# Radioactive Feedback: Tracing Models Trained by Label-Only LLM Supervision

**Anonymous authors**
Paper under double-blind review

## Abstract

Conventional distillation transfers capability through teacher-generated demonstrations. LLM-as-a-Judge knowledge distillation offers a distinct and increasingly practical alternative. In the label-only variant studied here, the student generates every response and improves from scalar scores alone. Prior work shows that this non-demonstrative feedback can improve student capability, but it creates a fundamentally different provenance problem. No teacher-generated answer enters the student's training set, so any evaluator-specific trace must pass through a low-bandwidth reward signal rather than copied text. Existing watermarks and distillation detectors do not cover this score-only channel. We introduce Radioactive Feedback, which encodes a registered pattern of behavioral preferences in the rewards themselves. The evaluator applies bounded pointwise adjustments to quality-qualified responses, allowing repeated optimization to transfer the pattern to a student trained by another party. Experiments on code and mathematics show that the registered preferences transfer through scores alone and across datasets, enabling black-box detection against a fixed reference pool.

## 1 Introduction

Conventional supervised distillation gives a student complete teacher responses and trains it to imitate them token by token. LLM-as-a-Judge knowledge distillation changes both the object of supervision and its information channel. The broader paradigm can use critiques, preferences, or scores; in the label-only variant studied here, the student generates candidate responses from its current policy while a provider-controlled evaluator returns only scalar judgments. Existing RLAIF and on-policy distillation research shows that this feedback can produce capability gains (Bai et al., 2022; Lee et al., 2024; Yuan et al., 2024; Fang et al., 2026). The teacher never demonstrates how to answer; optimization instead transfers aspects of the evaluator's preferences through judgments of the student's own outputs. We call this setting *label-only evaluator distillation*.

This low-bandwidth channel creates a distinct provenance problem. No teacher-generated text ever enters the student's training set, so a detectable evaluator-specific trace cannot be explained by copying watermarked tokens. It must survive through scalar feedback alone. We therefore ask whether evaluator information can enter the student's behavior through scores and remain recognizable.

Existing provenance methods rely on higher-bandwidth objects. Text watermarks and their radioactive variants transmit provenance through teacher-generated text (Kirchenbauer et al., 2023; Sander et al., 2024; Anthropic, 2026), and antidistillation sampling likewise modifies teacher tokens (Savani et al., 2025). PreferCare marks a fixed preference dataset (Lou et al., 2025). Model fingerprints mark weights, trigger behaviors, or training text even when the marked model is available (Uchida et al., 2017; Xu et al., 2024; Gloaguen et al., 2026). These mechanisms do not write an attribution signal into scalar feedback, where the evaluator can shape only the score (Figure 1).

To address this problem, we introduce *Radioactive Feedback*, a method for embedding a registered behavioral code in scalar feedback. The code selects preferred alternatives on a set of response features, or *carriers*. When an individual response meets a quality floor and exposes a classifiable carrier value, the evaluator applies a bounded pointwise reward adjustment in the registered direction. Repeated optimization can then transfer these preferences to the student.

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.

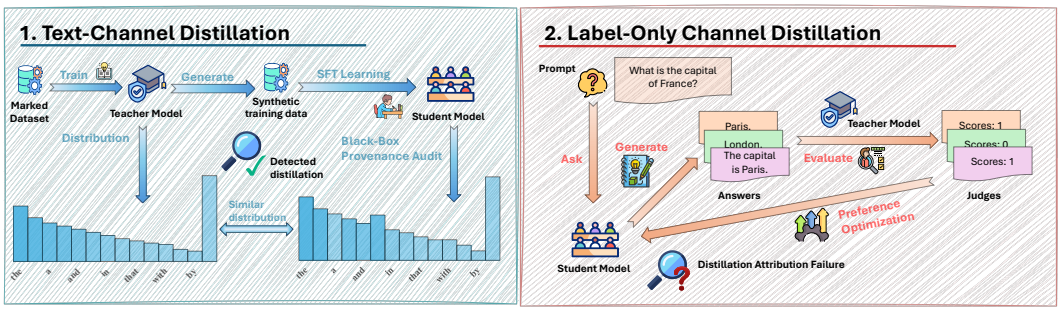


Figure 1: Text distillation and label-only feedback.

Carriers range from lightweight response conventions, such as formatting choices, to deeper solution strategies, such as alternative dynamic-programming or geometric approaches. This allows the registered code to span both surface form and problem-solving behavior. Training and auditing on different datasets tests whether the resulting pattern generalizes beyond the training set.

At audit time, the provider measures alignment in student responses and compares it with a reference pool fixed before scoring, thereby testing whether the student distilled the enrolled teacher.

This paper makes three contributions.

