WHO MADE THIS LLM? BLACK-BOX FAMILY PROVENANCE FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Frontier large language model (LLM) families such as Llama, Qwen and Mistral are valuable assets, and the ease of producing derivative models through fine-tuning, distillation or merging exposes them to theft, unauthorized reuse and false ownership claims. Identifying the family a deployed LLM descends from is therefore central to ownership auditing, but existing solutions fall short under realistic conditions. Watermarking requires proactive deployment and cannot be retrofitted to already-released models, white-box fingerprinting requires parameter access that API-served models do not expose, and extracting reliable family-level signals from black-box responses remains challenging under model modifications. We propose FamilyTrace, a black-box fingerprint for family-level LLM provenance that supports both closed-set attribution among known families and open-set rejection of unknown families. Our design starts from two observations: much of the knowledge a model learns in pre-training is encoded in its feed-forward layers, which hold most of its parameters, and typical post-training changes these weights only slightly. We therefore expect the models of a family to share similar word associations, and FamilyTrace captures them by asking a model to rank candidate words by their relation to an ambiguous word, first with little context and then with progressively more context. We evaluate FamilyTrace on two benchmarks covering six LLM families, a non-adversarial benchmark of 55 LLMs and an anti-provenance benchmark of 50 LLMs. FamilyTrace achieves 96.3% closed-set and 87.9% open-set accuracy on the non-adversarial benchmark, and 85.4% closed-set and 79.5% open-set accuracy on the anti-provenance benchmark, outperforming seven black-box fingerprinting baselines.

1 INTRODUCTION

Large language models (LLMs) have rapidly become a foundational technology, powering a growing ecosystem of products from coding assistants to scientific copilots (Achiam et al., 2023; Comanici et al., 2025; Liu et al., 2024; Kumar, 2024). Producing a frontier LLM family requires substantial compute, curated data, alignment effort and deployment infrastructure (Raza et al., 2025; Hadi et al., 2023; Jiang et al., 2026), making such families economically and strategically valuable assets. Once a family is released, however, descendant models can be obtained cheaply through prompt-based adaptation (Xu et al., 2024c; Giray, 2023), parameter-efficient (Dettmers et al., 2023; Hu et al., 2022) or full fine-tuning (Lv et al., 2024), distillation (Xu et al., 2024b) and merging (Yang et al., 2026; Wortsman et al., 2022), and may then be redistributed under repackaged identities (Shao et al., 2025a; Carlini et al., 2024; Yona et al., 2024). The exact source checkpoint is often obscured by these operations, yet the inherited architectural and behavioral lineage typically remains. The practical attribution question is therefore not which exact checkpoint produced a target model, but which family it descends from. We refer to this question as *family-level LLM provenance identification*.

Existing provenance techniques fall into two broad classes, both with structural limitations under realistic conditions. Watermarking methods proactively embed ownership signals before deployment (Kirchenbauer et al., 2023; Pan et al., 2024), which makes them inapplicable to the vast majority of already-released models, and the embedded signals can be weakened by paraphrasing, rewriting, removal attacks and trigger collisions (Liang et al., 2026). Fingerprinting methods, in contrast, recover ownership directly from the model itself. White-box fingerprints exploit invariants of internal parameters or activations and achieve strong attribution accuracy (Zeng et al., 2024; Zhang et al.,

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.

2024a; Yang & Wu, 2024), but they require parameter access that is usually unavailable when the target is served only through an API. This leaves black-box fingerprinting as the most practical regime for third-party auditing, and recent work has accordingly turned to text-level cues such as response style, query-response traces, knowledge boundaries and prefix-based behavioral triggers (Pasquini et al., 2025; Gao et al., 2024; Jin et al., 2024; Shao et al., 2025b).

We argue that current black-box methods are limited by what they choose to observe. They read what a model *says* under crafted prompts, but they do not target the associations between concepts that a model learns in pre-training. Surface-level signals such as stylistic patterns or trigger responses describe expression, not learned associations, and are therefore vulnerable to deployment-time changes such as system prompts, persona templates and decoding configurations. A more stable basis for identifying a family is how a model relates concepts to one another, namely its word association preferences and how it revises them as more context is added. This choice is based on two observations. First, feed-forward network (FFN) layers, which hold most of a Transformer's parameters, store a substantial part of the associative knowledge learned in pre-training (Geva et al., 2021; 2022; Meng et al., 2022). Second, typical post-training updates are small and low-rank relative to the pre-trained weights (Aghajanyan et al., 2021; Hu et al., 2022; Zeng et al., 2024) and mainly change style and format rather than knowledge (Lin et al., 2024). We therefore hypothesize that the models of a family largely share their word association preferences, and we study whether these preferences can be reliably identified under strict black-box access.

To measure these preferences, we use queries in which several answers are reasonable. Models rank candidate words by their relatedness to an ambiguous anchor word. For example, given *bank* in *She waited by the bank*, a model may rank *stream* and *shore* above *account* and *capital*, or the other way around, depending on its own associations. Since models also differ in how strongly they rely on prior knowledge when it conflicts with the context (Longpre et al., 2021; Xie et al., 2024), we present each anchor in sentences that add context step by step. The first ranking gives the default preference of the model, which we refer to as the *associative prior*, and the changes in the later rankings give its *contextual revision*. If our hypothesis holds, models of the same family should rank candidates similarly even after typical post-training, and queries whose answers are fixed by common knowledge should reveal less about the family.

Building on this design, we propose FamilyTrace, a black-box family-level provenance method that turns these rankings into a fingerprint. FamilyTrace makes three design choices. First, the candidates for each anchor are the other anchors of the list, so every word serves both as an anchor and as a candidate and the comparison between families does not depend on how extra candidate words are chosen. Second, each anchor is presented in four stages whose context becomes more specific, so the same queries yield both the associative prior and its contextual revision. Third, we keep the full rankings of each anchor across stages and compare them with the average rankings of each known family, anchor by anchor. Our contributions are summarized as follows:

- We propose a new view of black-box LLM provenance. Instead of fingerprinting how a model expresses itself, we hypothesize that the models of a family largely share their word association preferences, and we measure these preferences with a ranking task over ambiguous words in which several orderings are reasonable.
- We instantiate this view as FamilyTrace, a black-box provenance method that needs only the text responses of a model to 24 ranking queries. FamilyTrace compares the rankings of a target model with the average rankings of each known family, which allows it both to attribute the model to a known family and to reject models from unknown families. We show that this comparison has a simple interpretation: it measures how closely the target agrees with the members of a family, relative to how closely the members agree with each other.
- We construct two benchmarks covering six families, where FamilyTrace achieves 96.3% closed-set and 87.9% open-set accuracy on a non-adversarial benchmark of 55 LLMs, and 85.4% closed-set and 79.5% open-set accuracy on an anti-provenance benchmark of 50 LLMs, outperforming all seven black-box baselines while remaining stable across query-list variants. The results are also consistent with both predictions of our hypothesis. Among models fine-tuned from three base models on the same data, models that share a base model are more similar than models that share fine-tuning data. Replacing our queries with words whose order is fixed by common sense lowers accuracy, and replacing them with random words makes accuracy both lower and much less stable.

2 RELATED WORK

LLM ownership protection: LLM ownership protection is commonly studied through watermarking and fingerprinting (Shao et al., 2025a; Xu et al., 2024a). Watermarking embeds ownership signals into the model before release through generation-time or training-time interventions (Kirchenbauer et al., 2023; Pan et al., 2024; Liang et al., 2026), but it requires proactive deployment and the embedded signals may be weakened by rewriting, removal or downstream adaptation. Fingerprinting instead performs post-hoc provenance inference from model-specific signatures, and our work follows this fingerprinting direction with a focus on family-level provenance.

Model fingerprinting: White-box fingerprinting identifies model origin from internal parameters, representations or gradients and achieves strong source verification (Zeng et al., 2024; Zhang et al., 2024a; Yang & Wu, 2024), but it assumes parameter access that is rarely available for API-served models. Black-box fingerprinting instead compares the responses of a target to queries against reference signatures, using crafted or optimized prompts or statistical tests on outputs (Gao et al., 2024; Yan et al., 2026; Shao et al., 2025b; Pasquini et al., 2025; Jin et al., 2024; Gubri et al., 2024). Closer to our setting, ErrorTrace assigns a suspect model to one of the known families from its error patterns (Zang et al., 2025), and Nikolic et al. (2025) test whether a model is derived from a given candidate. Most of these methods are designed for source-specific verification or closed-set classification, while we study family-level provenance for modified LLMs with explicit unknown-family rejection. A method-by-method comparison is provided in Appendix A.

Model modification and provenance robustness: Deployed LLMs are frequently modified through prompt adaptation, sampling, SFT, LoRA and DPO (Giray, 2023; Hewitt et al., 2022; Lv et al., 2024; Hu et al., 2022; Dettmers et al., 2023; Rafailov et al., 2023), and model merging and distillation can further mix behavioral signals across variants or families (Wortsman et al., 2022; Yang et al., 2026; Xu et al., 2024b). Such modifications may preserve, weaken or obscure provenance cues, especially when used adversarially through noise injection or teacher imitation, so we evaluate provenance under both standard post-training modifications and harder anti-provenance transformations.

3 PROBLEM FORMULATION

3.1 FAMILY PROVENANCE TASK

We focus on family-level attribution under black-box access. Given a target model L_t exposed only through an API, the auditor needs to decide whether L_t descends from a known family $\mathcal{F}_o$. A family here refers to a foundation model together with all of its derived variants, where the variants share a common lineage with the foundation model through direct copying, prompt-based adaptation (Giray, 2023) or post-training transformations such as parameter-efficient fine-tuning (PEFT) (Dettmers et al., 2023; Hu et al., 2022), supervised fine-tuning (SFT) (Lv et al., 2024) and direct preference optimization (DPO) (Rafailov et al., 2023). Each family is identified by one pre-trained base checkpoint, so different model sizes of the same series belong to different families. We formalize this task as follows.

Definition 1 (LLM Family Provenance Identification)**.** Let $\mathcal{F}_o$ denote an original LLM family and let L_t denote a target LLM. A provenance identification algorithm $\mathcal{D}$ takes $(L_t, \mathcal{F}_o)$ as input and returns a binary decision $y = \mathcal{D}(L_t, \mathcal{F}_o) \in \{0, 1\}$, where $y = 1$ indicates that L_t belongs to the lineage of $\mathcal{F}_o$ and $y = 0$ otherwise.

This binary formulation generalizes naturally to two practical settings. In the *closed-set* setting, the auditor is given a reference set of K known families $\{\mathcal{F}_1, \ldots, \mathcal{F}_K\}$ and must assign L_t to its most likely family. In the *open-set* setting, the auditor must additionally allow L_t to come from a family that is not in the reference set, in which case L_t is rejected rather than attributed to any known family. The closed-set setting captures controlled provenance disputes among known families, and the open-set setting reflects the more realistic auditing scenario in which the space of candidate families is not exhaustively known in advance. Formally, the closed-set decision assigns L_t to one of the labels $\{1, \ldots, K\}$, and the open-set decision adds the label UNKNOWN, so Definition 1 is the open-set case with $K = 1$.

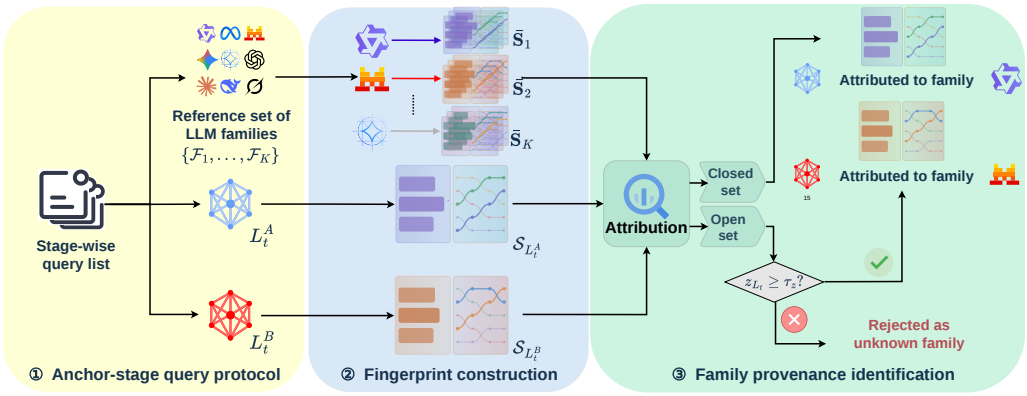


Figure 1: The FamilyTrace framework.

3.2 Threat Model

We consider a setting in which an attacker attempts to reuse, modify or rebrand models from an original LLM family, while an auditor seeks to verify whether a target LLM belongs to that family.

Roles: The attacker appropriates an existing LLM family, possibly modifies it, deploys the resulting model under a new identity, and may further assert ownership over the deployed model. The auditor, who is either the original family owner or an authorized third party, seeks to determine whether a given target L_t descends from a specific reference family $\mathcal{F}_o$.

Attacker's capabilities: We grant the attacker white-box access to all open-weight members of $\mathcal{F}_o$, including their parameters and architecture. Based on this access, the attacker may apply parameter-altering operations such as PEFT (Hu et al., 2022; Dettmers et al., 2023), SFT (Lv et al., 2024) and DPO (Rafailov et al., 2023), parameter-independent operations such as system-prompt or template configuration, and arbitrary combinations of the two. We further consider stronger adversarial transformations such as embedding-noise injection and teacher-based imitation tuning, which we evaluate in our anti-provenance benchmark.

Auditor's capabilities: The auditor may operate against L_t and the LLMs in $\mathcal{F}_o$ under either of two access regimes. In the *white-box regime*, the auditor has full access to internal parameters, hidden states and architecture. In the *black-box regime*, the auditor can only issue prompts through an API and observe the resulting text responses, while logits, hidden states and deployment configurations remain hidden. We focus on the black-box regime in this work, since it corresponds to the realistic case of independent third-party auditing of API-served models. Under this regime, the auditor extracts a model-specific fingerprint from L_t through query-response interactions and uses it to decide whether L_t belongs to the lineage of $\mathcal{F}_o$. We do not assume that the provider detects, filters or adapts its responses to fingerprinting queries.

Design goals: A useful black-box family fingerprint should satisfy three goals. First, *lineage stability* requires the fingerprint to be tied to what the variants of a family share after post-training, rather than to surface-level behaviors easily overridden by deployment configurations. Second, *black-box practicality* requires the extraction to rely only on text outputs and a small query budget, without assumptions about logits, hidden states or developer cooperation. Third, *open-set reliability* requires the fingerprint to support both closed-set attribution and open-set attribution with rejection of unknown families, since the auditor cannot guarantee that every candidate family appears in the reference library. FamilyTrace, introduced next, is designed around these three goals.

4 Method

4.1 Overview

FamilyTrace captures the word association preferences of a model through three components. The first is an anchor-stage query protocol that elicits two complementary views on the same controlled

task: the associative prior under minimal context and its contextual revision as context becomes more specific. The second is a fingerprint that keeps the complete stage-wise ranking of each anchor as a small matrix, the preference profile, which contains both views without loss. The third is a family-level provenance scheme that compares these profiles with the average profiles of known families anchor by anchor, supporting either closed-set attribution or open-set attribution with rejection of unknown families. The full pipeline and key notation are given in Algorithm 1 and Table 2 in the Appendix. An overview of FamilyTrace is shown in Figure 1.

4.2 Anchor-stage query protocol

Why forced-choice ranking under controlled context: Free-form generation mixes the preferences of a model with behavior learned during alignment, which makes it a noisy signal for family identity. Forced-choice tasks instead require the model to commit to one decision among fixed options, so that answers can be compared directly across models. We therefore design a task in which the model ranks a fixed candidate set around an anchor word, and we vary the surrounding context in a controlled way so that the same queries yield both the associative prior and its contextual revision.

Anchors and symmetric candidate design: An anchor is a polysemous or relationally meaningful word that serves as a reference point. Let $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$ be the anchor set. Rather than pairing each anchor with a separately constructed candidate pool, we let the candidate set for a_i be the rest of the anchor pool, $\mathcal{C}_i = \mathcal{A} \setminus \{a_i\}$. This design enforces a symmetric semantic structure where every word serves both as an anchor and as a candidate, so the rankings do not depend on how extra candidate words are chosen. We further partition $\mathcal{A}$ into related groups so that, for any anchor, the candidate pool contains a controlled mix of relatively near, intermediate and far semantic relations. Because every candidate is related to a polysemous anchor under at least one of its senses, several orderings are reasonable, and the ranking depends on which sense and relation the model itself prefers, so it reflects the associations the model has learned.

