Setups

Models and datasets. We use PolicyQA (Ahmad et al., 2020), PubMedQA (Jin et al., 2019), MBPP+ (Liu et al., 2023), and MATH-500 (Lightman et al., 2024), covering question answering, code generation, and mathematical reasoning. We consider two multiclass settings. The *fine-grained* setting contains 20 models from GPT, Claude, Qwen, DeepSeek, and Gemini (Achiam et al., 2023; Anthropic, 2024; Yang et al., 2024; Liu et al., 2024; Team et al., 2024). The *coarse-grained* setting contains one representative model from each of eight families: GPT, Claude, Qwen, DeepSeek, Gemini, Kimi, MiniMax, and GLM (Kimi Team, 2025; Team et al., 2025; Glm et al., 2024).

Baselines. We compare LLMmap, FLIPS, and GateScope, three SOTA model fingerprinting methods. LLMmap uses active queries for model identification (Pasquini et al., 2025). FLIPS uses pseudo-random binary sequences for instance-level fingerprinting (Richardeau et al., 2026). GateScope audits API gateways through behavioral consistency, routing, billing, and latency signals (Lin et al., 2026). All methods use the same task-coupled protocol: fingerprinting probes are concatenated with the same real task and submitted in a single API request. Identification uses only probe responses, whereas the real-task answer is used only to evaluate task performance.

Evaluation metrics. For multiclass identification, we report closed-set top-1 accuracy, open-set correct classification rate (CCR) for known models, open-set unknown rejection rate (URR), and open-set accuracy. For binary identification, we report accuracy and the area under the receiver operating characteristic curve (AUROC). We also compare task performance with and without fingerprint probes. Multiclass results are reported as means and standard deviations over random splits.

Settings of RMF. In RMF, an FM model is trained to estimate fingerprint effectiveness using LLM-synthesized source-task data constructed only from task descriptions, without accessing any target-benchmark examples. For example, when evaluating on MBPP+, we use independently synthesized coding tasks for FM training, ensuring benchmark-independent probe search and generalization to unseen applications. The candidate probe pool contains $P = 20$ probes. FM uses objective weight $\lambda = 0.7$, rank $r = 3$, and an ensemble of 30 bootstrap models.

Table 1: Coarse- and fine-grained model identification across four real-task domains. RMF outperforms the strongest baseline in both closed-set and open-set evaluation.