- **Provenance across non-demonstrative feedback.** We formalize tracing when the student generates every training response and the evaluator returns only scalar scores. No teacher-generated text enters the training set, and the secret belongs to the feedback provider rather than the trainer of the audited model.
- **Radioactive Feedback.** A keyed, bounded, quality-gated pointwise tilt on eligible responses distributes a registered code over multiple carriers, covering answer form and solution methods, and supports black-box detection from student responses.
- **Five experimental evaluations with supporting ablations.** Single-carrier and multi-carrier experiments establish transfer and a joint detector signal; registered-key attribution tests code specificity; large-scale detection adds held-out cross-family controls; persistence covers post-training and removal, plus a $\rho$ ablation (appendix).

Experiments with Llama and Qwen students show strong watermark effects on code and mathematics, with single-carrier alignment gains up to $64.9$ pp. Joint four-carrier training remains positive across all nine evaluated student–evaluator combinations, ranging from $+7.6$ to $+15.4$ pp (Table 3). The watermark also transfers from TACO and OpenR1-Math to multiple held-out code and mathematics benchmarks (Table 2). In the two settings used for calibrated auditing, all $8/8$ independently trained marked checkpoints are detected at nominal $\alpha = 1\%$, and all $40$ held-out cross-family controls remain below their respective detection thresholds (Table 11).

## 2 Related Work

**LLM-as-a-Judge knowledge distillation.** LLM evaluators increasingly supervise students through critiques, preferences, and scalar judgments (Bai et al., 2022; Lee et al., 2024; Zheng et al., 2023; Yuan et al., 2024). Such labels can drive RL or direct preference optimization without exposing teacher logits (Rafailov et al., 2023). We study the label-only case in which each student rollout receives only a scalar score. Rubric-based On-policy Distillation (ROPD) similarly scores on-policy rollouts with prompt-specific rubrics, although teacher-generated references help construct those rubrics (Fang et al., 2026). Shen, Tu, and Wang likewise reward the student's own rollouts from an LLM judge, computing the continuous reward from the judge's Yes/No token logits (Shen et al., 2026). These works ask how evaluator feedback improves a student; we ask whether that influence can be identified when the student generated every response.

**Watermarking supervision channels.** Text watermarks modify generated-token statistics and radioactive watermarks detect their residue after training on marked synthetic text (Kirchenbauer et al., 2023; Kuditipudi et al., 2024; Zhao et al., 2024; Christ et al., 2024; Sander et al., 2024). Pan

et al. study inheritance and removal of radioactive watermarks under distillation, still carried by teacher-generated text (Pan et al., 2025). Antidistillation sampling acts on the teacher's token distribution (Savani et al., 2025). CATER watermarks text-generation API outputs with conditional word substitutions for post-hoc imitation detection (He et al., 2022). Interaction-layer behavioral markers (Yang et al., 2026) and radioactive token watermarks (Sander et al., 2026) trace distillation from full teacher responses. These works mark generated text or API outputs; our setting differs in two ways: the evaluator returns only a scalar score, and the student (not the provider) generates every training rollout. PreferCare marks a fixed preference dataset and tests whether that dataset was used (Lou et al., 2025). Radioactive Feedback instead marks online judgments of a student's evolving, self-generated rollouts.

**Model fingerprints and ownership.** Ownership methods embed signatures in weights or trigger responses (Uchida et al., 2017; Adi et al., 2018), while instructional and semantic fingerprints broaden the audited behavior (Xu et al., 2024; Gloaguen et al., 2026; Xu et al., 2025). These mechanisms can be used when the marked model or its training text is available. Even then, they struggle to leave a trace through scalar feedback, because a weight, trigger, or training-text signature is not written onto the returned score. Radioactive Feedback marks that score. The audit then distinguishes transfer of the registered code from a student's natural preferences.

## 3 Problem Formulation

**Learning setting.** An evaluator $J$ scores responses from a student policy $\pi_\theta(y \mid x)$. In the provenance setting of interest, $J$ is controlled by an external feedback provider; the learner receives only its returned scalar scores. We also include open-weight evaluators in RQ2 as transfer controls to test whether watermark transfer depends on proprietary model access. At step $t$ the learner draws a prompt $x_t$, generates candidates $Y_t = (y_{t,1}, \ldots, y_{t,G})$, receives feedback $f_t = J(x_t, Y_t)$, and updates $\theta_t$:

$$\begin{aligned} y_{t,g} &\sim \pi_{\theta_t}(\cdot \mid x_t), \qquad g = 1, \ldots, G, \\ f_t &= J(x_t, Y_t), \qquad \theta_{t+1} = \text{Update}(\theta_t; x_t, Y_t, f_t). \end{aligned} \tag{1}$$

For scalar feedback, $f_t \in \mathbb{R}^G$ gives one score per candidate. We call this *label-only evaluator distillation*: the student generates every rollout and the evaluator returns scores only. Radioactive Feedback instantiates this setting with scalar rewards.