Progressive stages and query format: A stage is one level of contextual specificity. For each anchor a_i we construct a sequence of stage-wise queries $q_i^{(1)}, q_i^{(2)}, \ldots, q_i^{(T)}$, where $q_i^{(1)}$ embeds a_i in a minimally informative sentence and each subsequent stage injects more semantic evidence, eventually disambiguating which sense or relation of a_i is intended. Adding context step by step controls how much contextual evidence the input provides, so early stages capture the model's prior preference while later stages capture how that preference is refined under stronger evidence. Each query asks the model to rank the candidates in $\mathcal{C}_i$ in decreasing order of semantic relatedness to a_i, yielding a ranking $r_i^{(t)} : \mathcal{C}_i \to \{1, \ldots, |\mathcal{C}_i|\}$ at stage t, where smaller rank means higher affinity. We suppress the model identity from all per-candidate quantities below for readability, and it reappears only in the fingerprint and in cross-model comparisons. A representative query list with anchor *bank*, candidate set {*stream*, *shore*, *account*, *capital*, *channel*} and four stages is shown in Appendix B.1, where the contextual evidence shifts from minimal at the first stage to a clear river sense at the last. The protocol therefore produces, for each model L, a tensor of ranks $\{r_{i,c}^{(t)}\}$ with $r_{i,c}^{(t)} := r_i^{(t)}(c)$, which the next subsection arranges into a fingerprint that keeps both the associative prior and its contextual revision.

4.3 Fingerprint construction

We turn the rankings of each anchor into a small score matrix, the preference profile, and show that it contains both views of Section 4.2.

Stage-wise preference profile: For anchor a_i at stage t, we convert the ranking into scores $s_{i,c}^{(t)} = |\mathcal{C}_i| + 1 - r_{i,c}^{(t)}$, so that the most related candidate receives the highest score. Because each stage ranks all candidates, the score vector $\mathbf{s}_i^{(t)} = [s_{i,c}^{(t)}]_{c \in \mathcal{C}_i}$ is a permutation of $\{1, \ldots, |\mathcal{C}_i|\}$. We stack the T stages into the preference profile $\mathbf{S}_i = [\mathbf{s}_i^{(1)}; \ldots; \mathbf{s}_i^{(T)}] \in \mathbb{R}^{T \times |\mathcal{C}_i|}$, whose t-th row records the preference of the model at stage t. The fingerprint of a model L is the set of its N profiles, $\mathcal{S}_L = \{\mathbf{S}_{L,1}, \ldots, \mathbf{S}_{L,N}\}$.

Associative prior and contextual revision: The profile contains both views that motivate our design. Its first row records the preference of the model under minimal context, the Associative Prior (AP). The later rows record how this preference changes as the context becomes more specific, the

Contextual Revision (CR). For the anchor *bank*, for example, the AP shows whether a model first places the words related to rivers or those related to finance on top, and the CR shows how quickly it moves the words related to rivers up as the sentences mention water and the river. Since every row equals the first row plus the changes accumulated up to that stage, the profile keeps both views without loss, and comparing profiles compares both views at once.

4.4 FAMILY PROVENANCE IDENTIFICATION

Given a reference library of fingerprints from K known families $\{\mathcal{F}_1, \ldots, \mathcal{F}_K\}$, the auditor uses the fingerprint to decide the provenance of a target L_t under one of two mutually exclusive settings. The closed-set setting reflects controlled provenance disputes among known families and assumes that L_t belongs to one of them, so the auditor only needs to assign L_t to its most likely family. The open-set setting allows L_t to come from a family that is not in the reference library, so the auditor must additionally decide whether L_t should be rejected as unknown rather than attributed to any known family. Both settings share the family profile and similarity defined next as a common preliminary step.

Family profile and similarity: For each known family $\mathcal{F}_k$, let $\mathcal{F}'_k = \mathcal{F}_k \setminus \{L_t\}$ be the family without the target. We form the family profile of every anchor by averaging the profiles of its members, $\bar{\mathbf{S}}_{k,i} = |\mathcal{F}'_k|^{-1} \sum_{L \in \mathcal{F}'_k} \mathbf{S}_{L,i}$. Each row of $\bar{\mathbf{S}}_{k,i}$ holds the average score of every candidate within the family at that stage, and ordering the candidates by these averages gives their Borda count ranking, a standard rule for combining several rankings into one. We compare the target with a family anchor by anchor. For anchor a_i, let $\rho_i(L_t, \mathcal{F}_k)$ be the Pearson correlation between the flattened profiles $\mathrm{vec}(\mathbf{S}_{L_t,i})$ and $\mathrm{vec}(\bar{\mathbf{S}}_{k,i})$. The similarity between the target and the family is the median over anchors, $\mathrm{Sim}(L_t, \mathcal{F}_k) = \mathrm{median}_i\ \rho_i(L_t, \mathcal{F}_k)$. We take the median rather than the mean because it limits the influence of individual anchors: with six anchors, the similarity stays within the range of the unaffected anchors even if two anchors produce unusual rankings. The proposition below explains what this similarity measures.

Proposition 1 (Similarity as normalized rank agreement)**.** Assume that at every stage of anchor a_i, L_t and every member of $\mathcal{F}'_k$ rank all candidates without ties. Then $\rho_i(L_t, \mathcal{F}_k) = \bar{\rho}_i(L_t, \mathcal{F}'_k) \big/ \sqrt{\bar{\rho}_i(\mathcal{F}'_k, \mathcal{F}'_k)}$, where $\bar{\rho}_i(L_t, \mathcal{F}'_k)$ is the Spearman correlation between the rankings of L_t and a member of $\mathcal{F}'_k$, averaged over the T stages and over all members, and $\bar{\rho}_i(\mathcal{F}'_k, \mathcal{F}'_k)$ is the same average taken over all ordered pairs of members, including each member paired with itself.

The numerator measures how closely the target agrees with the members of the family, and the denominator measures how closely the members agree with each other, so a family whose members agree strongly requires stronger agreement from the target. Averaging the similarities between the target and individual members would keep only the numerator. The proof is given in Appendix B.2.

Closed-set attribution: The target is assigned to the most similar family, $\hat{k} = \arg\max_{k \in \{1,\ldots,K\}} \mathrm{Sim}(L_t, \mathcal{F}_k)$, and the predicted family of L_t in the closed-set setting is $\mathcal{F}_{\hat{k}}$.

Open-set attribution with family-normalized rejection: In the open-set setting, returning the closest known family alone cannot distinguish a model that genuinely belongs to that family from one that is merely closer to it than to the others. We therefore reuse the closed-set procedure to obtain $\hat{k}$, and calibrate the similarity $\mathrm{Sim}(L_t, \mathcal{F}_{\hat{k}})$ against the similarities that members of $\mathcal{F}_{\hat{k}}$ obtain under the same procedure. To keep the target out of its own calibration, we first remove L_t from the reference library. For each remaining member L_j of $\mathcal{F}_{\hat{k}}$, we compute $\mathrm{Sim}(L_j, \mathcal{F}_{\hat{k}})$ with L_j also left out of the family profile, and estimate the mean $\mu_{\hat{k}}$ and standard deviation $\sigma_{\hat{k}}$ of these values. This calibration requires each reference family to contain at least three models. We then compute the family-normalized score $z_{L_t} = \big(\mathrm{Sim}(L_t, \mathcal{F}_{\hat{k}}) - \mu_{\hat{k}}\big)/(\sigma_{\hat{k}} + \epsilon)$, where ϵ is a small constant that avoids division by zero. If $z_{L_t} < \tau_z$ for a chosen threshold τ_z, L_t is rejected as not belonging to any known family, otherwise it is attributed to $\mathcal{F}_{\hat{k}}$. By calibrating against the predicted family's own spread rather than against a global threshold, the rejection rule adapts to the internal agreement of each family. The choice of τ_z controls the trade-off between the unknown-family detection rate and the known-family false-rejection rate, which we study empirically in Section 5.

Table 1: Global closed-set (CS) and open-set (OS) metrics of all methods on the two benchmarks.

(a) Non-adversarial modification

Metric	HideNSeek	DuFFin	ProFLingo	SEF	MET	LLMmap	ZEROPRINT	FamilyTrace
CS Acc. ↑	0.545	0.727	0.455	0.909	0.818	0.926	0.727	0.963
OS Acc. ↑	0.415	0.724	0.545	0.755	0.709	0.675	0.642	0.879
TPR ↑	0.373	0.582	0.584	0.679	0.619	0.214	0.612	0.883
FPR ↓	0.306	0.116	0.307	0.128	0.236	0.103	0.184	0.113
AUROC ↑	0.583	0.850	0.775	0.783	0.888	0.830	0.807	0.950

(b) Anti-provenance modification

Metric	HideNSeek	DuFFin	ProFLingo	SEF	MET	LLMmap	ZEROPRINT	FamilyTrace
CS Acc. ↑	0.520	0.720	0.400	0.700	0.540	0.820	0.720	0.854
OS Acc. ↑	0.467	0.617	0.317	0.583	0.450	0.477	0.617	0.795
TPR ↑	0.410	0.376	0.529	0.544	0.433	0.558	0.452	0.765
FPR ↓	0.353	0.219	0.353	0.249	0.300	0.342	0.244	0.177
AUROC ↑	0.581	0.778	0.548	0.794	0.575	0.684	0.785	0.854

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

LLM benchmark and variants: We evaluate FamilyTrace and the baselines on two benchmarks. The non-adversarial benchmark contains 55 LLMs from six source families, namely Gemma-2-2B (Team, 2024a), Llama-3.1-8B (Grattafiori et al., 2024), Mistral-7B (Jiang et al., 2023), Qwen-2.5-7B (Yang et al., 2024; Team, 2024b), Qwen-2.5-14B (Yang et al., 2024; Team, 2024b) and TinyLlama-1.1B (Zhang et al., 2024b), and covers instruct tuning, prompt-based adaptation (Giray, 2023), sampling (Hewitt et al., 2022), SFT (Lv et al., 2024), DPO (Rafailov et al., 2023) and LoRA (Hu et al., 2022). The anti-provenance benchmark contains 50 LLMs from the same six families, organized into five adversarial settings of 10 models each: Noisy SFT (N-SFT) (Dang et al., 2025), Single-Teacher Imitation (STI), Multi-Teacher Imitation (MTI) (Gharami et al., 2025), Noisy Single-Teacher Imitation (N-STI) and Noisy Multi-Teacher Imitation (N-MTI). For STI, MTI, N-STI and N-MTI, the provenance label remains the original base family rather than the teacher family. Detailed benchmark construction is described in Appendix B.3.

Baselines: We compare FamilyTrace with seven black-box baselines: HideNSeek (Iourovitski et al., 2024), DuFFin (Yan et al., 2026), ProFLingo (Jin et al., 2024), SEF (Shao et al., 2025a), MET (Gao et al., 2024), LLMmap (Pasquini et al., 2025) and ZEROPRINT (Shao et al., 2025b). Each baseline is described in Appendix B.4.

Evaluation metrics: For the closed-set setting, we report **CS Accuracy (CS Acc.)**, the proportion of LLMs correctly attributed to their source family. For the open-set setting, we treat unknown samples as the positive class and known samples as the negative class, and report four metrics: **OS Accuracy (OS Acc.)** is the proportion of LLMs that are either correctly attributed to their known family or correctly rejected as unknown, **TPR** is the proportion of unknown LLMs correctly rejected as unknown, **FPR** is the proportion of known LLMs incorrectly rejected as unknown, and **AUROC** summarizes the overall known-versus-unknown separability across all rejection thresholds. Higher CS Acc., OS Acc., TPR and AUROC indicate stronger provenance performance, while lower FPR indicates fewer false rejections of known LLMs.

Parameter settings: FamilyTrace uses a symmetric query list with 6 anchors and 4 stages per anchor, yielding 24 queries per target LLM. All baselines are evaluated under the same query budget for fairness. For the open-set evaluation, we adopt a leave-one-family-out protocol where one of the six families is held out as unknown in each round and the remaining five serve as known families. The default open-set threshold is set to $\tau_z = -2$, and the constant ϵ is set to 10^{-6}. For FamilyTrace, we report the mean over 100 query lists generated by the same design rule.

5.2 EXPERIMENTAL RESULTS

Effectiveness of FamilyTrace**:** Table 1 reports the overall metrics of all methods on the two benchmarks, and Table 4 (Appendix) breaks them down by LLM modification type. FamilyTrace achieves the strongest overall closed-set and open-set performance among all methods, with 0.963 CS Acc., 0.879 OS Acc. and 0.950 AUROC on the non-adversarial benchmark, and 0.854 CS Acc., 0.795 OS Acc. and 0.854 AUROC on the anti-provenance benchmark. A detailed per-modification analysis and baseline comparison are provided in Appendix B.5.

Performance across modification types: Across the modification types in Table 4 (Appendix), FamilyTrace obtains the highest OS Acc. in 9 of the 11 settings. On the non-adversarial benchmark, its CS Acc. stays at or above 0.909 under every modification, and the main drops are in OS Acc., which falls to 0.692 under sampling and 0.773 under SFT. On the anti-provenance benchmark, the hardest settings are noisy SFT, with 0.710 CS Acc., and teacher imitation without noise, with 0.823 and 0.839 CS Acc. for STI and MTI, compared with 0.950 for their noisy counterparts.

Comparison across LLM families: Table 6 (Appendix) reports family-wise CS Acc. and OS Acc. of all methods. To further visualize the family separability of each method's fingerprint, we also include 2D t-SNE projections (Van der Maaten & Hinton, 2008) in Figures 4, 5, 6 and 7 (Appendix). FamilyTrace yields more compact within-family clusters and clearer cross-family separation than the baselines on both benchmarks. A family-level analysis is provided in Appendix B.6.

Efficiency of FamilyTrace**:** Table 7 (Appendix) compares the end-to-end provenance runtime of all methods on both benchmarks. FamilyTrace achieves the lowest runtime, taking 813 s on the non-adversarial benchmark and 963 s on the anti-provenance benchmark, since it avoids the LLM-as-a-judge calls used by HideNSeek and the inference-model training used by LLMmap.

Impact of anchor and stage numbers: Tables 8 and 9 (Appendix) report FamilyTrace under different anchor and stage numbers, each averaged over 30 query lists. More anchors and stages improve accuracy, mainly on the anti-provenance benchmark, where CS Acc. rises from 0.735 with one stage to 0.857 with four stages. The default design with 6 anchors and 4 stages balances accuracy against a budget of 24 queries.

Impact of threshold τ_z on open-set setting: We study how the open-set threshold τ_z controls the trade-off between unknown-family rejection and known-family preservation. Table 10 (Appendix) reports TPR, FPR and OS Acc. under different τ_z values, and Figures 8 and 9 (Appendix) visualize the trade-off and the ROC curves. The best OS Acc. is reached at $\tau_z = -3.5$ on both benchmarks, and thresholds in $[-4, -2]$ form a stable operating range. A detailed analysis is provided in Appendix B.9.

Robustness to anchor tampering: We simulate an attacker who controls the responses to k of the six anchors and either replaces their rankings with the profile of another family (impersonation) or reverses them (reversal). As shown in Table 11 (Appendix), with two tampered anchors on the non-adversarial benchmark, taking the median over anchors keeps CS Acc. at 0.826 under impersonation and 0.903 under reversal, while the mean falls to 0.691 and 0.505. With three or more tampered anchors, both aggregations fail. Advanced attacks such as targeted fine-tuning on the anchor words are discussed as a limitation in Appendix B.18.

6 DISCUSSION

Same-series LLM provenance across base-model sizes: We further evaluate whether FamilyTrace can separate families from the same model series that differ in base-model size. We construct five Qwen-2.5 families of 0.5B, 3B, 7B, 14B and 32B, each containing 10 modified variants of its base model. As shown in Table 12 (Appendix), FamilyTrace achieves 0.940 CS Acc. and 0.868 OS Acc. on these families, which suggests that association preferences also differ between base models of the same series. A possible extension is hierarchical provenance, which further separates successive generations within one series.

Generalization across query lists: To test whether FamilyTrace depends on a specific query list, we evaluate it on 100 query lists built with the same design, with six anchors, four stages and symmetric candidates. The lists fall into three groups: 41 paraphrase the queries, 38 change the anchor words and 21 change the instruction. As shown in Table 13 and Figure 11 (Appendix), the average CS Acc.

of the three groups differs by at most 0.015 and 0.019 on the two benchmarks. FamilyTrace therefore does not depend on a particular wording or set of anchor words.

Effect of the query design: To examine whether the anchor-stage design is necessary, we replace our queries with two alternatives that keep the ranking task and the budget of 24 queries: random common nouns with the same staged sentences, and unambiguous anchors whose candidates follow a common-sense ordering, with four equally informative sentences. As shown in Table 14 (Appendix), over 10 lists of each type, random words reach 0.722 and 0.648 CS Acc. on the two benchmarks, with standard deviations of 0.129 and 0.163 across lists, and common-sense orderings reach 0.844 and 0.780, while 10 lists of our design reach 0.960 and 0.832. Random words thus give unstable results, and common-sense orderings carry less family information, which is consistent with our hypothesis.

Base model versus fine-tuning data: To separate the effect of the base model from that of the fine-tuning data, we fine-tune three base models, Llama-3.1-8B, Mistral-7B-v0.1 and Qwen-2.5-14B, on the same three datasets with three random seeds each, which gives 27 models. Models that share a base model have a mean similarity of 0.241, while models fine-tuned on the same data from different base models have a mean similarity of -0.005, and a permutation test finds this difference significant, with a p-value below 10^{-4}. Even when all models fine-tuned on the same dataset as the target are removed from the reference library, 18 of the 27 models are still attributed to their base family, compared with 9 expected by chance (Appendix B.14).