Fine-grained setting (Five families, twenty individual models)						
Domain	Method	Closed-set Acc.	Open-set CCR	Open-set URR	Open-set Acc.	Average
PolicyQA	LLMmap	68.07 ± 4.94	71.57 ± 1.55	17.97 ± 6.56	60.85 ± 1.63	54.62 ± 3.67
	FLIPS	45.57 ± 1.58	47.59 ± 1.42	9.62 ± 2.03	40.00 ± 1.13	35.70 ± 1.54
	GateScope	59.55 ± 2.48	62.00 ± 1.23	12.50 ± 2.72	52.10 ± 1.32	46.54 ± 1.94
	RMF	$\mathbf{79.75} \pm 2.48$	$\mathbf{72.97} \pm 2.89$	$\mathbf{57.98} \pm 4.59$	$\mathbf{69.97} \pm 1.82$	$\mathbf{70.17} \pm 2.95$
MBPP+	LLMmap	62.80 ± 4.82	68.18 ± 2.60	14.83 ± 6.14	57.51 ± 2.78	50.83 ± 4.09
	FLIPS	43.60 ± 2.19	45.80 ± 2.04	9.15 ± 1.72	38.47 ± 1.61	34.26 ± 1.89
	GateScope	61.63 ± 1.45	63.96 ± 1.55	12.38 ± 1.32	53.65 ± 1.31	47.91 ± 1.41
	RMF	$\mathbf{74.53} \pm 1.06$	$\mathbf{69.33} \pm 2.26$	$\mathbf{49.23} \pm 6.52$	$\mathbf{65.31} \pm 1.08$	$\mathbf{64.60} \pm 2.73$
MATH-500	LLMmap	63.27 ± 5.39	67.04 ± 2.76	17.85 ± 6.26	57.20 ± 3.19	51.34 ± 4.40
	FLIPS	41.95 ± 1.98	44.26 ± 1.52	11.05 ± 2.62	37.62 ± 1.08	33.72 ± 1.80
	GateScope	56.37 ± 1.42	58.95 ± 1.17	13.02 ± 2.00	49.76 ± 1.00	44.53 ± 1.40
	RMF	$\mathbf{74.35} \pm 1.46$	$\mathbf{67.00} \pm 2.78$	$\mathbf{58.90} \pm 2.18$	$\mathbf{65.38} \pm 1.98$	$\mathbf{66.41} \pm 2.10$
PubMedQA	LLMmap	61.37 ± 6.15	64.55 ± 2.24	15.32 ± 6.69	54.71 ± 2.41	48.99 ± 4.37
	FLIPS	43.45 ± 2.88	45.00 ± 2.15	9.70 ± 2.27	37.94 ± 1.83	34.02 ± 2.28
	GateScope	59.03 ± 2.48	60.51 ± 1.60	11.57 ± 1.89	50.72 ± 1.44	45.46 ± 1.85
	RMF	$\mathbf{75.48} \pm 1.13$	$\mathbf{65.73} \pm 1.65$	$\mathbf{52.93} \pm 3.72$	$\mathbf{63.17} \pm 1.31$	$\mathbf{64.33} \pm 1.95$
Coarse-grained setting (Eight families, one representative model each)						
Domain	Method	Closed-set Acc.	Open-set CCR	Open-set URR	Open-set Acc.	Average
PolicyQA	LLMmap	85.92 ± 4.15	87.90 ± 2.44	32.54 ± 12.41	74.06 ± 4.46	70.11 ± 5.87
	FLIPS	65.25 ± 2.37	67.04 ± 1.83	15.25 ± 5.85	54.09 ± 2.35	50.41 ± 3.10
	GateScope	69.71 ± 3.29	69.94 ± 1.73	9.83 ± 2.81	54.92 ± 1.42	51.10 ± 2.31
	RMF	$\mathbf{97.71} \pm 0.86$	$\mathbf{90.21} \pm 3.26$	$\mathbf{74.58} \pm 12.41$	$\mathbf{86.30} \pm 2.33$	$\mathbf{87.20} \pm 4.72$
MBPP+	LLMmap	70.92 ± 8.08	72.58 ± 3.57	18.75 ± 9.48	59.13 ± 3.44	55.35 ± 6.14
	FLIPS	65.12 ± 3.40	66.86 ± 3.32	13.67 ± 4.83	53.56 ± 2.95	49.80 ± 3.63
	GateScope	72.58 ± 1.80	75.04 ± 0.83	18.38 ± 3.39	60.88 ± 1.27	56.72 ± 1.82
	RMF	$\mathbf{82.71} \pm 1.41$	$\mathbf{68.51} \pm 2.16$	$\mathbf{66.08} \pm 5.51$	$\mathbf{67.91} \pm 1.71$	$\mathbf{71.30} \pm 2.70$
MATH-500	LLMmap	84.83 ± 3.49	85.61 ± 1.42	29.58 ± 14.31	71.60 ± 3.95	67.91 ± 5.79
	FLIPS	63.33 ± 2.08	65.26 ± 0.99	12.62 ± 4.40	52.10 ± 1.44	48.33 ± 2.23
	GateScope	72.83 ± 2.28	72.85 ± 2.07	19.21 ± 3.78	59.44 ± 1.82	56.08 ± 2.49
	RMF	$\mathbf{96.08} \pm 1.13$	$\mathbf{85.15} \pm 3.53$	$\mathbf{73.79} \pm 7.10$	$\mathbf{82.31} \pm 3.88$	$\mathbf{84.33} \pm 3.91$
PubMedQA	LLMmap	76.33 ± 6.52	76.10 ± 4.52	28.42 ± 14.39	64.18 ± 6.15	61.26 ± 7.90
	FLIPS	61.46 ± 3.35	62.62 ± 3.28	15.92 ± 4.50	50.95 ± 2.27	47.74 ± 3.35
	GateScope	72.46 ± 2.51	70.49 ± 1.36	19.46 ± 5.61	57.73 ± 1.47	55.04 ± 2.74
	RMF	$\mathbf{97.83} \pm 0.78$	$\mathbf{82.97} \pm 4.49$	$\mathbf{57.83} \pm 9.31$	$\mathbf{76.69} \pm 4.46$	$\mathbf{78.83} \pm 4.76$