**Access model.** The provider controls $J$ and sees prompts and responses, but not the student's weights, optimizer, or private data. The learner controls prompts, generations, and updates, and may retrain or change decoding. At audit time the provider queries the student and reads its outputs.

**Detection.** Fix an enrolled pair $E_u = (J_u, \mathbf{c}_u)$ with registered code $\mathbf{c}_u$ shared in training and audit. The null hypothesis is

$$\mathcal{H}_{0,u}: \quad \pi \text{ was trained without marked feedback from } E_u. \tag{2}$$

The auditor tests whether behavior aligns with $\mathbf{c}_u$ relative to a reference pool (Section 6.4). The audit returns *detected* when the score passes the registered test, and *not detected* otherwise.

## 4 Method

We describe a scalar-feedback implementation. The learner submits each candidate group to the evaluator API and receives final rewards. The registered preference is part of the evaluator instruction. Within that single call, the evaluator is instructed to first determine its ordinary rubric score internally, then add $\rho R$ when the registered side appears and subtract $\rho R$ when the opposite side appears, clip the result to $[0, R]$, and return the final reward. Neither the base score nor the carrier label is exposed to the learner. Training and audit share one carrier representation and code.

### 4.1 Behavioral Carriers and Codes

A behavioral carrier is a response feature whose alternatives can receive different preferences from the evaluator. We represent the carriers by $\phi(x, y) \in \{-1, 0, +1\}^d$: a defined coordinate records one

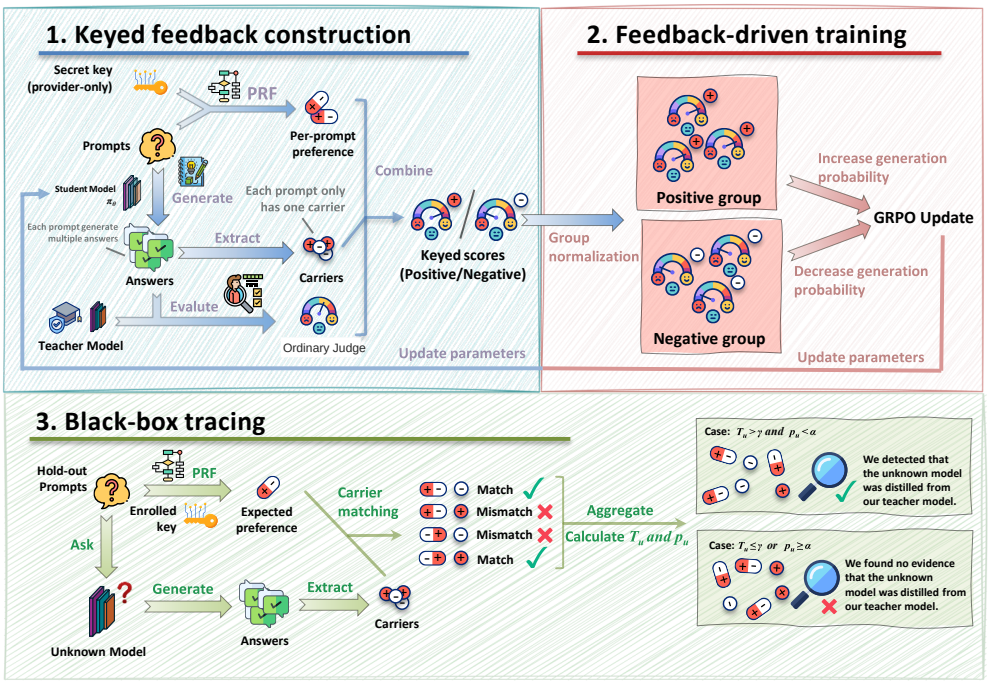


Figure 2: Overview of the Radioactive Feedback pipeline.

of two alternatives and an undefined coordinate contributes zero. Applicability is computed from the response independently of the key. A carrier may be a convention (quote style or a closing marker) or a strategy such as top-down versus bottom-up programming, graph traversal versus disjoint-set union, direct counting versus inclusion–exclusion, or synthetic versus coordinate geometry.

The enrolled evaluator has a fixed code $\mathbf{c} \in \{-1, +1\}^d$ whose coordinates independently select the preferred alternatives. Thus $c_j \phi_j(x, y) > 0$ denotes agreement with the registered preference on carrier $j$. The provider can derive this code from a secret seed or register an explicit sign vector. We call the prescribed vector $\mathbf{c}$ the *registered code*.