Variants of FamilyTrace**:** Table 16 (Appendix) compares FamilyTrace with AP-only, which keeps stage 1, and CR-only, which keeps stages 2 to 4. On the anti-provenance benchmark, AP-only drops to 0.769 CS Acc., while CR-only and the full profile reach 0.848 and 0.854, so the later stages are more robust to anti-provenance modifications, and the full profile obtains the highest OS Acc. and AUROC on the non-adversarial benchmark. For family-level aggregation (Appendix B.15), averaging the similarities to individual members, which drops the normalization in Proposition 1, lowers CS Acc. on the anti-provenance benchmark from 0.854 to 0.587.

External API model separability: Table 18 (Appendix) evaluates the open-set rejection of four closed-source API models, Claude Sonnet 4 (Anthropic, 2025), DeepSeek Chat v2 (DeepSeek-AI, 2024), OpenAI GPT-4o (Hurst et al., 2024) and xAI Grok-3 (xAI, 2025), against the six known open-source families. When each API model is treated as an unknown family, its AUROC ranges from 0.851 to 0.905 with the non-adversarial benchmark as the reference library and from 0.798 to 0.887 with the anti-provenance benchmark. These results suggest that FamilyTrace provides a useful open-set signal for rejecting external families.

Performance on model-merging and distillation LLMs: Model merging (Yang et al., 2026; Wortsman et al., 2022) and distillation (Xu et al., 2024b) can mix lineage evidence from several checkpoints or transfer teacher behavior into a student model. We evaluate FamilyTrace on a 30-LLM pool covering both modifications with the default query list. As shown in Table 19 (Appendix), it achieves 0.864 CS Acc., 0.773 OS Acc. and 0.870 AUROC on model-merging variants, and 0.750 CS Acc., 0.625 OS Acc. and 0.817 AUROC on distillation variants, where the teacher partially shifts the association preferences of the student. Over the whole pool, FamilyTrace achieves the highest CS Acc., OS Acc., TPR and AUROC among the evaluated methods (Appendix B.17.1), and the pool construction is described in Appendix B.17.

7 Conclusion

In this paper, we proposed FamilyTrace, a black-box family-level provenance method that captures the word association preferences of a model through a forced-choice ranking task, combining its associative prior under minimal context with its contextual revision under progressively informative context. We showed that comparing a model with the average rankings of a family measures its agreement with the family relative to the agreement among the family members. Experiments on non-adversarial and anti-provenance benchmarks across six LLM families show that FamilyTrace outperforms seven black-box baselines and remains stable across query lists. These results suggest that association preferences, measured with ranking queries in which several answers are reasonable, provide a practical signal for black-box LLM provenance. Future work includes hierarchical provenance that separates successive generations within one model series, stronger open-set calibration among closely related families, and interventions on model components to locate this signal.

ETHICS STATEMENT

FamilyTrace is intended for defensive provenance auditing, such as checking whether a deployed model is likely to descend from a known family when only black-box access is available. Provenance signals could also be misused for competitive intelligence or to design evasion strategies, and a false attribution may cause legal or reputational harm. FamilyTrace outputs should be treated as statistical evidence within a broader audit rather than as a standalone legal or commercial determination. All experiments use publicly available model checkpoints, public datasets and commercial APIs under their terms of service, and involve no human subjects or private user data. Further discussion is given in Appendix B.18.

REPRODUCIBILITY STATEMENT

Section 4 and Algorithm 1 specify the full procedure, and Appendix B.1 shows the query format. Appendix B.3 describes the construction of both benchmarks, including training data and hyperparameters of the self-constructed variants, and Appendix B.4 describes the baselines. Appendix B.19 gives a step-by-step reproduction protocol, and Appendix B.7 reports the compute resources. The implementation, query lists and model identifiers will be released upon acceptance. Appendix B.2 gives the proof of Proposition 1.

AI USE STATEMENT

Generative AI was used in the experimental construction described in this paper. Teacher LLMs generated training responses for the imitation variants in Appendix B.3 and the distillation variants in Appendix B.17. The GPT-5.4 API generated the 100 query-list variants described in Appendix B.12. We also used LLMs to aid and polish the writing of this paper, and the authors reviewed all AI-assisted content and take full responsibility for it.

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.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7319–7328, 2021.

Anthropic. System card: Claude Opus 4 & Claude Sonnet 4. `https://www.anthropic.com/claude-4-system-card`, 2025.

Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A. Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, Eric Wallace, David Rolnick, and Florian Tramèr. Stealing part of a production language model. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=VE3yWXt3KB`.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Kieu Dang, Phung Lai, NhatHai Phan, Yelong Shen, Ruoming Jin, and Abdallah Khreishah. δ-STEAL: LLM stealing attack with local differential privacy. *arXiv preprint arXiv:2510.21946*, 2025.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL `https://arxiv.org/abs/2405.04434`.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023.

Irena Gao, Percy Liang, and Carlos Guestrin. Model equality testing: Which model is this API serving? *arXiv preprint arXiv:2410.20247*, 2024.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 30–45, 2022.

Kanchon Gharami, Hansaka Aluvihare, Shafika Showkat Moni, and Berker Peköz. Clone what you can't steal: Black-box LLM replication via logit leakage and distillation. *arXiv preprint arXiv:2509.00973*, 2025.

Louie Giray. Prompt engineering with ChatGPT: a guide for academic writers. *Annals of biomedical engineering*, 51(12):2629–2633, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 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.

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea preprints*, 1(3):1–26, 2023.

John Hewitt, Christopher D Manning, and Percy Liang. Truncation sampling as language model desmoothing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 3414–3427, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Dmitri Iourovitski, Sanat Sharma, and Rakshak Talwar. HideNSeek: Fingerprinting large language models with evolutionary learning. *arXiv preprint arXiv:2408.02871*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B, 2023. URL `https://arxiv.org/abs/2310.06825`.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *ACM Transactions on Software Engineering and Methodology*, 35(2): 1–72, 2026.

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.

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.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. OpenAssistant conversations – democratizing large language model alignment. In *Advances in Neural Information Processing Systems*, volume 36, pp. 47669–47681, 2023.

Pranjal Kumar. Large language models (LLMs): survey, technical frameworks, and future challenges. *Artificial Intelligence Review*, 57(10):260, 2024.

Yuqing Liang, Jiancheng Xiao, Wensheng Gan, and Philip S Yu. Watermarking techniques for large language models: A survey. *Artificial Intelligence Review*, 59(2):74, 2026.

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. The unlocking spell on base LLMs: Rethinking alignment via in-context learning. In *International Conference on Learning Representations*, pp. 24907–24933, 2024.

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.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7052–7063, 2021.

Kai Lv, Yuqing Yang, Tengxiao Liu, Qipeng Guo, and Xipeng Qiu. Full parameter fine-tuning for large language models with limited resources. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8187–8198, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372, 2022.

Ivica Nikolic, Teodora Baluta, and Prateek Saxena. Model provenance testing for large language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 34126–34153, 2025.

Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, et al. MarkLLM: An open-source toolkit for LLM watermarking. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 61–71, 2024.

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.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Mubashar Raza, Zarmina Jahangir, Muhammad Bilal Riaz, Muhammad Jasim Saeed, and Muhammad Awais Sattar. Industrial applications of large language models. *Scientific Reports*, 15(1): 13755, 2025.

Shuo Shao, Yiming Li, Yu He, Hongwei Yao, Wenyuan Yang, Dacheng Tao, and Zhan Qin. SoK: Large language model copyright auditing via fingerprinting. *arXiv preprint arXiv:2508.19843*, 2025a.

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. *arXiv preprint arXiv:2510.06605*, 2025b.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. OpenAI GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

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

Gemma Team. Gemma. 2024a. doi: 10.34740/KAGGLE/M/3301. URL `https://www.kaggle.com/m/3301`.

Qwen Team. Qwen2.5: A party of foundation models, September 2024b. URL `https://qwenlm.github.io/blog/qwen2.5/`.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

xAI. Grok 3 beta — the age of reasoning agents. `https://x.ai/news/grok-3`, 2025.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *International Conference on Learning Representations*, pp. 35623–35646, 2024.

Jiashu Xu, Fei Wang, Mingyu Derek 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*, pp. 3277–3306, 2024a.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024b.

Zhao Xu, Fan Liu, and Hao Liu. Bag of tricks: Benchmarking of jailbreak attacks on LLMs. *Advances in Neural Information Processing Systems*, 37:32219–32250, 2024c.

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.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in LLMs, MLLMs, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 58(8):1–41, 2026.

Zhiguang Yang and Hanzhou Wu. A fingerprint for large language models. *arXiv preprint arXiv:2407.01235*, 2024.

Itay Yona, Ilia Shumailov, Jamie Hayes, and Nicholas Carlini. Stealing user prompts from mixture of experts. *arXiv preprint arXiv:2410.22884*, 2024.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, pp. 45040–45061, 2024.

Chuanchao Zang, Xiangtao Meng, Wenyu Chen, Tianshuo Cong, Zha Yaxing, Dong Qi, Zheng Li, and Shanqing Guo. ErrorTrace: A black-box traceability mechanism based on model family error space. In *Advances in Neural Information Processing Systems*, volume 38, pp. 68550–68578, 2025.

Boyi Zeng, Lizheng Wang, Yuncong Hu, Yi Xu, Chenghu Zhou, Xinbing Wang, Yu Yu, and Zhouhan Lin. HuRef: Human-readable fingerprint for large language models. *Advances in Neural Information Processing Systems*, 37:126332–126362, 2024.

Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. REEF: Representation encoding fingerprints for large language models. *arXiv preprint arXiv:2410.14273*, 2024a.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024b.

A DETAILED RELATED WORK

This appendix expands on the compact related-work overview in Section 2 and discusses watermarking, white-box fingerprinting, black-box fingerprinting and model modifications in greater detail.

Watermarking: Watermarking methods embed ownership signals into LLMs before release, either through generation-time interventions that bias token sampling toward an embedded pattern or through training-time interventions that imprint specific responses to trigger inputs (Kirchenbauer et al., 2023; Pan et al., 2024; Liang et al., 2026). The embedded signal can later be detected by the model owner to verify authenticity. However, watermarking requires proactive deployment, which makes it inapplicable to the vast majority of already-released LLMs. The embedded signals can also be weakened by paraphrasing, rewriting, removal attacks and trigger collisions when the watermarked model is further adapted, which limits its reliability for post-hoc provenance auditing.

White-box fingerprinting: White-box fingerprinting identifies model origin from internal parameters, hidden-state representations, gradients or other internal evidence. HuRef (Zeng et al., 2024) extracts human-readable fingerprints from parameter invariants, REEF (Zhang et al., 2024a) encodes representation-level signatures across layers, and additional parameter-level methods further demonstrate strong source verification accuracy (Yang & Wu, 2024). The strength of these methods comes from direct access to internal evidence, but the same access requirement also limits their applicability, since commercial APIs and suspicious deployed models typically expose only text outputs. This limitation motivates black-box provenance methods that operate through queries alone.

Black-box fingerprinting: Black-box fingerprinting probes a target model through prompts and compares its observed text behavior against reference signatures. Following the targeted and untargeted taxonomy of Shao et al. (2025a), untargeted methods use shared or automatically constructed probes and derive fingerprints from various behavioral cues. MET (Gao et al., 2024) compares output distributions of the target and reference LLMs through a kernel-based maximum mean discrepancy statistic. DuFFin (Yan et al., 2026) combines a trigger-pattern fingerprint encoded from response trajectories with a knowledge-level fingerprint formed from cross-domain answer patterns. ZEROPRINT (Shao et al., 2025b) estimates a Jacobian-based surrogate from controlled prompt perturbations on code-oriented queries. LLMmap (Pasquini et al., 2025) encodes query-response traces with a learned transformer-based inference model and uses contrastive embeddings for unseen LLMs. Targeted methods such as ProFLingo (Jin et al., 2024) instead optimize prefix-based queries on source-specific question-target pairs to elicit characteristic responses for verification. These methods provide useful black-box evidence, but are often designed for source-specific verification, known-model classification or closed-set settings, whereas we study family-level provenance for modified LLMs with explicit unknown-family rejection.

Model modification and provenance robustness: Deployed LLMs are frequently modified through prompt adaptation (Giray, 2023), sampling (Hewitt et al., 2022), SFT (Lv et al., 2024), LoRA (Hu et al., 2022; Dettmers et al., 2023) and DPO (Rafailov et al., 2023), all of which can be applied with limited compute and either preserve or weaken family-level provenance cues. Model merging (Wortsman et al., 2022; Yang et al., 2026) and distillation (Xu et al., 2024b) are more disruptive, since merging can blend behavioral signals across same-family variants and distillation can transfer cross-family signals from teacher to student. When used adversarially through embedding-noise injection or teacher-imitation tuning, these modifications can further obscure provenance cues. We therefore evaluate FamilyTrace under both standard post-training modifications and harder anti-provenance transformations.

B TECHNICAL APPENDICES

B.1 ALGORITHM AND QUERY LIST EXAMPLE

Algorithm 1 summarizes the full procedure of FamilyTrace, and Table 2 lists the main notation. The box below shows the default query list for the anchor *bank*.

Algorithm 1 FamilyTrace Family Provenance Identification

Input: target LLM L_t; anchor set $\mathcal{A} = \{a_1, \ldots, a_N\}$; stage count T; reference profiles $\{\mathcal{S}_L : L \in \bigcup_{k=1}^{K} \mathcal{F}_k\}$; provenance setting $\mathrm{mode} \in \{\text{CLOSED-SET}, \text{OPEN-SET}\}$; rejection threshold τ_z (used only in open-set mode); stability constant ϵ

Output: predicted family index $\hat{k}$, or rejection as unknown family in open-set mode

// Step 1: build the stage-wise profiles of the target
for each anchor $a_i \in \mathcal{A}$ **do**
  $\mathcal{C}_i \leftarrow \mathcal{A} \setminus \{a_i\}$
  for $t = 1, \ldots, T$ **do**
    Issue $q_i^{(t)}$ to L_t, obtain ranking $r_i^{(t)}$, and set $s_{i,c}^{(t)} \leftarrow |\mathcal{C}_i| + 1 - r_{i,c}^{(t)}$
  end for
  $\mathbf{S}_{L_t,i} \leftarrow [\mathbf{s}_i^{(1)}; \ldots; \mathbf{s}_i^{(T)}]$
end for
// Step 2: compare the target with every known family
for $k = 1, \ldots, K$ **do**
  $\bar{\mathbf{S}}_{k,i} \leftarrow$ mean of $\mathbf{S}_{L,i}$ over $L \in \mathcal{F}_k \setminus \{L_t\}$, for each anchor a_i
  $\mathrm{Sim}(L_t, \mathcal{F}_k) \leftarrow$ median over anchors of $\mathrm{Pearson}(\mathrm{vec}\,\mathbf{S}_{L_t,i}, \mathrm{vec}\,\bar{\mathbf{S}}_{k,i})$
end for
$\hat{k} \leftarrow \arg\max_k \mathrm{Sim}(L_t, \mathcal{F}_k)$
if mode $=$ CLOSED-SET **then**
  return $\hat{k}$
end if
// Step 3: open-set rejection calibrated on the predicted family
Remove L_t from the reference library
for each $L_j \in \mathcal{F}_{\hat{k}} \setminus \{L_t\}$ **do**
  $u_j \leftarrow \mathrm{Sim}(L_j, \mathcal{F}_{\hat{k}})$, with L_j also left out of the family profile
end for
$\mu_{\hat{k}}, \sigma_{\hat{k}} \leftarrow$ mean and standard deviation of $\{u_j\}$
$z_{L_t} \leftarrow (\mathrm{Sim}(L_t, \mathcal{F}_{\hat{k}}) - \mu_{\hat{k}})/(\sigma_{\hat{k}} + \epsilon)$
if $z_{L_t} < \tau_z$ **then**
  return UNKNOWN
else
  return $\hat{k}$
end if

Default query list for anchor *bank*

Query template (identical for all stages):

```
The activity described is simple:  {sentence}
Rank these five words from most to least related in meaning to "bank"
in that context:
stream, shore, account, capital, channel
Reply with one line only:
ANSWER: word > word > word > word > word
ANSWER:
```

Stage 1: She waited by the bank.
Stage 2: She waited by the bank and watched the water.
Stage 3: She waited by the bank and watched the river flow quietly.
Stage 4: She waited by the bank and watched the river flow quietly beside the shore.

The six words in this list form three semantic groups, G1 (*stream*, *shore*) related to rivers, G2 (*account*, *capital*) related to finance and G3 (*bank*, *channel*) related to both. The groups are shown for readability and are not part of the prompts. Table 3 lists all 24 queries of the default query list. Every query uses the template shown above, with the anchor and its five candidates substituted.

Table 2: Summary of key notation.