4.2 Performance on Multiclass Model Identification

We evaluate RMF in two multiclass identification scenarios across four real-task domains. Fine-grained identification treats each individual model from five model families as a separate class, whereas coarse-grained identification uses one representative model from each of eight families to assess family-level recognition. For each domain, we construct a source task from the task description, collect enrollment and calibration responses without using target-benchmark examples, and evaluate identification on held-out benchmark samples. Table 1 summarizes the resulting closed-set and open-set performance. RMF consistently outperforms the baselines across domains. Averaged across domains, it improves closed-set and open-set accuracy by 12.15 and 8.39 percentage points in the fine-grained setting, and by 14.08 and 11.06 points in the coarse-grained setting. The consistent gains across both identification scenarios and task domains support the portability of the runtime fingerprint representation.

4.3 Performance on Binary Model Identification

For binary identification, we construct a one-vs-rest detector for each target model, treating the target as the positive class and all remaining endpoints as the negative class. Each detector uses source-task responses for enrollment and is evaluated on held-out responses from the four real-task domains, resulting in 16 target–task pairs. Figure 3 reports the corresponding ROC curves. RMF achieves an

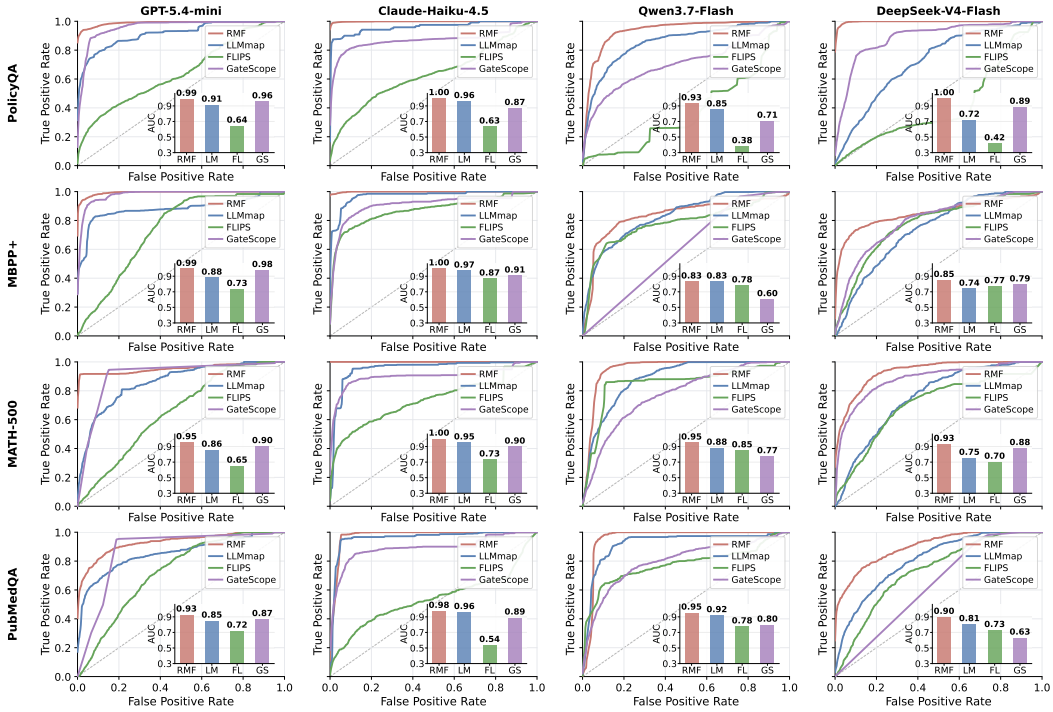

Figure 3: Binary model identification across four real-task domains.