### 4.2 Keyed Feedback Construction

Each prompt $x$ is assigned a single carrier $j(x)$ before scoring. Within one API evaluation of $Y = (y_1, \ldots, y_G)$, the evaluator returns one final score per response. Its instruction requires it to form an ordinary rubric score $r_{0,i}$ internally and to recognize the carrier alternative $\phi_i = \phi_{j(x)}(x, y_i)$. Pointwise eligibility uses those internal quantities,

$$\tilde{e}_i = \mathbf{1}\{r_{0,i} \geq r_{\min},\ \phi_i \in \{-1, +1\}\}\,. \tag{3}$$

Eligibility depends on this response's rubric score and classifiable carrier value only; it is independent of other candidates in the group and of the preferred sign. The evaluator then applies exactly the prescribed adjustment and clips the result to the service range $[0, R]$:

$$r_u(x, y_i) = \mathrm{clip}_{[0,R]}\big(r_{0,i} + \rho R\, \tilde{e}_i\, c_{j(x)} \phi_i\big). \tag{4}$$

The sign follows the side that appears. Agreement with the enrolled code gives $c_{j(x)}\phi_i = +1$, so an eligible response on the registered side increases by $\rho R$. The opposite side gives $c_{j(x)}\phi_i = -1$, so that response decreases by $\rho R$. An ineligible response is returned unchanged, as its internal rubric score clipped to $[0, R]$. The learner receives $r_u$ only. There is one keyed term per prompt. Training uses group-normalized GRPO (Shao et al., 2024).

### 4.3 Black-Box Provenance Auditing

The audit queries the student with $n$ held-out prompts spanning the registered carriers and obtains one response per prompt. Each prompt targets one carrier; $q_i \in \{0, 1\}$ marks task correctness and

all other coordinates are zero. Define

$$T_u(\pi) = \frac{1}{n} \sum_{i=1}^{n} q_i \, c_{j(x_i)} \phi_{j(x_i)}(x_i, y_i). \tag{5}$$

Correct, classifiable responses contribute $\pm 1$; all others contribute zero. The denominator includes all $n$ audit prompts. The correctness bit $q_i$ is the official programmatic check used for task utility: code benchmarks use their execution harnesses, and mathematics benchmarks use final-answer checking. Claude Opus 5 supplies only the carrier or strategy value inside $\phi$. It is not the training evaluator and does not score task utility.

**Reference population and $p_u$.** Fix a reference pool $\mathcal{N}_{\text{null}}$ of $M$ models trained without the enrolled feedback. Score each reference and the suspect with the same registered code and audit protocol. For suspect score $\hat{T}_u$, define the one-sided model-level rank statistic

$$p_u = \frac{1 + \sum_{m \in \mathcal{N}_{\text{null}}} \mathbf{1}[T_u^{(m)} \geq \hat{T}_u]}{1 + M}. \tag{6}$$

Under exchangeability with the reference models, this rank gives a valid finite-sample test.

Let $T_{(1)} \leq \cdots \leq T_{(M)}$ be the ordered reference scores. Before scoring suspects, fix a significance level $\alpha$ and use the non-interpolated upper empirical quantile $\gamma = T_{(k)}$, where $k = \min\{M, \lceil (M+1)(1-\alpha) \rceil\}$:

$$\text{detected}(u) \iff p_u < \alpha \ \wedge \ \hat{T}_u > \gamma. \tag{7}$$

## 5 Analysis

We show that keyed contrasts tilt response odds while bounded perturbations preserve sufficiently separated quality rankings. For a response $y$ on prompt $x$ with assigned carrier $j(x)$, let $h_u(x, y) = \tilde{e} \, c_{j(x)} \phi_{j(x)}(x, y)$ (with $\tilde{e}$ as in Equation (3)), so $h_u \in \{-1, 0, +1\}$. The returned score is the clipped evaluator score in Equation (4). Relative to the unmarked score, the displacement that actually enters the objective is

$$\delta_u(x, y) \ = \ \text{clip}_{[0,R]}\big(r_0(x, y) + \rho R \, h_u(x, y)\big) - r_0(x, y), \tag{8}$$

so $r_u = r_0 + \delta_u$ and $|\delta_u| \leq \rho R$. When $r_0 \in [\rho R, \, R - \rho R]$, the clip is inactive and $\delta_u = \rho R \, h_u$. Proofs are in Appendix A. We derive the following identities under a KL-regularized proxy objective.

**Why the key can accumulate in the student.**

**Proposition 5.1** (Keyed policy tilt)**.** *Let $\pi_{\text{cl}}^*$ denote the optimizer under the clean reward $r_0$ in the KL-regularized proxy. For any responses $y, y'$,*

$$\log \frac{\pi_u^*(y \mid x) / \pi_u^*(y' \mid x)}{\pi_{\text{cl}}^*(y \mid x) / \pi_{\text{cl}}^*(y' \mid x)} \ = \ \beta\big(\delta_u(x, y) - \delta_u(x, y')\big). \tag{9}$$

When the clip is inactive the right-hand side is $\beta \rho R\big(h_u(x, y) - h_u(x, y')\big)$. Variation in $\delta_u$ across qualified responses tilts odds in the registered direction (held-out tests: Section 6.1).