Symbol	Meaning
L_t	Target LLM to be attributed
$\mathcal{A} = \{a_1, \ldots, a_N\}$	Anchor set with N anchors
$\mathcal{C}_i = \mathcal{A} \setminus \{a_i\}$	Candidate set for anchor a_i
T	Number of progressive query stages
$r_i^{(t)}$	Ranking over $\mathcal{C}_i$ returned at stage t of anchor a_i
$\mathcal{F}_1, \ldots, \mathcal{F}_K$	Set of K known LLM families
$s_{i,c}^{(t)}$	Score $\lvert\mathcal{C}_i\rvert + 1 - r_{i,c}^{(t)}$ of candidate c at stage t of anchor a_i
$\mathbf{S}_{L,i}$	Stage-wise preference profile of model L at anchor a_i
$\mathcal{S}_L$	Fingerprint of model L, the set of its N profiles
AP, CR	First row of a profile, and its later rows (stages 2 to T)
$\bar{\mathbf{S}}_{k,i}$	Family profile of $\mathcal{F}_k$ at anchor a_i (target left out)
$\rho_i(L, \mathcal{F}_k)$	Pearson correlation of flattened profiles at anchor a_i
$\mathrm{Sim}(L, \mathcal{F}_k)$	Median of ρ_i over anchors
$\hat{k}$	Predicted top-1 family index for L_t
z_{L_t}	Family-normalized z-score for open-set rejection
τ_z	Open-set rejection threshold

Table 3: All 24 queries of the default query list. Each sentence is inserted into the template shown in the example above.

Anchor	Candidates	Stage	Sentence
stream	shore, account, capital, bank, channel	1	The hikers rested near the stream.
		2	The hikers rested near the stream and heard running water.
		3	The hikers rested near the stream and heard running water over stones.
		4	The hikers rested near the stream and heard running water over stones in the narrow river bed.
shore	stream, account, capital, bank, channel	1	They walked along the shore.
		2	They walked along the shore beside the water.
		3	They walked along the shore beside the water as waves touched the sand.
		4	They walked along the shore beside the water as waves touched the sand by the river mouth.
account	stream, shore, capital, bank, channel	1	She checked the account.
		2	She checked the account before making a payment.
		3	She checked the account before making a payment at the bank.
		4	She checked the account before making a payment at the bank and reviewing the balance.
capital	stream, shore, account, bank, channel	1	The firm needed more capital.
		2	The firm needed more capital to expand the business.
		3	The firm needed more capital to expand the business through the bank.
		4	The firm needed more capital to expand the business through the bank and support investment.
bank	stream, shore, account, capital, channel	1	She waited by the bank.
		2	She waited by the bank and watched the water.
		3	She waited by the bank and watched the river flow quietly.
		4	She waited by the bank and watched the river flow quietly beside the shore.
channel	stream, shore, account, capital, bank	1	The boat moved through the channel.
		2	The boat moved through the channel between two water edges.
		3	The boat moved through the channel between two water edges toward the river.
		4	The boat moved through the channel between two water edges toward the river shore.

B.2 Proof of Proposition 1

Proof of Proposition 1. Write $C = |\mathcal{C}_i|$, $n = |\mathcal{F}'_k|$ and $\mathbf{X} = \mathbf{S}_{L_t,i}$, and let $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$ be the profiles of the members of $\mathcal{F}'_k$ at anchor a_i, so that $\bar{\mathbf{S}}_{k,i} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{Y}_j$.

Step 1: centering. Every row of $\mathbf{X}$ and of each $\mathbf{Y}_j$ is a permutation of $\{1, \ldots, C\}$. Hence every row has mean $\bar{s} = (C+1)/2$ and sum of squared deviations

$$\kappa = \sum_{v=1}^{C} (v - \bar{s})^2 = \frac{C(C^2-1)}{12}.$$

The rows of $\bar{\mathbf{S}}_{k,i}$ are averages of such rows and also have mean $\bar{s}$. Let $\tilde{\mathbf{X}}$, $\tilde{\mathbf{Y}}_j$ and $\tilde{\mathbf{W}}$ denote $\mathbf{X}$, $\mathbf{Y}_j$ and $\bar{\mathbf{S}}_{k,i}$ with $\bar{s}$ subtracted from every entry. Since $\bar{s}$ is also the mean of all entries of each flattened matrix, the Pearson correlation can be written as

$$\rho_i(L_t, \mathcal{F}_k) = \frac{\langle \tilde{\mathbf{X}}, \tilde{\mathbf{W}} \rangle_F}{\|\tilde{\mathbf{X}}\|_F \, \|\tilde{\mathbf{W}}\|_F},$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product.

Step 2: inner products of profiles. For two rows $\mathbf{x}, \mathbf{y}$ that are permutations of $\{1, \ldots, C\}$, the Spearman correlation equals the Pearson correlation of the scores,

$$\rho_S(\mathbf{x}, \mathbf{y}) = \frac{1}{\kappa} \sum_{c=1}^{C} (x_c - \bar{s})(y_c - \bar{s}).$$

Summing over the T rows gives

$$\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}}_j \rangle_F = \kappa T \, \bar{\rho}(\mathbf{X}, \mathbf{Y}_j),$$
$$\|\tilde{\mathbf{X}}\|_F^2 = \kappa T,$$

where $\bar{\rho}(\mathbf{X}, \mathbf{Y}_j)$ is the Spearman correlation averaged over the T stages.

Step 3: numerator and denominator. Since $\tilde{\mathbf{W}} = \frac{1}{n}\sum_{j=1}^{n} \tilde{\mathbf{Y}}_j$, linearity of the inner product gives

$$\langle \tilde{\mathbf{X}}, \tilde{\mathbf{W}} \rangle_F = \frac{1}{n} \sum_{j=1}^{n} \kappa T \, \bar{\rho}(\mathbf{X}, \mathbf{Y}_j) = \kappa T \, \bar{\rho}_i(L_t, \mathcal{F}'_k),$$

$$\|\tilde{\mathbf{W}}\|_F^2 = \frac{1}{n^2} \sum_{j=1}^{n} \sum_{l=1}^{n} \kappa T \, \bar{\rho}(\mathbf{Y}_j, \mathbf{Y}_l) = \kappa T \, \bar{\rho}_i(\mathcal{F}'_k, \mathcal{F}'_k).$$

Substituting these quantities into the Pearson correlation gives

$$\rho_i(L_t, \mathcal{F}_k) = \frac{\kappa T \, \bar{\rho}_i(L_t, \mathcal{F}'_k)}{\sqrt{\kappa T} \, \sqrt{\kappa T \, \bar{\rho}_i(\mathcal{F}'_k, \mathcal{F}'_k)}} = \frac{\bar{\rho}_i(L_t, \mathcal{F}'_k)}{\sqrt{\bar{\rho}_i(\mathcal{F}'_k, \mathcal{F}'_k)}},$$

which is the identity in Proposition 1. The denominator is well defined because $\bar{\rho}_i(\mathcal{F}'_k, \mathcal{F}'_k) = \|\tilde{\mathbf{W}}\|_F^2/(\kappa T) \geq 0$, with equality only when all candidates receive the same average score at every stage, in which case the correlation with the family profile is undefined. □

B.3 LLM benchmark and variants description

We use two complementary benchmarks for evaluation. The non-adversarial benchmark contains 55 fine-tuned or instruction-aligned LLMs from six source families, namely Gemma-2-2B (Team, 2024a), Llama-3.1-8B (Grattafiori et al., 2024), Mistral-7B (Jiang et al., 2023), Qwen-2.5-7B, Qwen-2.5-14B (Yang et al., 2024; Team, 2024b) and TinyLlama-1.1B (Zhang et al., 2024b). It includes 6 instruct-tuning variants, 11 sampling variants, 12 system-prompt variants, 7 SFT variants, 3 DPO variants, 8 LoRA variants and 8 QLoRA variants, which together cover common post-training and deployment-time model changes. The detailed information of these variants is given below.

- **Instruct**: We use the original instruction-aligned or chat models from each source family as the basic aligned variants, including Gemma-2-2B-IT, Llama-3.1-8B-Instruct, Mistral-7B-Instruct, Qwen-2.5-Instruct models, and TinyLlama-Chat.
- **Prompt** (Giray, 2023): Models reuse the weights of their corresponding instruct models but prepend a fixed system prompt before each query. We consider two prompt styles, a helpful-honest assistant prompt (*You are a helpful, harmless, and honest assistant.*) and an expert-assistant prompt (*You are an expert assistant that provides clear and accurate answers.*).
- **Sampling** (Hewitt et al., 2022): Models reuse the weights of their corresponding instruct models but change the decoding configuration. We consider two stochastic decoding settings, sampling temperature $\in \{0.3, 0.7\}$ with a nucleus sampling threshold of 0.95.
- **SFT** (Lv et al., 2024): Models are fine-tuned with full-parameter supervised fine-tuning on instruction-response data, and the self-constructed standard SFT variants are trained on Alpaca (Taori et al., 2023). This group also includes task-specialized variants such as coding, math and job-description matching models, since these are obtained through supervised task-oriented adaptation.
- **DPO** (Rafailov et al., 2023): Models are aligned with Direct Preference Optimization on paired preference data from OpenAssistant (Köpf et al., 2023). These variants test whether preference alignment changes the family-level provenance signal.
- **LoRA/QLoRA** (Hu et al., 2022; Dettmers et al., 2023): Models are fine-tuned with parameter-efficient LoRA or QLoRA adapters of rank $r = 16$ or $r = 32$, while preserving the same source-family provenance label as the base model.

We further construct an anti-provenance benchmark of 50 adversarially modified LLMs from the same six source families. It contains five attack settings, with 10 models per setting. All variants are trained with LoRA SFT (Hu et al., 2022; Lv et al., 2024) for 3 epochs using a batch size of 2 and a learning rate of 2×10^{-5}. The five settings are described below.

- **Noisy SFT (N-SFT) (Dang et al., 2025)**: Models are fine-tuned on the `ultrachat_200k_sft` dataset (Ding et al., 2023) with Laplace noise injected into the input embeddings of 3,000 training samples before training. The noise scale is set to `noise_std` $= 0.05$.
- **Single-Teacher Imitation (STI) (Gharami et al., 2025)**: Models are trained on imitation data generated by a single teacher model. Query prompts are sampled from a subset of `ultrachat_200k_sft` that is disjoint from the data used in N-SFT, and the teacher responses are generated with temperature 0.8. Each model uses 3,000 query-response pairs for LoRA SFT without embedding noise.
- **Multi-Teacher Imitation (MTI) (Gharami et al., 2025)**: Models are trained on imitation data from multiple teachers. Three teacher models each generate 1,000 query-response pairs, resulting in 3,000 mixed-supervision samples per target model, and LoRA SFT is performed without embedding noise.
- **Noisy Single-Teacher Imitation (N-STI)**: Models follow the same single-teacher imitation pipeline as STI with 3,000 generated query-response pairs, but additionally inject Laplace noise into the input embeddings before training, with `noise_std` $= 0.05$.
- **Noisy Multi-Teacher Imitation (N-MTI)**: Models follow the same multi-teacher imitation pipeline as MTI with three teachers and 1,000 samples each, and additionally inject Laplace noise into the input embeddings before training, with `noise_std` $= 0.05$.

B.4 Baseline introduction

We compare with seven black-box fingerprint baselines: HideNSeek (Iourovitski et al., 2024), DuFFin (Yan et al., 2026), ProFLingo (Jin et al., 2024), SEF (Shao et al., 2025a), MET (Gao et al., 2024), LLMmap (Pasquini et al., 2025) and ZEROPRINT (Shao et al., 2025b).

HideNSeek: HideNSeek (Iourovitski et al., 2024) employs LLMs in two roles, a query auditor and a provenance detector. The auditor first generates discriminative prompts in a free-form manner, the prompts are then submitted to the target and reference LLMs, and the detector analyzes the responses to identify the most similar reference model. The detector's feedback is further used to refine subsequent prompts, forming an iterative query-and-provenance process.

DuFFin: DuFFin (Yan et al., 2026) constructs two types of fingerprints, a trigger-pattern fingerprint encoded from response trajectories and a knowledge-level fingerprint formed from answer patterns

across various domains. Provenance is then determined by comparing the weighted similarity of the two fingerprints between the target and reference models.

ProFLingo: ProFLingo (Jin et al., 2024) is a targeted fingerprint method that optimizes prefix-based queries on a set of question-target pairs to elicit specific target responses from the reference model. The responses of the target and reference LLMs to these queries are then collected, and provenance is determined by comparing whether the target model exhibits a similarly high target response rate under the same query set.

SEF: SEF (Shao et al., 2025a) uses a manually selected query set covering reasoning, ethics, domain knowledge and code generation tasks to elicit model-specific responses. It encodes the responses with a sentence embedding model and averages the embeddings into a fingerprint, and provenance is determined by the cosine similarity between the target and reference fingerprints.

MET: MET (Gao et al., 2024) is a statistical distribution-based provenance method. It queries the target and reference LLMs with the same prompt set and treats the sampled completions as samples from the corresponding output distributions. The two distributions are compared using a kernel-based maximum mean discrepancy statistic, and provenance is determined by their statistical closeness.

LLMmap: LLMmap (Pasquini et al., 2025) uses manually selected queries to collect query-response traces from the target and reference LLMs. These traces are encoded by a lightweight transformer-based inference model into learned fingerprints. For known LLMs, it performs supervised classification, while for unseen LLMs it compares contrastively learned embeddings against known fingerprints by similarity.

ZEROPRINT: ZEROPRINT (Shao et al., 2025b) uses code-oriented queries and their semantically perturbed variants to probe the target and reference LLMs. By comparing the changes in model outputs under these controlled input perturbations, it estimates a surrogate of the local Jacobian as the fingerprint, and provenance is then performed by measuring the similarity between the Jacobian-based fingerprints of different models.

B.5 Detailed analysis of main results

Table 4: Closed-set (CS) and open-set (OS) accuracy of all methods grouped by LLM modification type on the two benchmarks. For FamilyTrace, values are averaged over the 100 query lists used for the main results.

Method	Metric	Non-adversarial modification						Anti-provenance modification				
		Instruct	Prompt	Sampling	SFT	DPO	LoRA/QLoRA	N-SFT	STI	MTI	N-STI	N-MTI
HideNSeek	CS Acc. ↑	0.500	0.417	0.636	0.571	0.667	0.562	0.400	0.600	0.400	0.600	0.600
	OS Acc. ↑	0.333	0.278	0.379	0.318	0.611	0.562	0.337	0.497	0.500	0.517	0.483
DuFFin	CS Acc. ↑	0.833	0.833	0.818	0.286	0.667	0.750	0.700	**0.900**	0.700	0.800	0.500
	OS Acc. ↑	0.617	0.806	0.803	0.519	0.667	0.750	**0.583**	0.647	0.500	0.687	0.667
ProFLingo	CS Acc. ↑	0.667	0.667	0.636	0.143	0.000	0.312	0.600	0.300	0.300	0.400	0.400
	OS Acc. ↑	0.643	0.722	0.712	0.135	0.278	0.458	0.333	0.250	0.250	0.417	0.333
SEF	CS Acc. ↑	0.833	**1.000**	**1.000**	0.571	**1.000**	0.938	0.700	0.800	0.500	0.800	0.700
	OS Acc. ↑	0.643	0.792	0.773	0.567	0.833	0.750	**0.583**	0.417	0.667	0.583	0.667
MET	CS Acc. ↑	**1.000**	0.833	0.909	0.286	0.667	0.938	0.500	0.600	0.500	0.700	0.400
	OS Acc. ↑	0.800	0.750	**0.909**	0.600	0.500	0.844	0.417	0.403	0.513	0.333	0.583
LLMmap	CS Acc. ↑	0.911	0.911	0.909	**0.962**	0.956	0.933	0.800	**0.900**	0.800	0.800	0.800
	OS Acc. ↑	0.678	0.679	0.674	0.678	0.674	0.671	0.450	0.508	0.500	0.442	0.483
ZEROPRINT	CS Acc. ↑	0.333	0.833	0.545	0.714	**1.000**	0.875	**0.900**	0.800	0.600	0.800	0.500
	OS Acc. ↑	0.429	0.722	0.515	0.540	0.889	0.740	0.573	0.590	0.587	0.750	0.583
FamilyTrace (Ours)	CS Acc. ↑	0.988	0.915	0.974	0.909	**1.000**	**1.000**	0.710	0.823	**0.839**	**0.950**	**0.950**
	OS Acc. ↑	**0.970**	**0.919**	0.692	**0.773**	**0.961**	**0.960**	0.576	**0.763**	**0.820**	**0.903**	**0.914**

Non-adversarial LLM modifications: As shown in Table 4, standard modifications do not affect provenance difficulty equally on the non-adversarial benchmark. Parameter-independent changes such as prompt-based adaptation and sampling, together with parameter-altering fine-tuning methods such as SFT (Lv et al., 2024), DPO (Rafailov et al., 2023) and LoRA (Dettmers et al., 2023; Hu et al., 2022), generally preserve enough family-level signal for strong attribution. The difficulty becomes more visible in the open-set setting, where most methods show a clear drop.

Adversarial anti-provenance LLM modifications: Adversarial anti-provenance transformations make the task substantially harder for all methods, and the degradation is especially clear in the

open-set metrics. For FamilyTrace, the overall closed-set accuracy decreases from 0.963 on the non-adversarial benchmark to 0.854 on the anti-provenance benchmark, while the open-set accuracy decreases from 0.879 to 0.795. Adding noise on top of teacher-based adaptation does not consistently produce additional degradation.