AUROC of at least 0.83 in all 16 target–task pairs, with a mean AUROC of 0.949 compared with 0.865 for the strongest baseline. These results demonstrate strong binary identification performance across target models and application tasks.

4.4 Task Performance Preservation and Token Efficiency

Next, we demonstrate that our method is near-lossless in both task accuracy and computational overhead. First, we evaluate task performance with and without runtime fingerprint probes. Fig. 4 reports the task performance and the bootstrap uncertainty. The results show no statistically significant difference in task accuracy between the two conditions ($p = 0.4307 > 0.05$), indicating that runtime fingerprinting introduces negligible task interference. Furthermore, to quantify the additional cost, Table 2 reports the extra input and output tokens introduced by different fingerprinting methods, with monetary costs estimated using GPT-5.6-Luna pricing. RMF incurs the lowest overall token overhead and cost among the compared methods. This overhead is also inexpensive in practice, amounting to $1.89 for 10,000 requests.

Table 2: Token overhead and cost per request (USD $\times 10^{-4}$, gpt-5.6-Luna pricing).

Method	Input	Output	Sum	Cost
LLMmap	668.3	110.1	778.4	$2.60
FLIPS	117.4	667.1	784.4	$8.05
GateScope	608.2	844.4	1452.6	$11.08
RMF	296.8	111.4	**408.1**	**$1.89**

4.5 Generalization Evaluation

We compare two training–testing configurations. In the real-task data-aware fingerprint setting, fingerprint centers and calibration are learned from the target benchmark task and evaluated on held-out samples from the same task. In the real-task data-agnostic fingerprint setting, they are learned only from a domain-matched constructed task and evaluated on held-out benchmark tasks. This comparison tests whether constructed source tasks can replace target-benchmark data during enrollment. Binary AUROC is evaluated separately using the one-vs-rest protocol described above. The real-task data-agnostic setting remains effective on unseen benchmark tasks. Although its closed-set accuracy is lower than that of the real-task data-aware setting, the gap remains bounded across domains, and

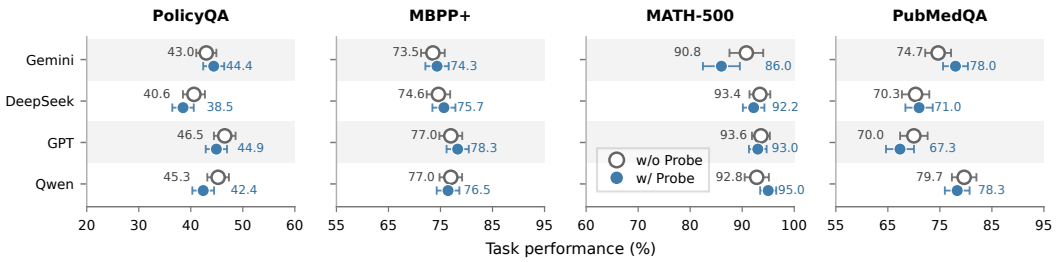

Figure 4: Performance of real-world tasks with and without runtime fingerprint probes. Probe integration introduces no significant degradation in task accuracy.

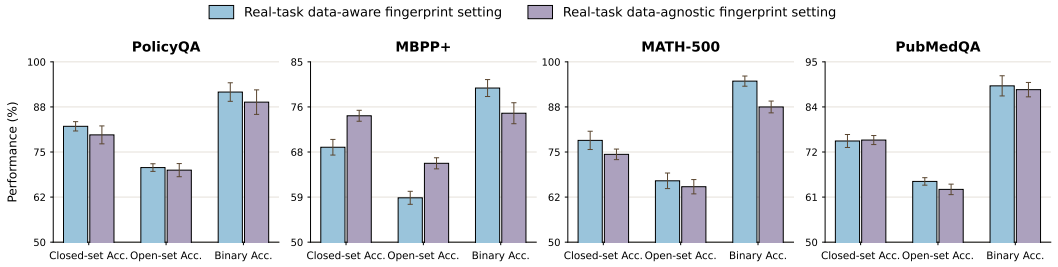

Figure 5: Generalization of adaptive fingerprint search. Data-aware settings use i.i.d. samples from real-world tasks for probe search, whereas data-agnostic settings rely only on LLM-generated data. The comparable performance of the two settings demonstrates the generalization of our method.