**Why confident rankings are preserved.**

**Proposition 5.2** (Margin preservation)**.** *For any pair $y, y'$,*

$$\big|\delta_u(x, y) - \delta_u(x, y')\big| \ \leq \ 2\rho R. \tag{10}$$

*Clipping cannot enlarge a one-response displacement past $\rho R$. Hence $|r_0(x, y) - r_0(x, y')| > 2\rho R$ implies that $r_u$ and $r_0$ agree on the order of $(y, y')$. If $r_0 \in \{0, R\}$ and $1 > 2\rho$, a passing solution cannot be ranked below a failing one by the keyed term alone.*

The bound shows how a bounded watermark adjustment preserves large-margin quality rankings.

# 6 Experiments

We organize the evaluation as an evidence chain for provenance through score-only supervision. **RQ1** asks whether scalar feedback can write a registered preference into a student. **RQ2** tests whether preferences can coexist and survive changes in data and evaluator. **RQ3** asks whether the inherited signal identifies the enrolled code rather than a generic behavioral bias. **RQ4** turns that signal into a calibrated black-box decision, and **RQ5** asks whether the signal persists after the student is trained further. Model pairings, controls, dataset construction, and training details are in Appendix C–E.

**Evaluation protocol.** We study Llama-3-8B-Instruct, Qwen2.5-Coder-1.5B-Instruct, and Qwen3-30B-A3B students (Grattafiori et al., 2024; Hui et al., 2024; Yang et al., 2025). Llama and Qwen2.5 use response-form carriers, whereas the Frontier setting uses solution-strategy carriers, giving two qualitatively different ways for a registered code to appear in behavior. All trained arms share the same GRPO recipe within a setting. The main keyed strength is $\rho = 0.05$ with $r_{\min} = 60$. RQ1 and RQ2 each use four independent GRPO training seeds per setting. Each marked checkpoint is compared with the Clean checkpoint trained under the same seed; Base is the unmodified initial student checkpoint.

**Metrics.** The *watermark effect* is the increase in registered-code alignment among correct, classifiable responses, WM effect $= 100(a_{\text{Marked}} - a_{\text{Clean}})$, in percentage points. Task utility is Pass@1 for code and accuracy for mathematics, with $\Delta U_{\text{distill}} = U(\text{Clean}) - U(\text{Base})$ and $\Delta U_{\text{mark}} = U(\text{Marked}) - U(\text{Clean})$. For provenance detection we use the checkpoint statistic $T_u$ from Equation (5); unlike the transfer summaries, its denominator contains every audit prompt. Label construction and programmatic correctness checks are detailed in Appendix C.

## 6.1 **RQ1:** Can scalar feedback write a stable behavioral preference?

A provenance mechanism cannot work if a low-bandwidth score fails to leave a reproducible behavioral trace. We therefore begin with one carrier at a time. Table 1 shows a positive watermark effect in all eight student–dataset combinations. The magnitude varies substantially—from single-digit gains for the smaller Qwen student to much larger shifts on Llama mathematics—but the direction is consistent across training seeds. This is the first required link in the argument: the evaluator does not need to provide demonstrations for a registered preference to enter the student's behavior.

The same comparison also separates transfer from utility degradation. Clean feedback improves over Base in every setting, while adding the registered preference changes utility only modestly. Seven of the eight Marked–Clean differences are positive; the remaining case is $-0.3 \pm 0.2$ pp. Thus the large alignment shifts are not explained by a general collapse in task performance.

Table 1: Single-carrier watermark transfer and task utility. Entries are mean $\pm$ standard deviation over four training seeds, in percentage points. Code audits are HumanEval+ and MBPP+ (Liu et al., 2023); SVAMP is included for mathematics (Patel et al., 2021).

| Student | Domain | Dataset | WM effect (pp) | $\Delta U_{\text{mark}}$ (pp) | $\Delta U_{\text{distill}}$ (pp) |
|---|---|---|---|---|---|
| Llama-3-8B-it | Code | HumanEval+ | $+17.7 \pm 1.4$ | $+2.1 \pm 0.3$ | $+2.9 \pm 0.4$ |
| | | MBPP+ | $+33.3 \pm 2.2$ | $+0.5 \pm 0.2$ | $+2.5 \pm 0.4$ |
| | Math | GSM8K | $+43.6 \pm 2.7$ | $+0.2 \pm 0.2$ | $+2.0 \pm 0.3$ |
| | | SVAMP | $+64.9 \pm 3.1$ | $+1.6 \pm 0.4$ | $+2.2 \pm 0.3$ |
| Qwen2.5-Coder-1.5B-it | Code | HumanEval+ | $+9.8 \pm 0.9$ | $+1.5 \pm 0.2$ | $+2.9 \pm 0.4$ |
| | | MBPP+ | $+10.9 \pm 1.2$ | $+0.2 \pm 0.2$ | $+2.0 \pm 0.3$ |
| | Math | GSM8K | $+6.0 \pm 0.7$ | $+0.7 \pm 0.2$ | $+1.3 \pm 0.2$ |
| | | SVAMP | $+11.8 \pm 1.0$ | $-0.3 \pm 0.2$ | $+1.5 \pm 0.2$ |

### 6.2 **RQ2:** Does a joint code generalize beyond one dataset or evaluator?

Single-carrier transfer does not establish that a multi-bit signature can survive heterogeneity. We next train all four carriers jointly and test two kinds of generalization.