Baseline comparison: The baseline results highlight different limitations of existing provenance strategies. HideNSeek relies on LLM-based judging and iterative prompt feedback, which leads to unstable performance across modification types. ProFLingo uses optimized prefix-based queries to trigger specific target responses, but this targeted signal provides limited information under a small query budget and is less suited to family-level attribution. LLMmap and SEF are stronger baselines, since LLMmap benefits from learned trace representations and SEF uses response embeddings from selected queries. However, their open-set rejection remains limited compared with their closed-set attribution. For example, on the anti-provenance benchmark, LLMmap reaches 0.820 closed-set accuracy but only 0.477 open-set accuracy and 0.684 AUROC, while SEF reaches 0.700 closed-set accuracy with 0.583 open-set accuracy and 0.794 AUROC. This suggests that strong known-family classification does not necessarily translate into reliable unknown-family rejection.

Variation across query lists: Table 1 reports the mean of FamilyTrace over the 100 query lists used for the main results, which are generated by the same design rule. Table 5 gives the standard deviations across these lists together with the results of the default query list.

Table 5: Mean and standard deviation of FamilyTrace over the 100 query lists used for the main results, and results on the default query list.

	Non-adversarial		Anti-provenance	
Metric	Mean $\pm$ SD	Default list	Mean $\pm$ SD	Default list
CS Acc. $\uparrow$	0.963 ± 0.022	0.945	0.854 ± 0.035	0.820
OS Acc. $\uparrow$	0.879 ± 0.028	0.849	0.795 ± 0.029	0.780
TPR $\uparrow$	0.883 ± 0.108	0.773	0.765 ± 0.076	0.833
FPR $\downarrow$	0.113 ± 0.026	0.095	0.177 ± 0.033	0.222
AUROC $\uparrow$	0.950 ± 0.020	0.947	0.854 ± 0.031	0.886

Overall performance over 100 query lists: Figures 2 and 3 show the confusion matrices of FamilyTrace over the 100 query lists used for the main results. On the non-adversarial benchmark, the matrix is close to block-diagonal, with an overall closed-set accuracy of 0.963. On the anti-provenance benchmark, the family blocks remain clear, with an overall closed-set accuracy of 0.854.

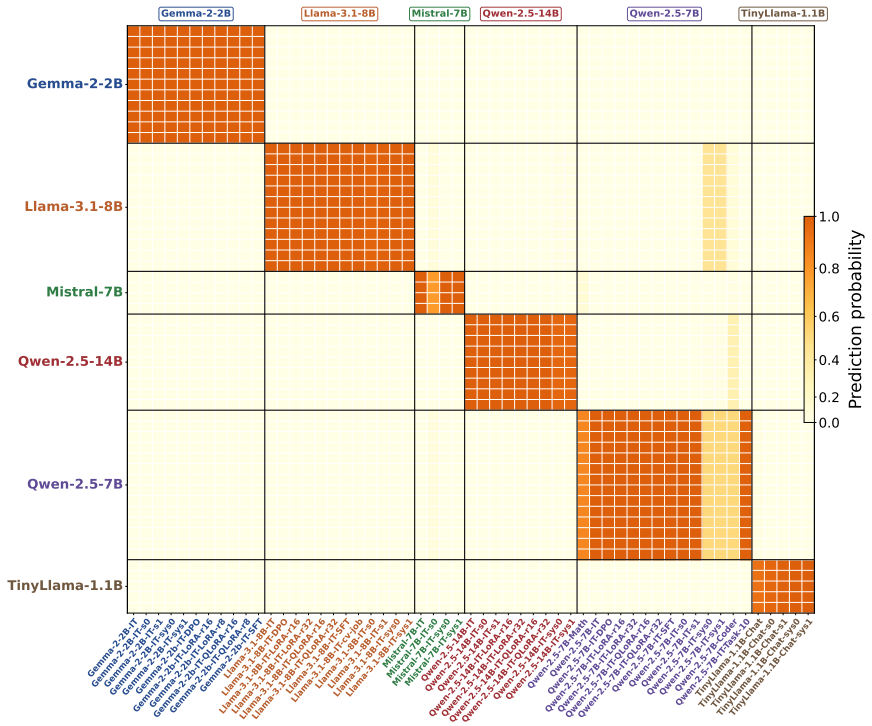


Figure 2: Confusion matrix of FamilyTrace on the non-adversarial modification benchmark over the 100 query lists used for the main results. Each column is one model, each row block is a predicted family, and the color shows how often the model is attributed to that family.

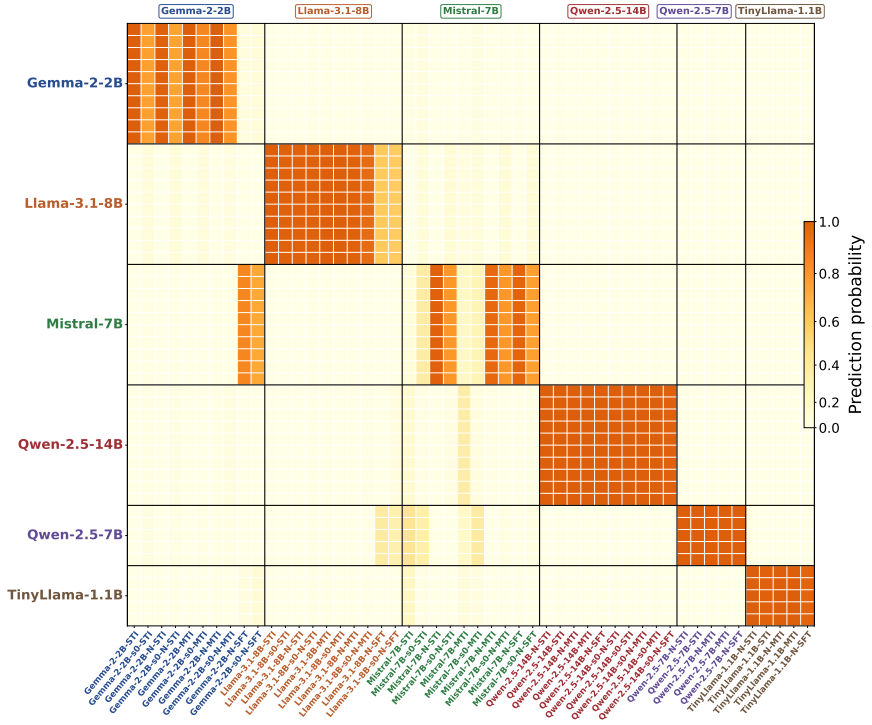


Figure 3: Confusion matrix of FamilyTrace on the anti-provenance modification benchmark over the 100 query lists used for the main results. Each column is one model, each row block is a predicted family, and the color shows how often the model is attributed to that family.

B.6 DISCUSSION ON LLM FAMILY

Table 6: Family-wise closed-set (CS) and open-set (OS) accuracy of all methods on the two benchmarks. For FamilyTrace, we report the mean over the 100 query lists used for the main results.

(a) Non-adversarial modification

Method	Metric	Gemma-2-2B	Llama-3.1-8B	Mistral-7B	Qwen-2.5-14B	Qwen-2.5-7B	TinyLlama-1.1B
HideNSeek	CS Acc. ↑	0.545	0.500	0.250	0.556	0.571	0.800
	OS Acc. ↑	0.436	0.400	0.400	0.364	0.400	0.491
DuFFin	CS Acc. ↑	0.909	0.917	0.750	0.556	0.571	0.600
	OS Acc. ↑	0.764	0.746	0.673	0.746	0.746	0.746
ProFLingo	CS Acc. ↑	0.273	0.500	0.750	0.667	0.286	0.600
	OS Acc. ↑	0.582	0.527	0.546	0.436	0.655	0.527
SEF	CS Acc. ↑	0.909	0.917	0.750	0.889	0.929	1.000
	OS Acc. ↑	0.909	0.907	0.941	0.435	0.487	0.940
MET	CS Acc. ↑	0.818	0.750	1.000	0.889	0.786	0.800
	OS Acc. ↑	0.709	0.709	0.673	0.709	0.782	0.673
LLMmap	CS Acc. ↑	0.873	0.994	0.867	0.911	0.971	0.827
	OS Acc. ↑	0.762	0.731	0.156	0.812	0.720	0.869
ZEROPRINT	CS Acc. ↑	0.909	0.667	0.750	0.778	0.643	0.600
	OS Acc. ↑	0.582	0.618	0.673	0.600	0.600	0.673
FamilyTrace (Ours)	CS Acc. ↑	1.000	0.998	0.945	0.993	0.886	0.978
	OS Acc. ↑	0.898	0.878	0.806	0.926	0.856	0.878

(b) Anti-provenance modification

Method	Metric	Gemma-2-2B	Llama-3.1-8B	Mistral-7B	Qwen-2.5-14B	Qwen-2.5-7B	TinyLlama-1.1B
HideNSeek	CS Acc. ↑	0.700	0.500	0.700	0.300	0.200	0.600
	OS Acc. ↑	0.380	0.420	0.380	0.460	0.500	0.460
DuFFin	CS Acc. ↑	0.600	1.000	0.800	0.600	0.400	0.800
	OS Acc. ↑	0.620	0.540	0.580	0.620	0.700	0.640
ProFLingo	CS Acc. ↑	0.400	0.600	0.300	0.300	0.400	0.400
	OS Acc. ↑	0.180	0.260	0.380	0.380	0.360	0.340
SEF	CS Acc. ↑	0.700	0.500	0.800	0.800	0.800	0.600
	OS Acc. ↑	0.560	0.600	0.540	0.540	0.620	0.640
MET	CS Acc. ↑	0.600	0.500	0.600	0.500	0.600	0.400
	OS Acc. ↑	0.340	0.440	0.440	0.500	0.480	0.500
LLMmap	CS Acc. ↑	0.750	0.925	0.750	0.725	0.900	1.000
	OS Acc. ↑	0.825	0.230	0.725	0.200	0.835	0.220
ZEROPRINT	CS Acc. ↑	0.700	0.700	0.800	0.700	0.800	0.600
	OS Acc. ↑	0.600	0.600	0.580	0.600	0.640	0.680
FamilyTrace (Ours)	CS Acc. ↑	0.736	0.910	0.630	0.999	1.000	0.994
	OS Acc. ↑	0.766	0.891	0.690	0.781	0.846	0.848

Family-wise accuracy: Table 6 shows that provenance difficulty varies across LLM families and becomes more pronounced in the open-set setting. On the non-adversarial benchmark, several baselines achieve strong closed-set accuracy on some families but their open-set accuracy drops noticeably. For example, LLMmap obtains 0.994 closed-set accuracy on Llama-3.1-8B and 0.971 on Qwen-2.5-7B, while the corresponding open-set accuracies drop to 0.731 and 0.720. SEF shows a similar gap on Qwen-2.5-14B, with 0.889 closed-set accuracy but only 0.435 open-set accuracy. This gap is further amplified on the anti-provenance benchmark, where LLMmap reaches 0.925 and 1.000

closed-set accuracy on Llama-3.1-8B and TinyLlama-1.1B, but its open-set accuracies decrease to 0.230 and 0.220. These results suggest that some baselines can classify known-family models well but have greater difficulty rejecting unknown-family models. In comparison, FamilyTrace achieves more balanced family-wise open-set performance, with OS accuracy ranging from 0.806 to 0.926 on the non-adversarial benchmark and from 0.690 to 0.891 on the anti-provenance benchmark.

Family separability visualization: Figures 4 and 6 qualitatively compare the fingerprint spaces produced by different methods. On the non-adversarial benchmark, several baselines form partial family clusters, but their regions are often loose or overlapping, especially for response-level methods such as HideNSeek and ProFLingo. SEF and LLMmap show more structured distributions, but some families still exhibit limited separation. Under adversarial anti-provenance transformations, the clusters become more entangled for most methods, which is consistent with the larger open-set drops observed in Table 6. In comparison, FamilyTrace yields relatively clearer family-level groupings on both benchmarks, and the detailed 2D t-SNE visualizations of FamilyTrace fingerprints are shown in Figures 5 and 7. Although the anti-provenance benchmark introduces more overlap and larger intra-family spread, most main family structures remain visible. These visualizations provide an intuitive view of fingerprint separability and suggest that FamilyTrace forms more distinguishable family-level structures by capturing associative priors and their contextual revision.

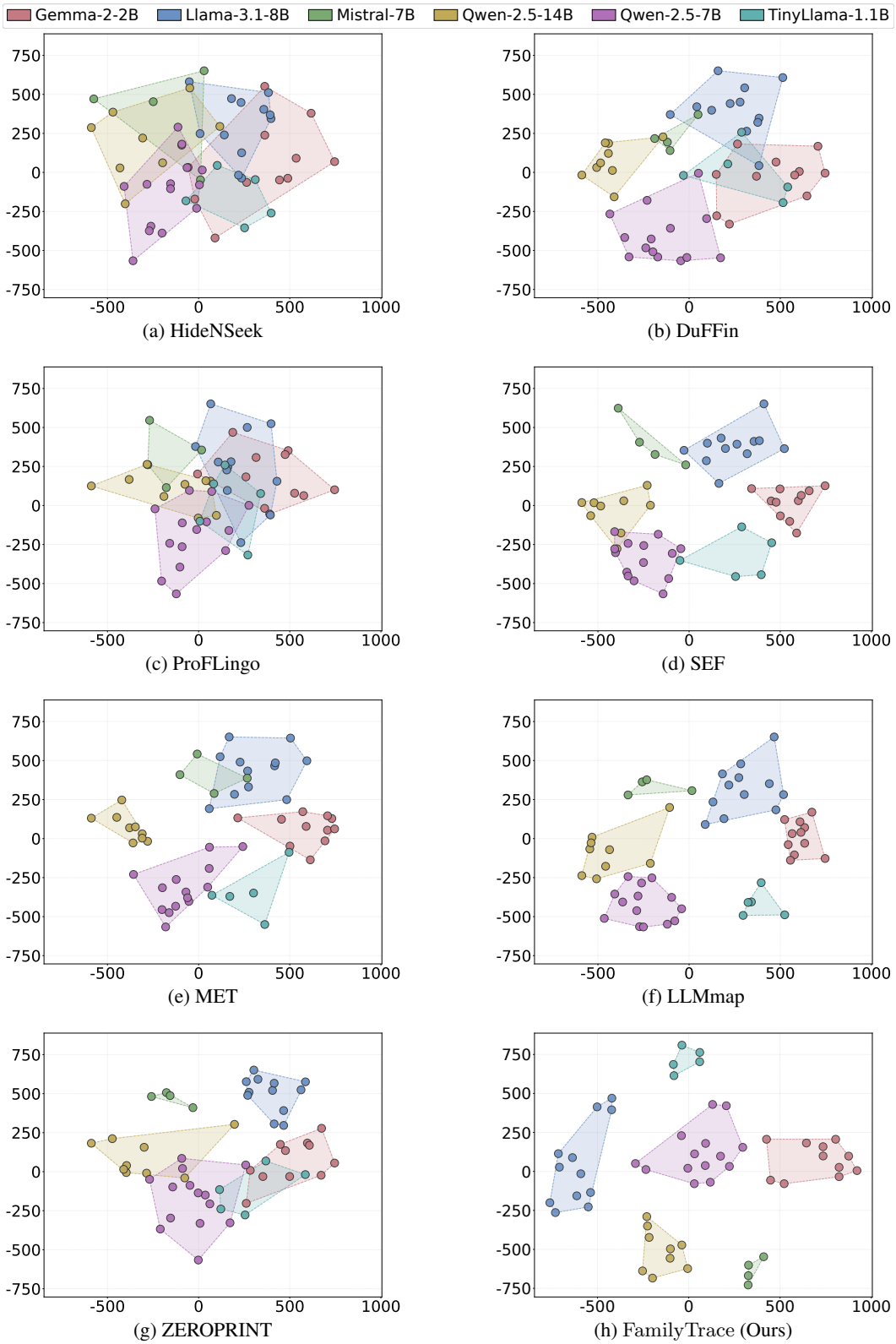


Figure 4: 2D t-SNE projection of fingerprints produced by different provenance methods on the non-adversarial modification benchmark. Each shaded area denotes one LLM family.

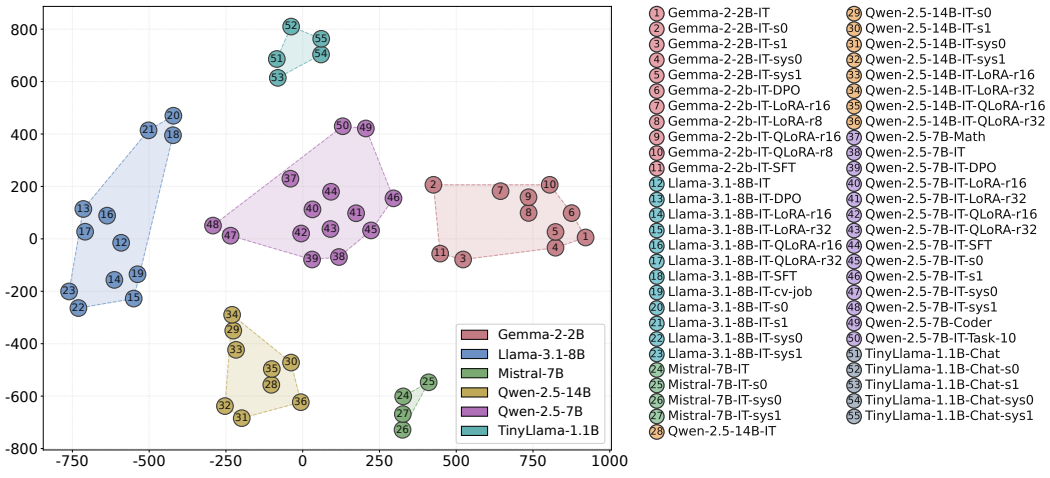


Figure 5: 2D t-SNE projection of FamilyTrace fingerprints with the corresponding model indices on the non-adversarial modification benchmark.