Table 3: Robustness to sampling parameters during model inference. RMF maintains strong performance even when the settings used for fingerprint construction and task execution are mismatched.

	Multi-Class Identification				Binary Identification	
Condition	Closed-set Acc.	Open-set CCR	Open-set URR	Open-set Acc.	Balanced Acc.	AUROC
Matching	79.75 ±2.48	72.97 ±2.89	57.98 ±4.59	69.97 ±1.82	88.82 ±4.20	0.9795 ±0.0078
Mismatching	78.47 ±0.71	65.39 ±2.03	70.02 ±3.37	66.31 ±1.74	85.18 ±5.47	0.9442 ±0.0146

open-set accuracy stays above chance in all reported cases. These results support constructed source tasks as a practical enrollment option when target-benchmark data are unavailable.

4.6 Fingerprint Visualization

To examine the structure of the learned fingerprints across task domains, we visualize 50 fingerprints from six models using principal-component analysis (PCA). Figure 6 shows the projections for PolicyQA, MBPP+, MATH-500, and PubMedQA. Across all four domains, the fingerprints form distinct model-specific regions. The visualization separates models from different families, such as GPT and DeepSeek, while also revealing smaller differences between closely related models, such as GPT-Terra and GPT-Luna. The recurrence of these patterns across domains is consistent with the fingerprint representation preserving both cross-family and fine-grained model information. This visualization is descriptive; all identification metrics are computed in the original fingerprint space rather than in the two-dimensional projection.

4.7 Robustness Under Practical Inference Settings

We evaluate robustness to differences in sampling parameters between fingerprint construction and runtime testing. In the Param-Matched condition, data for both stages are generated using the same temperature and top-p settings. In the Param-Mismatched condition, these settings differ between the two stages. Table 3 shows that parameter mismatch reduces multiclass closed-set and open-set accuracy by 1.28 and 3.66 percentage points, respectively. Binary AUROC remains high at 0.9442,

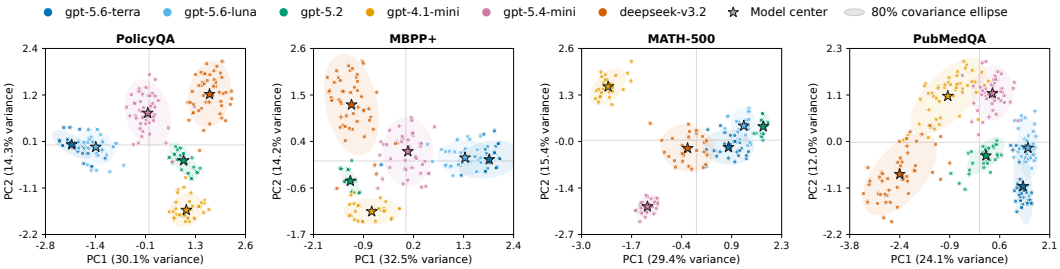

Figure 6: PCA visualization of fingerprint representation across models. Reuslts show that RMF can not only identify models across families (e.g., gpt v.s., deepseek), but also identify models with minor differences (e.g., gpt-5.6-terra and gpt-5.6-luna).