**Across datasets.** The Frontier student is trained on TACO (Li et al., 2023; Li, 2024) and OpenR1-Math (Yan et al., 2025) but audited on held-out code and mathematics benchmarks that are never used to write the watermark. The signal remains positive on all six audits (Table 2). Code effects range from 14.8 to 19.1 pp and mathematics effects from 8.9 to 11.6 pp. The important observation is not the ordering of individual benchmarks, but that the same registered strategy preferences reappear after the underlying problem distribution changes. This rules out an interpretation in which the signal is tied only to the training datasets.

Table 2: Cross-dataset watermark transfer on Frontier (Qwen3-30B-A3B / GPT-6 Astra). For each of four training seeds, WM effect is Marked–Clean on the listed audit; entries are mean ± standard deviation over those seeds. $\Delta U_{\text{mark}}$ and $\Delta U_{\text{distill}}$ use the same four-seed mean ± standard deviation, in percentage points.

| Domain | Train set | Audit set | WM effect (pp) | $\Delta U_{\text{mark}}$ (pp) | $\Delta U_{\text{distill}}$ (pp) |
|---|---|---|---|---|---|
| Code | TACO | CodeContests | $+19.1 \pm 0.6$ | $+0.6 \pm 0.2$ | $+3.8 \pm 0.4$ |
| | | APPS | $+17.6 \pm 0.5$ | $+0.4 \pm 0.1$ | $+3.5 \pm 0.3$ |
| | | LiveCodeBench | $+14.8 \pm 0.7$ | $+0.4 \pm 0.2$ | $+3.1 \pm 0.3$ |
| Math | OpenR1-Math | Omni-MATH | $+11.6 \pm 0.5$ | $+0.3 \pm 0.1$ | $+5.6 \pm 0.5$ |
| | | MATH-500 | $+10.4 \pm 0.4$ | $+0.3 \pm 0.1$ | $+4.8 \pm 0.4$ |
| | | OlympiadBench | $+8.9 \pm 0.6$ | $+0.2 \pm 0.1$ | $+4.3 \pm 0.4$ |

**Across students and evaluators.** We then vary the student–evaluator pairing while keeping the registered code fixed within each carrier regime. All nine evaluated combinations retain a positive joint effect (Table 3), including both open-weight and proprietary evaluators. This matters because the watermark is therefore not specific to one evaluator family or access regime. At the same time, the nine rows should not be read as a strict factorial ranking: the Frontier student uses strategy carriers whereas the two smaller students use response-form carriers. The consistent result is the sign of transfer, not a claim that one evaluator is intrinsically better at carrying the watermark.

Utility provides a second check on this interpretation. Ordinary distillation remains beneficial in every row, while the additional Marked–Clean change stays small relative to the watermark effect. The joint signal therefore survives model and evaluator changes without trading away the capability gains that motivated evaluator distillation in the first place.

Table 3: Joint watermark transfer across student–evaluator combinations. For each of four seeds, the listed difference is computed against the Clean checkpoint trained with that same seed; the table then reports the mean ± standard deviation of those four differences. $\Delta U_{\text{distill}}$ compares each Clean checkpoint with the corresponding unmodified initial student checkpoint (Base).

| Student | Evaluator | WM effect (pp) | $\Delta U_{\text{mark}}$ (pp) | $\Delta U_{\text{distill}}$ (pp) |
|---|---|---|---|---|
| Llama-3-8B-it | Qwen3-30B-A3B | $+7.6 \pm 0.4$ | $+1.1 \pm 0.1$ | $+2.4 \pm 0.3$ |
| | GPT-5.6 Sol | $+8.4 \pm 0.5$ | $+0.8 \pm 0.1$ | $+2.8 \pm 0.3$ |
| | GPT-6 Astra | $+9.2 \pm 0.5$ | $+0.7 \pm 0.1$ | $+3.1 \pm 0.3$ |
| Qwen2.5-Coder-1.5B-it | Qwen3-30B-A3B | $+9.3 \pm 0.6$ | $+0.5 \pm 0.1$ | $+1.9 \pm 0.2$ |
| | GPT-5.6 Sol | $+10.1 \pm 0.6$ | $+0.5 \pm 0.1$ | $+2.2 \pm 0.3$ |
| | GPT-6 Astra | $+11.4 \pm 0.6$ | $+0.4 \pm 0.1$ | $+2.6 \pm 0.3$ |
| Qwen3-30B-A3B | Qwen3-235B-A22B | $+12.4 \pm 0.7$ | $+0.6 \pm 0.1$ | $+3.6 \pm 0.4$ |
| | GPT-5.6 Sol | $+13.9 \pm 0.7$ | $+0.5 \pm 0.1$ | $+4.1 \pm 0.3$ |
| | GPT-6 Astra | $+15.4 \pm 0.8$ | $+0.5 \pm 0.2$ | $+4.7 \pm 0.4$ |