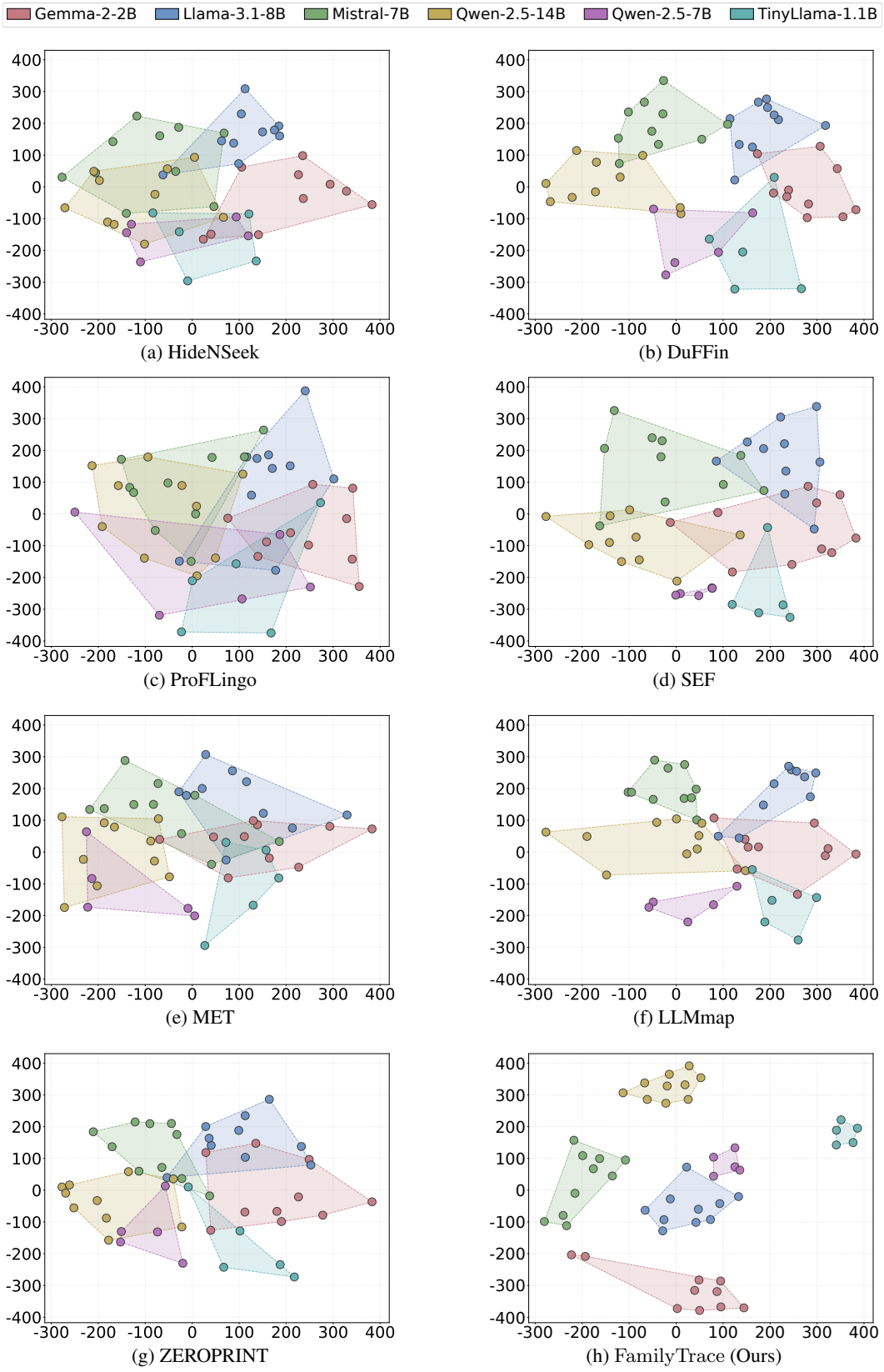
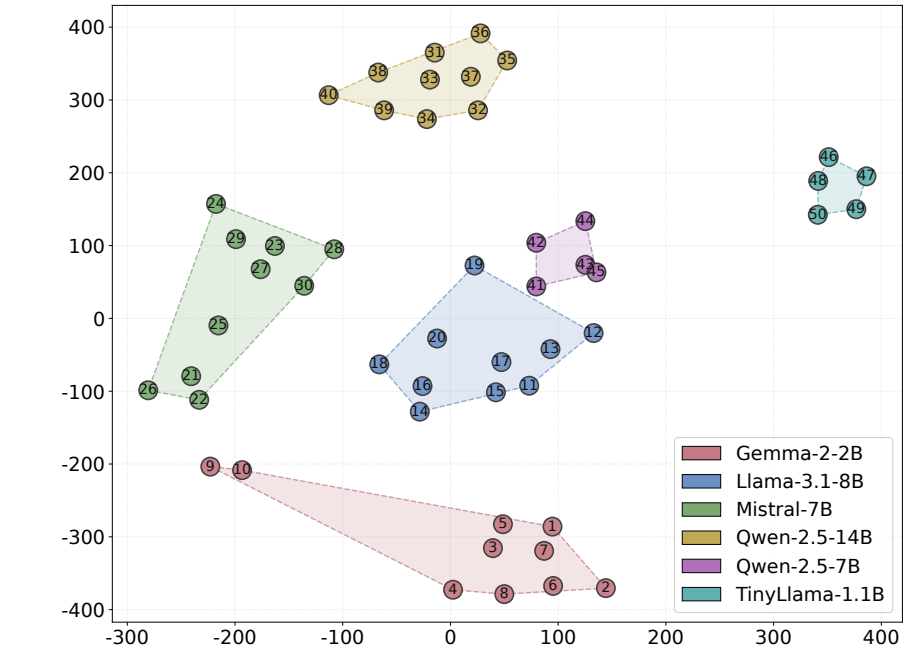


Figure 6: 2D t-SNE projection of fingerprints produced by different provenance methods on the anti-provenance modification benchmark. Each shaded area denotes one LLM family.



1 Gemma-2-2B-IT + T(Qwen-2.5-7B-IT)
2 Gemma-2-2B-s0 + T(Llama-3.1-8B-sys0)
3 Gemma-2-2B-IT + T(Qwen-2.5-7B-IT) + Noise
4 Gemma-2-2B-s0 + T(Llama-3.1-8B-sys0) + Noise
5 Gemma-2-2B-IT + MT (Llama-3.1-8B-IT/Qwen-2.5-7B-IT/TinyLlama-1.1B-Chat)
6 Gemma-2-2B-s0 + MT (Llama-3.1-8B-s0/Qwen-2.5-14B-s0/Mistral-7B-s0)
7 Gemma-2-2B-IT + MT + Noise (Llama-3.1-8B-IT/Qwen-2.5-7B-IT/TinyLlama-1.1B-Chat)
8 Gemma-2-2B-s0 + MT + Noise (Llama-3.1-8B-s0/Qwen-2.5-14B-s0/Mistral-7B-s0)
9 Gemma-2-2B-IT + Noise
10 Gemma-2-2B-s0 + Noise
11 Llama-3.1-8B-IT + T(Gemma-2-2B-IT)
12 Llama-3.1-8B-s0 + T(Qwen-2.5-14B-s1)
13 Llama-3.1-8B-IT + T(Gemma-2-2B-IT) + Noise
14 Llama-3.1-8B-s0 + T(Qwen-2.5-14B-s1) + Noise
15 Llama-3.1-8B-IT + MT (Gemma-2-2B-IT/Qwen-2.5-14B-IT/Mistral-7B-IT)
16 Llama-3.1-8B-s0 + MT (Gemma-2-2B-s0/Qwen-2.5-7B-s0/TinyLlama-1.1B-s0)
17 Llama-3.1-8B-IT + MT + Noise (Gemma-2-2B-IT/Qwen-2.5-14B-IT/Mistral-7B-IT)
18 Llama-3.1-8B-s0 + MT + Noise (Gemma-2-2B-s0/Qwen-2.5-7B-s0/TinyLlama-1.1B-s0)
19 Llama-3.1-8B-IT + Noise
20 Llama-3.1-8B-s0 + Noise
21 Mistral-7B-IT + T(Llama-3.1-8B-IT)
22 Mistral-7B-s0 + T(Gemma-2-2B-sys1)
23 Mistral-7B-IT + T(Llama-3.1-8B-IT) + Noise
24 Mistral-7B-s0 + T(Gemma-2-2B-sys1) + Noise
25 Mistral-7B-IT + MT (Llama-3.1-8B-IT/Qwen-2.5-7B-IT/Gemma-2-2B-IT)
26 Mistral-7B-s0 + MT (Gemma-2-2B-sys1/Llama-3.1-8B-sys0/Qwen-2.5-14B-sys0)
27 Mistral-7B-IT + MT + Noise (Llama-3.1-8B-IT/Qwen-2.5-7B-IT/Gemma-2-2B-IT)
28 Mistral-7B-s0 + MT + Noise (Gemma-2-2B-sys1/Llama-3.1-8B-sys0/Qwen-2.5-14B-sys0)
29 Mistral-7B-IT + Noise
30 Mistral-7B-s0 + Noise
31 Qwen-2.5-14B-IT + T(TinyLlama-1.1B-Chat) + Noise
32 Qwen-2.5-14B-s0 + T(Qwen-2.5-7B-sys0) + Noise
33 Qwen-2.5-14B-IT + T(TinyLlama-1.1B-Chat)
34 Qwen-2.5-14B-s0 + T(Qwen-2.5-7B-sys0)
35 Qwen-2.5-14B-IT + MT + Noise (Gemma-2-2B-IT/Llama-3.1-8B-IT/TinyLlama-1.1B-Chat)
36 Qwen-2.5-14B-s0 + MT + Noise (Qwen-2.5-7B-sys0/Mistral-7B-sys1/TinyLlama-1.1B-sys0)
37 Qwen-2.5-14B-IT + MT (Gemma-2-2B-IT/Llama-3.1-8B-IT/TinyLlama-1.1B-Chat)
38 Qwen-2.5-14B-s0 + MT (Qwen-2.5-7B-sys0/Mistral-7B-sys1/TinyLlama-1.1B-sys0)
39 Qwen-2.5-14B-IT + Noise
40 Qwen-2.5-14B-s0 + Noise
41 Qwen-2.5-7B-IT + T(Mistral-7B-IT) + Noise
42 Qwen-2.5-7B-IT + T(Mistral-7B-IT)
43 Qwen-2.5-7B-IT + MT + Noise (Qwen-2.5-14B-IT/Mistral-7B-IT/TinyLlama-1.1B-Chat)
44 Qwen-2.5-7B-IT + MT (Qwen-2.5-14B-IT/Mistral-7B-IT/TinyLlama-1.1B-Chat)
45 Qwen-2.5-7B-IT + Noise
46 TinyLlama-1.1B-Chat + T(Qwen-2.5-14B-IT) + Noise
47 TinyLlama-1.1B-Chat + T(Qwen-2.5-14B-IT)
48 TinyLlama-1.1B-Chat + MT + Noise (Qwen-2.5-14B-IT/Qwen-2.5-7B-IT/Llama-3.1-8B-IT)
49 TinyLlama-1.1B-Chat + MT (Qwen-2.5-14B-IT/Qwen-2.5-7B-IT/Llama-3.1-8B-IT)
50 TinyLlama-1.1B-Chat + Noise

Figure 7: 2D t-SNE projection of FamilyTrace fingerprints with the corresponding model indices on the anti-provenance modification benchmark.

B.7 Computational resources and runtime details

All open-weight inference, fingerprint extraction and benchmark evaluation experiments are run on one NVIDIA H100 NVL GPU worker with 95,830 MiB GPU memory and CUDA 12.9. Table 7 reports end-to-end provenance runtime, including model loading, query execution and method-specific fingerprint or provenance computation. The default FamilyTrace query list uses 6 anchors and 4 stages (24 queries) per target model. The 100 query-list variant analysis in Appendix B.12 repeats the same single-list pipeline independently for each query list, requiring 45.2 GPU-hours on the non-adversarial benchmark and 53.5 GPU-hours on the anti-provenance benchmark.

Table 7: Total provenance runtime in seconds of all methods on the two benchmarks. The runtime includes LLM loading, query and algorithm execution.

(a) Non-adversarial modification

Method	HideNSeek	DuFFin	ProFLingo	SEF	MET	LLMmap	ZEROPRINT	FamilyTrace (Ours)
Runtime ↓	21,480	2,908	1,035	2,385	2,496	3,862	2,498	813

(b) Anti-provenance modification

Method	HideNSeek	DuFFin	ProFLingo	SEF	MET	LLMmap	ZEROPRINT	FamilyTrace (Ours)
Runtime ↓	26,083	2,276	1,297	2,633	1,900	3,313	3,001	963

B.8 Number of anchors and stages

Tables 8 and 9 report FamilyTrace with different numbers of anchors and stages. The anchor results are averaged over 30 query lists generated with 7 anchors and 4 stages, and the stage results over 30 query lists generated with 6 anchors and 10 stages.

Table 8: Provenance metrics of FamilyTrace with different numbers of anchors and 4 stages, evaluated on 30 query lists generated with 7 anchors.

(a) Non-adversarial modification

Anchors	CS Acc. ↑	OS Acc. ↑	TPR ↑	FPR ↓	AUROC ↑
3	0.882	0.788	0.514	0.087	0.853
4	0.915	0.793	0.485	0.084	0.876
5	0.898	0.786	0.499	0.091	0.868
6	0.926	0.821	0.578	0.087	0.881
7	0.939	0.838	0.625	0.079	0.892

(b) Anti-provenance modification

Anchors	CS Acc. ↑	OS Acc. ↑	TPR ↑	FPR ↓	AUROC ↑
3	0.836	0.756	0.589	0.138	0.703
4	0.861	0.769	0.588	0.132	0.717
5	0.840	0.745	0.555	0.154	0.743
6	0.888	0.778	0.567	0.135	0.779
7	0.901	0.786	0.568	0.128	0.799

Table 9: Provenance metrics of FamilyTrace with 6 anchors and different numbers of stages, evaluated on 30 query lists generated with 10 stages (mean).

(a) Non-adversarial modification

Stages	CS Acc. ↑	OS Acc. ↑	TPR ↑	FPR ↓	AUROC ↑
1	0.898	0.812	0.691	0.107	0.905
2	0.905	0.818	0.692	0.096	0.906
3	0.940	0.846	0.757	0.101	0.917
4	0.947	0.847	0.755	0.098	0.915
5	0.947	0.834	0.719	0.105	0.913
6	0.947	0.830	0.712	0.107	0.916
7	0.948	0.846	0.735	0.088	0.928
8	0.945	0.847	0.725	0.084	0.931
9	0.942	0.849	0.712	0.078	0.932
10	0.943	0.843	0.675	0.072	0.928

(b) Anti-provenance modification

Stages	CS Acc. ↑	OS Acc. ↑	TPR ↑	FPR ↓	AUROC ↑
1	0.735	0.666	0.477	0.186	0.739
2	0.773	0.694	0.490	0.174	0.746
3	0.836	0.754	0.524	0.157	0.783
4	0.857	0.768	0.527	0.143	0.793
5	0.859	0.769	0.531	0.146	0.799
6	0.872	0.771	0.548	0.149	0.808
7	0.887	0.794	0.573	0.131	0.825
8	0.900	0.803	0.584	0.132	0.834
9	0.903	0.808	0.593	0.131	0.835
10	0.904	0.813	0.601	0.128	0.836

B.9 Discussion on threshold τ_z on open-set setting

This section provides a detailed analysis of the open-set threshold τ_z. We discuss how τ_z controls the trade-off between the rejection of unknown-family LLMs (TPR) and the false rejection of known-family LLMs (FPR), and we analyze the corresponding ROC behavior on both the non-adversarial benchmark and the anti-provenance benchmark.

Threshold trade-off: Figure 8 and Table 10 show that τ_z acts as a direct control for open-set provenance. As τ_z increases, the rejection rule becomes stricter, so more unknown-family LLMs are correctly rejected. On the non-adversarial benchmark, TPR increases from 0.736 at $\tau_z = -4.0$ to 0.999 at $\tau_z = 0.0$, but FPR also increases from 0.049 to 0.321, meaning that more known-family models are incorrectly rejected. A similar trade-off appears on the anti-provenance benchmark, where TPR increases from 0.659 at $\tau_z = -4.0$ to 0.820 at $\tau_z = -1.0$ while FPR increases from 0.125 to 0.244. The OS accuracy curve reflects the balance between these two effects, and the best OS accuracy appears at $\tau_z = -3.5$ on both benchmarks. Thresholds in $[-4, -2]$ form a practical operating range, where OS accuracy changes by less than 0.03 and unknown rejection improves while known-family false rejection remains relatively controlled.