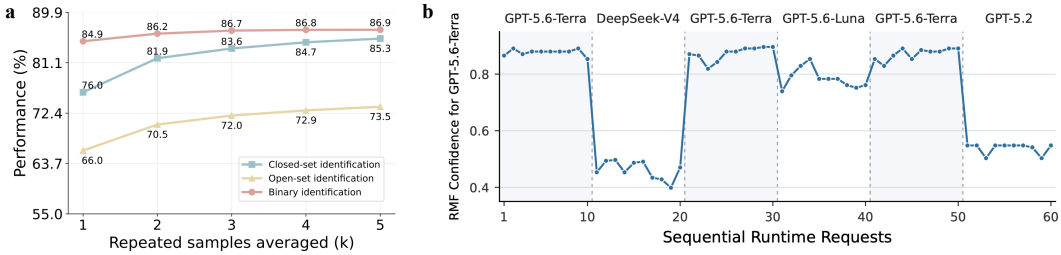

Figure 7: (a) Identification performance with varying numbers of task queries averaged across four benchmarks. Using more nearby queries enhances identification. (b) Runtime model identification under dynamic substitution, where RMF promptly captures changes in the serving model.

compared with 0.9795 under matched settings. These results support the robustness of RMF to the tested differences in sampling parameters.

We further examine whether averaging fingerprint scores improves identification stability. For each setting, we average scores over k samples and repeat sampling 100 times; positive and negative samples are averaged separately for binary evaluation. As shown in Figure 7 (a), increasing k consistently improves identification across all four domains, with the largest gain from $k = 1$ to $k = 2$. Closed-set accuracy increases from about 74–80% at $k = 1$ to 83–87% at $k = 5$, while open-set and binary identification show similar trends with diminishing returns. Overall, score averaging improves identification at the cost of additional samples.

We further evaluate whether RMF can detect dynamic model substitution during continuous runtime requests. In this experiment, the backend model is deliberately switched across sequential requests, while RMF continuously tracks the identification confidence for GPT-5.6-Terra. As shown in Figure 7(b), the confidence remains high when GPT-5.6-Terra serves the requests and drops sharply when it is replaced by other models, recovering once GPT-5.6-Terra is restored. Notably, RMF remains sensitive even when the substitute is a closely related model from the same GPT family, such as GPT-5.2 or GPT-5.6-Luna. These clear transitions demonstrate the ability of RMF to detect fine-grained model substitutions during runtime execution.

5 Conclusion

We presented RMF, a runtime model fingerprinting method for API fraud detection that integrates fingerprint probes with real-world tasks in a single API request. RMF represents model-specific preferences on underdetermined ranking tasks as compact behavioral fingerprints and adaptively searches for effective probe combinations across task contexts. Experiments on 23 models from eight families show substantial identification improvements in both multiclass and binary identification over existing baselines. Results also demonstrate that RMF preserves task accuracy and introduces negligible token overhead. Further results indicates strong generalization to unseen tasks and robustness across practical inference settings.

AI USE STATEMENT

In this work, we used generative AI tools to assist with discussions of the research methodology and experimental workflow, code development and checking, literature search and organization, Chinese–English translation, and language editing. All AI-assisted code was manually reviewed and validated through the corresponding experiments. The authors also verified the relevant literature, technical descriptions, and experimental data. Generative AI was not used to determine experimental labels or adjudicate the final results. The authors take full responsibility for all content, experimental results, and conclusions in this paper.

ETHICS STATEMENT

This work aims to improve the transparency and auditability of third-party large language model API services. It does not involve human participants or collect or process users' personal information. The experiments use public benchmark datasets and access model APIs through standard user-facing interfaces. Any released research materials will comply with the applicable dataset licenses and service terms and will exclude API keys and other sensitive configuration information. Because model fingerprinting may have dual-use implications, the proposed method should be used only for authorized model verification, routing audits, and service-quality assessment, rather than to circumvent access controls or conduct unauthorized system probing.

REPRODUCIBILITY STATEMENT

The Methods section describes the underdetermined ranking probes, pairwise preference encoding, factorization-machine-guided probe search, fingerprint center construction, and online inference procedure. The Experiments section specifies the benchmarks, model sets, data splits, baselines, and evaluation protocols. We provide an anonymous reproducibility package containing the code, experimental configurations, complete probe library, source-task templates, random seeds, and processed results used to generate the tables and figures. The package is available at `https://anonymous.4open.science/r/RMF-839B`.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

Wasi Ahmad, Jianfeng Chi, Yuan Tian, and Kai-Wei Chang. PolicyQA: A reading comprehension dataset for privacy policies. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 743–749. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.66. URL https://aclanthology.org/2020.findings-emnlp.66/.