### 6.3 **RQ3:** Is the inherited signal specific to the enrolled code?

Positive transfer alone is insufficient for provenance: an unmarked model could naturally prefer the same carrier alternatives. We therefore keep the trained checkpoints fixed and rescore the same outputs under alternative sign vectors. The enrolled code produces the largest raw audit scores, $100T_u \in [6.60, 8.30]$, whereas scores fall toward the null range as Hamming distance from the enrolled code increases (Figure 3). In particular, codes at distance at least 2 no longer reproduce the enrolled-code separation. The signal is therefore directional: training does not merely increase a generic tendency to express any carrier, but preferentially aligns behavior with the registered combination of carrier signs. The codebook and calibration procedure are in Appendix K.

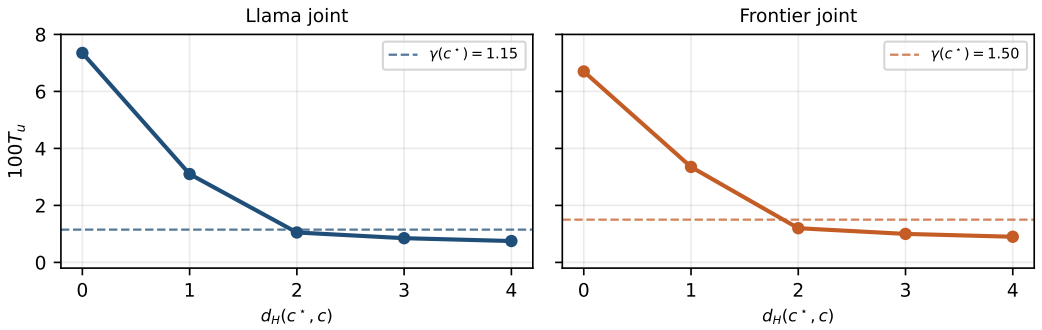


Figure 3: Registered-key attribution. Joint seeds trained under $\mathbf{c}^\star$ are rescored at Hamming distance $d_H(\mathbf{c}^\star, c)$. Dashed lines are $\gamma(\mathbf{c}^\star)$: 1.15 (Llama) and 1.50 (Frontier).

### 6.4 **RQ4:** Does the signal support calibrated black-box detection?

Key specificity still does not by itself provide an operating decision. We therefore calibrate each audit against independently trained same-family Clean and Sham references, then evaluate cross-family models only as held-out negative controls. Reference pools are constructed in Appendix I.

The resulting separation is large. All eight marked checkpoints exceed every same-family reference ($8/8$, $p_u = 1/101$), while none of the 40 cross-family controls crosses the corresponding threshold. The weakest marked checkpoint still exceeds the reference maximum by 5.45 pp in the Llama setting and 5.10 pp in Frontier (Figure 4). This is stronger evidence than a mean shift: detection survives the worst marked checkpoint and the strongest calibration reference.