ROC behavior: Figure 9 further shows the ROC behavior obtained by varying τ_z, where each operating point corresponds to a different threshold and moving toward a larger τ_z generally increases both TPR and FPR. On the non-adversarial benchmark, the ROC curve stays close to the upper-left region and achieves a high AUROC, indicating strong separability between known-family and unknown-family LLMs. The anti-provenance benchmark shows a more difficult pattern with lower TPR in the low-FPR region, which is consistent with the stronger confusion introduced by anti-provenance transformations. Overall, the threshold and ROC analyses show that FamilyTrace does not rely on a single fixed rejection point, and its z-score based open-set scheme provides a tunable decision rule that allows the auditor to choose a conservative or aggressive rejection policy depending on the acceptable false-rejection risk.

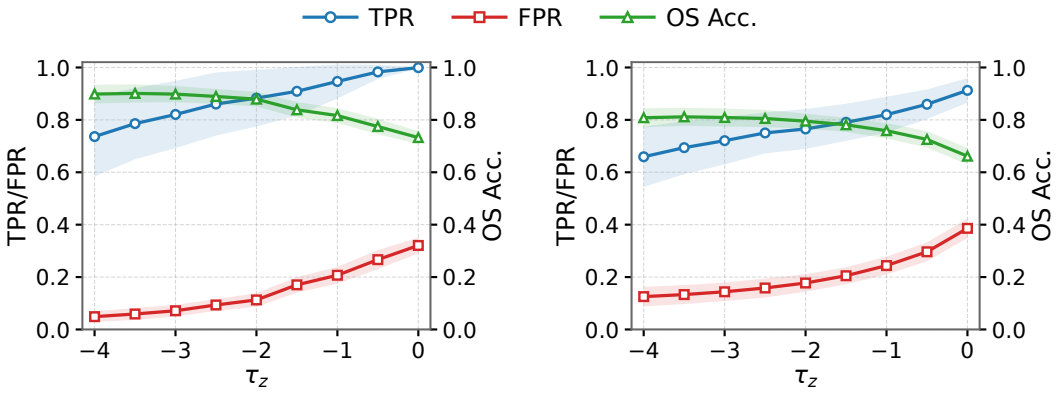
Table 10: Provenance metrics of FamilyTrace under different open-set rejection thresholds τ_z, averaged over the 100 query lists.

(a) Non-adversarial modification

Metric	−4.0	−3.5	−3.0	−2.5	−2.0	−1.5	−1.0	−0.5	0.0
TPR ↑	0.736	0.786	0.821	0.860	0.883	0.909	0.947	0.983	0.999
FPR ↓	0.049	0.059	0.071	0.093	0.113	0.170	0.207	0.266	0.321
OS Acc. ↑	0.899	0.901	0.898	0.889	0.879	0.838	0.816	0.775	0.732

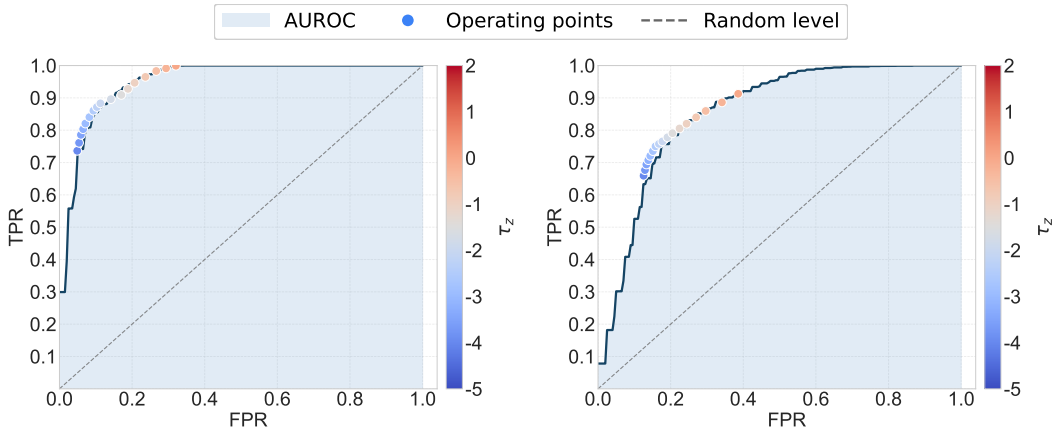
(b) Anti-provenance modification

Metric	−4.0	−3.5	−3.0	−2.5	−2.0	−1.5	−1.0	−0.5	0.0
TPR ↑	0.659	0.694	0.721	0.750	0.765	0.791	0.820	0.860	0.913
FPR ↓	0.125	0.133	0.144	0.158	0.177	0.205	0.244	0.296	0.386
OS Acc. ↑	0.808	0.812	0.809	0.805	0.795	0.781	0.759	0.725	0.661



(a) Non-adversarial modification (b) Anti-provenance modification

Figure 8: TPR, FPR and OS Acc. of FamilyTrace under different rejection thresholds τ_z on the two benchmarks, averaged over the 100 query lists used for the main results. Shaded bands show one standard deviation.



(a) Non-adversarial modification (b) Anti-provenance modification

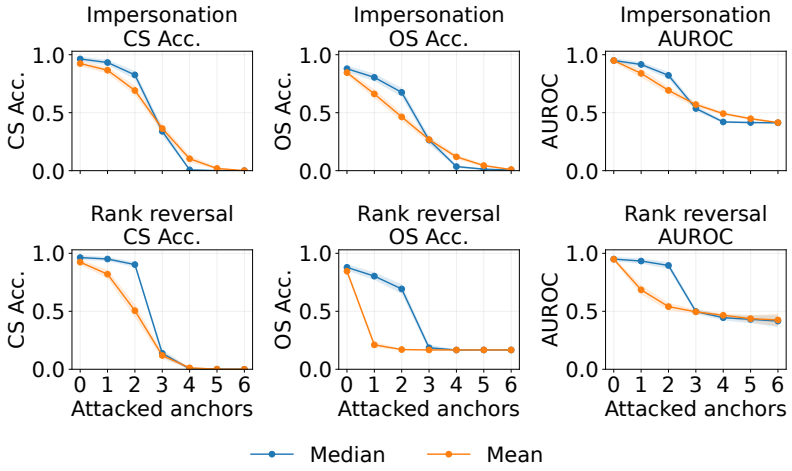
Figure 9: ROC curves of the FamilyTrace open-set scheme on the two benchmarks, averaged over the 100 query lists used for the main results and the leave-one-family-out folds. The shaded area shows the area under the curve, and the colored points are the operating points under different rejection thresholds τ_z.

B.10 SIMULATED ANCHOR TAMPERING

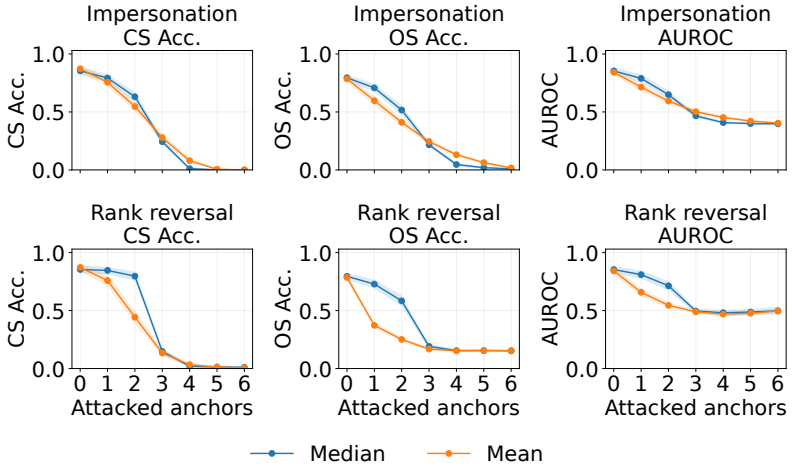
To test the robustness of the median over anchors, we simulate an attacker who controls the responses of a target model to k of the six anchors. Under impersonation, the rankings of these anchors at every stage are replaced by the rankings implied by the family profile of another, randomly chosen family. Under reversal, they are replaced by the reverse of the target's own rankings. The reference library is left unchanged. For every target, $k \in \{0, \ldots, 6\}$ and query list, we repeat the attack with five random choices of anchors and aggregate the anchor similarities either by the median or by the mean. Table 11 reports closed-set accuracy, and Figure 10 further shows open-set accuracy and AUROC. With at most two tampered anchors, the median keeps accuracy much closer to the value without tampering than the mean, and once three or more anchors are tampered with, both aggregations fail.

Table 11: Closed-set accuracy under simulated tampering of k anchors, averaged over the 100 query lists and five random choices of anchors.

Benchmark	Attack	Aggregation	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
Non-adversarial	Impersonation	Median	0.963	0.933	0.826	0.339	0.007
		Mean	0.924	0.866	0.691	0.363	0.103
	Reversal	Median	0.963	0.951	0.903	0.138	0.004
		Mean	0.924	0.820	0.505	0.119	0.012
Anti-provenance	Impersonation	Median	0.854	0.793	0.631	0.244	0.011
		Mean	0.872	0.757	0.547	0.279	0.079
	Reversal	Median	0.854	0.846	0.797	0.148	0.019
		Mean	0.872	0.759	0.443	0.134	0.033



(a) Non-adversarial modification



(b) Anti-provenance modification

Figure 10: Closed-set (CS) accuracy, open-set (OS) accuracy and AUROC of FamilyTrace when k anchors are tampered with, using the median or the mean over anchors. In each panel, the top row shows impersonation and the bottom row shows rank reversal. Lines show the mean over the 100 query lists.

B.11 FAMILIES OF DIFFERENT SIZES FROM THE SAME SERIES

We construct five Qwen-2.5 families of 0.5B, 3B, 7B, 14B and 32B, each containing 10 modified variants of its base model. Table 12 reports the provenance metrics of FamilyTrace for each family and over all families.

Table 12: Provenance metrics of FamilyTrace across five Qwen-2.5 base-size families.

Size Family	CS Acc. ↑	OS Acc. ↑	TPR ↑	FPR ↓	AUROC ↑
Qwen-2.5-0.5B	1.000	0.900	1.000	0.125	0.953
Qwen-2.5-3B	0.800	0.920	1.000	0.100	0.985
Qwen-2.5-7B	0.900	0.900	1.000	0.125	1.000
Qwen-2.5-14B	1.000	0.920	1.000	0.100	0.983
Qwen-2.5-32B	1.000	0.700	0.800	0.125	0.743
Overall	0.940	0.868	0.960	0.115	0.933

B.12 DISCUSSION ON QUERY LIST VARIANTS OF FamilyTrace

This section provides a detailed analysis of the 100 query-list variants used to evaluate the generalization of FamilyTrace. All variants are generated using the GPT-5.4 (Singh et al., 2025) API model and preserve the same core structure of symmetric anchor groups, stage-wise semantic progression, 6 anchors and 4 stages, while changing the surface realization of the query list. The main results and the analysis below use these 100 lists.

Comparison across query-list variant types: The 100 query lists used for the main results are grouped into three controlled categories:

- **Paraphrasing**: changes the surface wording of the query sentences while preserving the same anchors, candidates and stage logic.
- **Anchor set variation**: replaces the anchor words but keeps the semantic-symmetric grouping principle and the close, intermediate and far candidate relations.
- **Instruction variation**: changes the ranking instruction or output-format wording while keeping the underlying task unchanged.

Table 13 and Figure 11 show that the three categories perform similarly on the same benchmark. On the non-adversarial benchmark, all categories achieve similar closed-set accuracy, open-set accuracy and AUROC, with CS accuracy above 0.950 and OS accuracy above 0.870, and the boxplots show comparable accuracy distributions across the three categories. On the anti-provenance benchmark, the three categories show larger variance, which is consistent with the harder nature of anti-provenance transformations, but their average performance remains close, with CS accuracy around 0.850 and OS accuracy around 0.800. These results suggest that FamilyTrace has reasonable query-list generalization as long as the core anchor-stage design is preserved, rather than depending on a single handcrafted wording or a fixed anchor vocabulary. Examples of the three variant types are shown below.

Paraphrasing variant for anchor *bank*

Instruction (identical for all stages):

```
Read the sentence and rank the other five candidate words from most to
least related in meaning to the anchor.
Output one line with rank of options only in the format:
X > X > X > X > X
```

Candidates: *stream, shore, account, capital, channel*

Stage 1: They paused beside the bank and glanced around before continuing.
Stage 2: They paused beside the bank, where the land dipped softly and the air was moist.
Stage 3: They paused beside the bank, where the land dipped softly and water passed through a narrow flow.
Stage 4: They paused beside the bank, where the land dipped softly and water passed through a narrow flow below the shore.

Anchor set variation for anchor *spring*

Instruction (identical for all stages):
```
Read the sentence and rank the other five candidate words from most to
least related in meaning to the anchor.
Output one line with rank of options only in the format:
X > X > X > X > X
```
Candidates: *stream, valley, season, bloom, coil*

Stage 1: They walked toward the spring after noticing it nearby.
Stage 2: They walked toward the spring after hearing a faint sound nearby.
Stage 3: They walked toward the spring after hearing water rise from the ground.
Stage 4: They walked toward the spring after hearing water rise from the ground and flow through the valley into a narrow stream.

Instruction variation for anchor *bank*

Instruction (identical for all stages):
```
Given the sentence, compare the contextual meaning of the anchor with
each candidate word and order the candidates from closest to farthest
in semantic relation.
Return only a single ranked chain of options, such as:
X > X > X > X > X
```
Candidates: *stream, shore, account, capital, channel*

Stage 1: They stopped near the bank and looked around before moving on.
Stage 2: They stopped near the bank, where the ground sloped gently and the air felt damp.
Stage 3: They stopped near the bank, where the ground sloped gently and the water moved past in a narrow current.
Stage 4: They stopped near the bank, where the ground sloped gently and the water moved past in a narrow current below the shore.

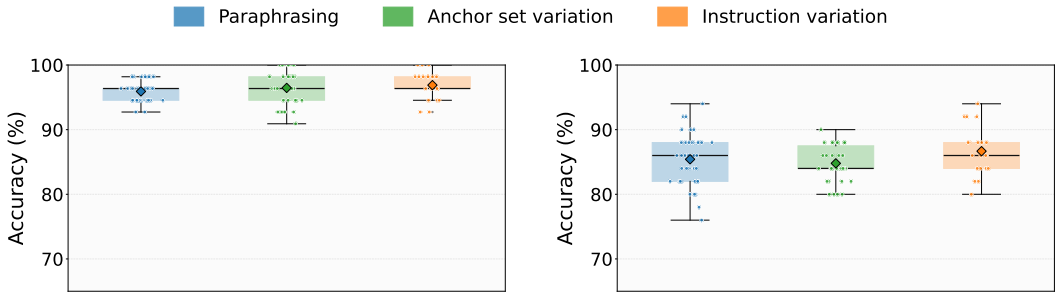
Table 13: Provenance metrics of FamilyTrace averaged within each query-list variant category on the two benchmarks.

(a) Non-adversarial modification

Category	CS Acc. ↑	OS Acc. ↑	TPR ↑	FPR ↓	AUROC ↑
Paraphrasing	0.959	0.883	0.908	0.114	0.953
Anchor set variation	0.969	0.875	0.854	0.111	0.947
Instruction variation	0.974	0.884	0.878	0.107	0.951
Overall	0.966	0.880	0.881	0.111	0.950

(b) Anti-provenance modification

Category	CS Acc. ↑	OS Acc. ↑	TPR ↑	FPR ↓	AUROC ↑
Paraphrasing	0.853	0.796	0.758	0.176	0.850
Anchor set variation	0.846	0.795	0.755	0.173	0.853
Instruction variation	0.865	0.804	0.762	0.175	0.859
Overall	0.853	0.797	0.758	0.175	0.853



(a) CS Acc. on non-adversarial modification

(b) CS Acc. on anti-provenance modification

(c) OS Acc. on non-adversarial modification

(d) OS Acc. on anti-provenance modification

Figure 11: Closed-set (CS) and open-set (OS) accuracy distribution of FamilyTrace across the three query-list variant categories on the two benchmarks, over the 100 query lists used for the main results. Each point is one query list, and the diamonds show the category means.

B.13 Effect of the query design

We compare the anchor-stage design with two alternative ways of building a list of 24 ranking queries. For random words, the six anchors of each list are drawn uniformly without replacement from the 500 most frequent common nouns of the Brown corpus, the candidates of each anchor are the other five anchors, and each anchor receives four progressively detailed sentences in the same instruction and output format as the default query list. For common-sense orderings, each list contains six unambiguous everyday nouns, each with its own five candidates chosen to follow a clear gradient from a closely related item of the same kind to an unrelated item, and each anchor receives four equally informative sentences without any progression. Both kinds of lists are generated with GPT-5.4 under fixed prompts, and no list is filtered based on the resulting accuracy. Table 14 compares 10 lists of each type with 10 lists of our design that were fixed in advance, together with the mean over the 100 lists used for the main results.

Table 14: Closed-set and open-set accuracy of FamilyTrace with different query designs, as mean ± standard deviation over query lists.