Anthropic. Introducing the next generation of Claude, 2024. URL https://www.anthropic.com/news/claude-3-family.

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. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.546. URL https://aclanthology.org/2025.findings-acl.546/.

Will Cai, Tianneng Shi, Xuandong Zhao, and Dawn Song. Are you getting what you pay for? auditing model substitution in LLM APIs. *arXiv preprint arXiv:2504.04715*, 2025. doi: 10.48550/arXiv.2504.04715. URL https://arxiv.org/abs/2504.04715.

Team Glm, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. ChatGLM: A family of large language models from GLM-130b to GLM-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024. URL https://arxiv.org/abs/2406.12793.

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. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.541. URL https://aclanthology.org/2026.acl-long.541/.

Dmitri Iourovitski, Sanat Sharma, and Rakshak Talwar. Hide and seek: Fingerprinting large language models with evolutionary learning. *arXiv preprint arXiv:2408.02871*, 2024. URL https://arxiv.org/abs/2408.02871.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2567–2577. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1259. URL https://aclanthology.org/D19-1259/.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025. URL https://arxiv.org/abs/2501.12599.

Kevin Kurian, Ethan Holland, and Sean Oesch. Attacks & defenses against LLM fingerprinting. In *2025 Annual Computer Security Applications Conference Workshops (ACSAC Workshops)*, pp. 480–490. IEEE, 2025. doi: 10.1109/ACSACW69556.2025.00057.

Jonah Leshin, Manish Shah, Ian Timmis, and Daniel Kang. Behavioral fingerprints for LLM endpoint stability and identity. In *Proceedings of the ACM Conference on AI and Agentic Systems (CAIS '26)*, pp. 1327–1331, 2026. doi: 10.1145/3786335.3813194. URL https://arxiv.org/abs/2603.19022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Guanjie Lin, Yinxin Wan, Shichao Pei, Ting Xu, Kuai Xu, and Guoliang Xue. Behavioral consistency and transparency analysis on large language model API gateways. In *Proceedings of the 2026 ACM Internet Measurement Conference (IMC '26)*, 2026. doi: 10.1145/3777912.3809156. URL https://arxiv.org/abs/2604.21083.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. URL https://arxiv.org/abs/2412.19437.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572. Curran Associates, Inc., 2023. doi: 10.52202/075280-0943. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/43e9d647ccd3e4b7b5baab53f0368686-Paper-Conference.pdf.

Jiaxu Liu, Sunnan Mu, Dong Huang, Liuyin Wang, Jing Shao, and Jie Zhang. READER: Robust evidence-based authorship decoding via extracted representations. *arXiv preprint arXiv:2606.10794*, 2026. URL https://arxiv.org/abs/2606.10794v1.

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. USENIX Association, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/pasquini.

Gurvan Richardeau, Gohar Dashyan, Erwan Le Merrer, and Gilles Tredan. FLIPS: Instance-fingerprinting for LLMs via pseudo-random sequences. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306. PMLR, 2026. URL https://arxiv.org/abs/2606.03330.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. URL https://arxiv.org/abs/2403.05530.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025. URL https://arxiv.org/abs/2501.08313.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. URL https://arxiv.org/abs/2407.10671.

A Supplementary Material

A.1 Additional Experimental Details

At each iteration of FM-guided probe search, we select the three unobserved bundles with the highest ensemble mean for exploitation and one bundle with the highest ensemble uncertainty for exploration. Open-set acceptance uses a calibrated 95% acceptance region.

In multiclass experiments, each model provides 20 source-task responses. We use 10 responses for enrollment and 10 for open-set calibration, and repeat the random split ten times. Fine-grained and coarse-grained open-set evaluation uses 16 known and 4 unknown models, and 6 known and 2 unknown models, respectively. For binary identification, each target model uses 70 source-task responses for enrollment and 120 held-out test responses.