		Non-adversarial		Anti-provenance	
Query design	Lists	CS Acc. ↑	OS Acc. ↑	CS Acc. ↑	OS Acc. ↑
Random words	10	0.722 ± 0.129	0.646 ± 0.111	0.648 ± 0.163	0.597 ± 0.138
Common-sense orderings	10	0.844 ± 0.042	0.743 ± 0.053	0.780 ± 0.038	0.705 ± 0.053
FamilyTrace (ours)	10	0.960 ± 0.024	0.870 ± 0.029	0.832 ± 0.040	0.782 ± 0.033
FamilyTrace (ours)	100	0.963 ± 0.022	0.879 ± 0.028	0.854 ± 0.035	0.795 ± 0.029

B.14 Base model versus fine-tuning data

We fine-tune three base checkpoints, Llama-3.1-8B, Mistral-7B-v0.1 and Qwen-2.5-14B, on the same three datasets, Alpaca (Taori et al., 2023), UltraChat (Ding et al., 2023) and MetaMathQA (Yu et al., 2024), with three random seeds each, which gives 27 models. Each run uses 3,000 examples and

trains LoRA adapters on the query and value projections for 3 epochs (rank 8, $\alpha = 16$, dropout 0.05, learning rate 2×10^{-5}, effective batch size 4, maximum length 1024). The training text is taken directly from each dataset without a chat template, so the three base models are trained on identical text. All 27 models are queried with one query list generated by the design rule of Section 4.2, using greedy decoding with at most 48 new tokens. The reference library consists of the 27 models themselves, grouped by base model, and the fingerprint and similarity are the same as in the main experiments.

Table 15 reports two analyses. First, we compare the similarity between models that share a base model but were fine-tuned on different datasets with the similarity between models that were fine-tuned on the same dataset but come from different base models. The mean similarity is 0.241 in the first group and -0.005 in the second. We take the difference between the two group means, averaged over the 27 models, as the test statistic and compare it with its values under 100,000 random permutations of the base labels. The observed value of 0.246 is exceeded once when the labels are permuted with nine models per base, giving a p-value of 2×10^{-5}, and never when they are permuted within each dataset, giving a p-value of at most 10^{-5}.

Second, we measure closed-set attribution to the base family under three rules for building the reference library. When only the target is removed, 26 of the 27 models are attributed correctly. When the other two seeds of the same configuration are also removed, 15 of the 27 models are attributed correctly. When all models fine-tuned on the target's dataset are removed from every family, 18 of the 27 models are attributed correctly, compared with 9 expected by chance. Qwen-2.5-14B is attributed correctly under all three rules.

A two-way decomposition of the variance of the profiles assigns 34.3% to the base model, 18.0% to the dataset, 36.7% to their interaction and 11.1% to differences between seeds. Fine-tuning thus changes the rankings substantially and in a way that depends on the base model, while the base model remains the largest single factor.

Table 15: Base model versus fine-tuning data for 27 models fine-tuned from three base models on three datasets with three seeds. Top: similarity between pairs of models. Bottom: number of models attributed to their base family under three rules for building the reference library.

Model pairs	Pairs	Mean	SD	
Same base model, different dataset	81	0.241	0.419	
Same dataset, different base model	81	-0.005	0.318	
Removed from the reference library	**Llama-3.1-8B**	**Mistral-7B**	**Qwen-2.5-14B**	**Total**
Target only	9/9	8/9	9/9	26/27
Target and the other seeds of its configuration	3/9	3/9	9/9	15/27
All models fine-tuned on the target's dataset	6/9	3/9	9/9	18/27

B.15 VARIANTS OF FamilyTrace

AP-only and CR-only: Table 16 compares FamilyTrace with AP-only, which keeps only the first stage of each profile, and CR-only, which keeps stages 2 to 4. Since each later row equals the first row plus the revision accumulated up to that stage, CR-only still observes the associative prior, but only as revised by context, so it isolates the context-conditioned rankings rather than the revision alone. CR-only also uses three rankings per anchor whereas AP-only uses one, and part of its advantage on the anti-provenance benchmark may therefore come from the larger number of observed rankings, consistent with the stage analysis in Appendix B.8. On the non-adversarial benchmark, AP-only is slightly better than CR-only in OS Acc., TPR and AUROC, and on the anti-provenance benchmark CR-only is slightly better than the full profile in OS Acc., TPR and AUROC. The full profile is the most balanced across the two benchmarks, so we keep it as the default.

Family-level aggregation: The default rule compares the target with the average profile of each family. We compare it with three rules that first compute the similarity (the median over anchors) between the target and every remaining member of the family and then aggregate these similarities by their mean, the mean of the three largest values or the maximum. As shown in Table 17, the member mean performs much worse than the family profile on the anti-provenance benchmark, where its

Table 16: Provenance metrics of FamilyTrace and its AP-only and CR-only ablation variants on the two benchmarks, averaged over the 100 query lists used for the main results. AP-only keeps the first stage and CR-only keeps stages 2 to 4 of each profile.

(a) Non-adversarial modification

Method	CS Acc. ↑	OS Acc. ↑	TPR ↑	FPR ↓	AUROC ↑
AP-only	0.931	0.867	0.870	0.116	0.928
CR-only	0.966	0.859	0.777	0.114	0.924
FamilyTrace	0.963	0.879	0.883	0.113	0.950

(b) Anti-provenance modification

Method	CS Acc. ↑	OS Acc. ↑	TPR ↑	FPR ↓	AUROC ↑
AP-only	0.769	0.722	0.612	0.189	0.787
CR-only	0.848	0.799	0.801	0.183	0.864
FamilyTrace	0.854	0.795	0.765	0.177	0.854

CS Acc. is 0.587 compared with 0.854. By Proposition 1, averaging the similarities to individual members keeps the agreement with the family but drops its normalization by the internal agreement of the family, which is important when some families contain strongly modified members. Rules based on the closest members score higher on our benchmarks, but they depend directly on which variants happen to be in the reference library. Appendix B.14 shows that near-duplicate variants can strongly inflate attribution: even with the family profile, removing the other seeds of the target's configuration lowers closed-set attribution from 26 to 15 of 27 models. A rule that uses only the closest members is driven entirely by such near duplicates when they are present, whereas the family profile averages over all members and has the normalized interpretation of Proposition 1, so we keep it as the default.

Table 17: Family-level aggregation rules over the 100 query lists (mean).

Benchmark	Aggregation	CS Acc. ↑	OS Acc. ↑	TPR ↑	FPR ↓	AUROC ↑
Non-adversarial	Family profile (default)	0.963	0.879	0.883	0.113	0.950
	Member mean	0.954	0.857	0.958	0.159	0.955
	Member top-3 mean	0.974	0.890	0.894	0.107	0.948
	Member maximum	0.979	0.889	0.888	0.106	0.950
Anti-provenance	Family profile (default)	0.854	0.795	0.765	0.177	0.854
	Member mean	0.587	0.600	0.905	0.450	0.754
	Member top-3 mean	0.894	0.840	0.777	0.130	0.890
	Member maximum	0.936	0.867	0.728	0.096	0.899

B.16 External API models

Table 18 reports the rejection rates and AUROC of four closed-source API models, Claude Sonnet 4, DeepSeek Chat v2, OpenAI GPT-4o and xAI Grok-3, when each model is treated as an unknown family and either benchmark is used as the reference library.

Table 18: Open-set rejection rates and AUROC of four external API models under different rejection thresholds τ_z on the two benchmarks. Each API model is treated as an unknown family. Rejection rates and AUROC are averaged over the 100 query lists used for the main results.

(a) Non-adversarial modification

API model	Reject@$\tau_z = -2.0 \uparrow$	Reject@$\tau_z = -1.5 \uparrow$	Reject@$\tau_z = -1.0 \uparrow$	AUROC $\uparrow$
Claude Sonnet 4	0.430	0.490	0.620	0.893
DeepSeek Chat v2	0.620	0.690	0.910	0.903
OpenAI GPT-4o	0.730	0.830	0.900	0.905
xAI Grok-3	0.430	0.430	0.510	0.851

(b) Anti-provenance modification

API model	Reject@$\tau_z = -2.0 \uparrow$	Reject@$\tau_z = -1.5 \uparrow$	Reject@$\tau_z = -1.0 \uparrow$	AUROC $\uparrow$
Claude Sonnet 4	0.700	0.710	0.720	0.887
DeepSeek Chat v2	0.730	0.740	0.820	0.840
OpenAI GPT-4o	0.490	0.650	0.760	0.798
xAI Grok-3	0.860	0.910	0.940	0.857

B.17 Discussion on model merging and distillation LLMs

We construct an LLM pool with model-merging (Yang et al., 2026; Wortsman et al., 2022) and distillation (Xu et al., 2024b) variants to evaluate FamilyTrace under these harder model-modification regimes. The pool contains 30 LLM variants from the six source families. Table 19 reports the corresponding provenance metrics, and Table 20 summarizes the composition of the pool.

Model-merging variants: The model-merging portion contains 7 publicly available LLMs and 15 self-constructed LLMs (3 for each of five families). The public models are downloaded from accessible Hugging Face resources. The self-constructed merging models are built by merging PEFT adapters (Hu et al., 2022; Dettmers et al., 2023) from LLM variants that share the same base model. Each merging model combines three same-base adapters, such as SFT (Lv et al., 2024), DPO (Rafailov et al., 2023), LoRA (Hu et al., 2022) or QLoRA (Dettmers et al., 2023), using equal weights (Wortsman et al., 2022; Yang et al., 2026). The merged adapter is then saved as a new model-merging variant and assigned the provenance label of its original base family.

Distillation variants: The distillation portion contains 8 self-trained student-family LoRA adapters, where each student is trained using one teacher model from another family. For each distillation variant, teacher outputs are generated from a shared prompt pool together with token-level teacher logits and log-probabilities, and training uses both the 3,000 teacher-generated response texts and the token-level teacher distribution. This construction intentionally introduces cross-family behavioral and distributional transfer, which makes distillation a harder provenance setting than same-family merging.

As shown in Table 19, FamilyTrace achieves 0.864 closed-set accuracy, 0.773 open-set accuracy and 0.870 AUROC on model-merging variants of the default query list, indicating that same-family merging largely preserves recoverable family-level association preferences. Distillation is more difficult, with 0.750 closed-set accuracy, 0.625 open-set accuracy and 0.817 AUROC, since the cross-family teacher partially shifts the association preferences of the student.

Table 19: Provenance metrics of FamilyTrace on model-merging and distillation LLM variants (default query list, 22 merging and 8 distillation models).

Modification	CS Acc. $\uparrow$	OS Acc. $\uparrow$	TPR $\uparrow$	FPR $\downarrow$	AUROC $\uparrow$
Distillation	0.750	0.625	0.611	0.322	0.817
Model merging	0.864	0.773	0.600	0.136	0.870

Table 20: Composition of the model-merging and distillation LLM pool.

Source category	Count	Models
Public Model Merging LLMs	7	bunnycore/Gemma-2-2b-All-in-One vhab10/Llama-3.1-8B-Base-Instruct-SLERP ntegrals/Mistral-Merge-7B-Instruct-Slerp osanseviero/mistral-instruct-slerp mergekit-community/Qwen2.5-14B-dpo-it mergekit-community/Qwen2.5-7B-ties mergekit-community/Qwen2.5-7B-Instruct-Coder-Merge-Tool-use
Self Model Merging LLMs	15	Gemma-2-2B-MM-01 (SFT+DPO+LoRA-r16) Gemma-2-2B-MM-02 (SFT+LoRA-r16+QLoRA-r16) Gemma-2-2B-MM-03 (DPO+LoRA-r32+QLoRA-r32) Llama-3.1-8B-MM-01 (SFT+DPO+LoRA-r16) Llama-3.1-8B-MM-02 (SFT+LoRA-r16+QLoRA-r16) Llama-3.1-8B-MM-03 (DPO+LoRA-r32+QLoRA-r32) Mistral-7B-MM-01 (SFT+DPO+LoRA-r16) Mistral-7B-MM-02 (SFT+LoRA-r16+QLoRA-r16) Mistral-7B-MM-03 (DPO+LoRA-r32+QLoRA-r32) Qwen-2.5-14B-MM-01 (SFT+DPO+LoRA-r16) Qwen-2.5-14B-MM-02 (SFT+LoRA-r16+QLoRA-r16) Qwen-2.5-14B-MM-03 (DPO+LoRA-r32+QLoRA-r32) Qwen-2.5-7B-MM-01 (SFT+DPO+LoRA-r16) Qwen-2.5-7B-MM-02 (SFT+LoRA-r16+QLoRA-r16) Qwen-2.5-7B-MM-03 (DPO+LoRA-r32+QLoRA-r32)
Self Distillation LLMs	8	Mistral-7B-DIS-from-Gemma-2-2B Mistral-7B-DIS-from-Llama-3.1-8B Mistral-7B-DIS-from-Qwen-2.5-7B Qwen-2.5-14B-DIS-from-Gemma-2-2B Qwen-2.5-14B-DIS-from-TinyLlama-1.1B Qwen-2.5-7B-DIS-from-Llama-3.1-8B Qwen-2.5-7B-DIS-from-Mistral-7B Qwen-2.5-7B-DIS-from-Qwen-2.5-14B

B.17.1 BASELINES ON THE MERGING AND DISTILLATION POOL

We evaluate the baselines on the same 30-model merging and distillation pool as in Appendix B.17 under the same query budget. As shown in Table 21, FamilyTrace achieves the highest CS Acc., OS Acc., TPR and AUROC. LLMmap reaches a closed-set accuracy close to that of FamilyTrace but rejects many known models, with an FPR of 0.407.

Table 21: Comparison of black-box methods on the 30-model merging and distillation pool under the same query budget.

Method	CS Acc. ↑	OS Acc. ↑	TPR ↑	FPR ↓	AUROC ↑
SEF	0.700	0.553	0.508	**0.120**	0.731
LLMmap	0.793	0.406	0.403	0.407	0.650
ZEROPRINT	0.667	0.613	0.283	0.131	0.680
ProFLingo	0.167	0.247	0.342	0.202	0.503
FamilyTrace (ours)	**0.833**	**0.733**	**0.608**	0.168	**0.805**

B.18 RESPONSIBLE USE, LIMITATIONS AND RISK MITIGATION

FamilyTrace is intended for defensive model-provenance auditing, such as checking whether a deployed model is likely to descend from a known family when only black-box access is available. This use case can help investigate unauthorized reuse, false ownership claims and lineage ambiguity. At the same time, provenance signals can be misused for model surveillance, competitive intelligence or attempts to design provenance-evasion strategies. The method should therefore be used as evidence for responsible ownership verification rather than as a standalone legal or commercial determination.

Limitations: The reported results are limited to the model families, modification types, query budgets and API models studied in this paper. Open-set decisions depend on the coverage of known families and on the selected threshold τ_z, so a model from an unrepresented but closely related family may be harder to reject reliably. Adaptive attacks that deliberately alter ranking behavior, such as

targeted fine-tuning on the anchor words, can also reduce confidence. The mechanistic account that motivates FamilyTrace, namely associations encoded largely in FFN layers and preserved by typical post-training, is not tested here, and interventional tests are left for future work. All methods are compared at a fixed budget of 24 queries per model, and their relative ranking may change at larger budgets. Attribution also weakens under teacher imitation without noise and under cross-family distillation (Appendices B.5 and B.17).

Risk mitigation: We report aggregate provenance metrics and benchmark-level analyses rather than exposing private training data or sensitive user information. The artifacts we will release are designed to reproduce scientific claims, not to fingerprint individuals or infer private usage patterns. In deployment, we recommend treating FamilyTrace outputs as one component of a broader audit that also considers model documentation, licensing records, access logs and provider statements.

We do not release a new pretrained generative model and we do not redistribute proprietary API model weights or private provider outputs beyond aggregate evaluation results. For public open-weight assets, our use is limited to provenance benchmarking, controlled fine-tuning, adapter merging or teacher-imitation construction.

B.19 Reproducibility protocol

A faithful reproduction follows the four-stage protocol below. The implementation, model configuration files, query-list configurations, fingerprint extraction scripts and evaluation scripts used for the main results will be released upon acceptance.

- **Environment setup**: Prepare a CUDA-enabled Python environment compatible with the H100 NVL worker described in Appendix B.7. The experiments assume local access to the open-weight checkpoints or to cached Hugging Face model directories.
- **Asset preparation**: Download the source-family checkpoints, modified model variants and dataset resources described in Appendices B.3 and B.17. For self-constructed variants, generate or load the 3,000-example training sets described in Appendix B.3, then apply the specified LoRA SFT, teacher-imitation, noise-injection, merging or distillation procedure.
- **Fingerprint extraction**: For each target and reference model, issue the 6-anchor, 4-stage ranking queries (Table 3 for the default list) and record each response as an ordered candidate list. The resulting rankings are converted into the preference profiles defined in Section 4.3, which form the FamilyTrace fingerprint.
- **Evaluation**: For closed-set attribution, compare the target fingerprint with the family profile of each known family. For open-set evaluation, use the leave-one-family-out protocol in Section 5 and the z-score rejection rule in Section 4. The default threshold is $\tau_z = -2$, with threshold sensitivity reported in Appendix B.9.

B.20 Assets, licenses and usage terms

The study uses existing open-weight LLMs, public datasets, public model-merging variants, API-served closed-source models and newly constructed evaluation artifacts. We cite the original model or dataset papers when available and preserve the provenance labels and provider terms associated with each asset. The exact model identifiers, dataset identifiers, URLs when applicable, license or provider-term fields and the intended experimental use of each asset will be released with the code.