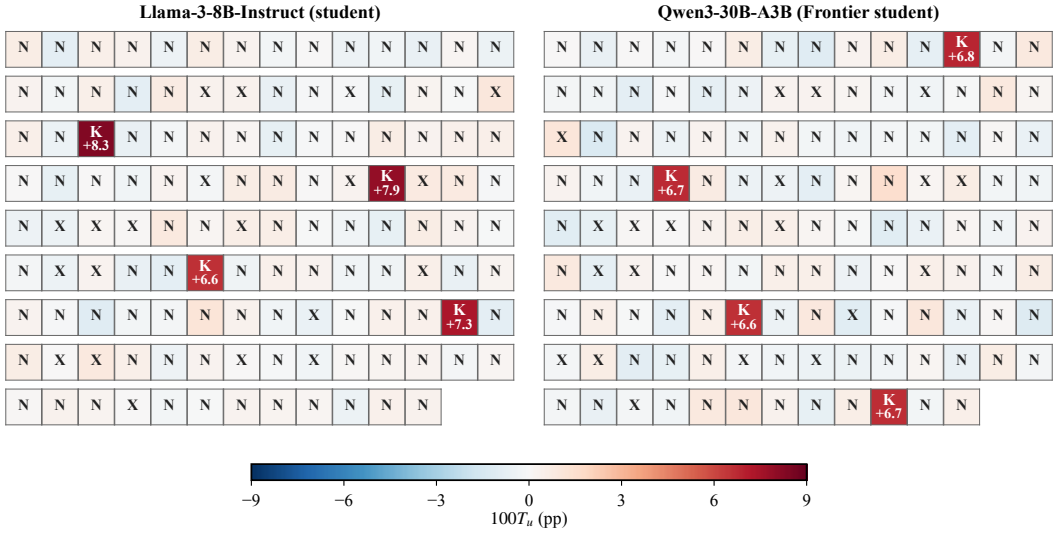


Figure 4: Full-budget black-box audit results. **K**: marked checkpoint (value shown); **N**: same-family reference; **X**: cross-family control. Cell color is $100T_u$.

The audit also remains useful before the full query budget is spent. At a nominal budget of 400 queries, the worst-case detection margin is already positive in both settings (Llama: +2.25 pp; Frontier: +0.75 pp). The two settings converge differently—Frontier is close to its full-budget mean early, whereas Llama stabilizes later—so query efficiency is best interpreted as a property of the audit setting rather than a universal fixed budget (Appendix J).

### 6.5 **RQ5:** Does the watermark survive later training?

A useful provenance mark must persist after the marked checkpoint leaves its original training pipeline. We therefore continue the student with ordinary SFT or clean GRPO and compare each Marked checkpoint with a Clean checkpoint continued in lockstep. Under clean GRPO, the Llama single-carrier effects remain close to their initial values, while the Frontier joint signal increases after the evaluator is switched. SFT is more erosive for the surface-form carriers, but it does not erase them; the Frontier strategy signal remains essentially unchanged after 1024 steps. The contrast suggests that persistence depends on what the carrier encodes: solution strategies are less easily overwritten by generic instruction tuning than lightweight response conventions.

We then replace ordinary post-training with explicit removal objectives. Balanced-carrier SFT reduces the Frontier effect to 7.2 pp after 2048 steps, and anti-watermark GRPO drives it further to 1.8 pp. Thus the mark is removable, but targeted optimization is substantially more effective than continued training. Schedules for Figure 5 are in Appendix G and L.

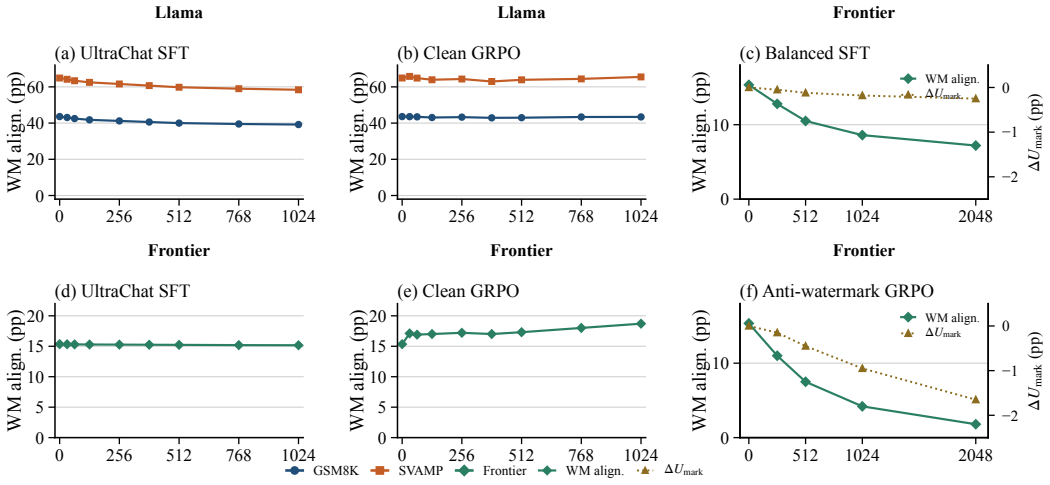


Figure 5: Persistence and removal. (a)–(e): 1024 steps; (c), (f): 2048 steps, with $\Delta U_{\text{mark}}$.

The main experiments use $\rho = 0.05$. Appendix M shows that weaker tilts transfer the code, while increasing $\rho$ beyond the main setting yields diminishing watermark gains at a higher utility cost.

The opening question was whether a score can leave a trace when the teacher never shows any text. It can. One registered preference is enough for the student to acquire it, and a joint code is still visible after the data, the student, or the evaluator changes. Rescoring the same outputs under other sign patterns shows that the student followed the enrolled code. The gap is large enough for a reference pool to flag the marked checkpoints and to leave the controls below the threshold. Ordinary later training does not wash the mark out, and adding it does not cancel the utility gain from distillation.

## 7 Conclusion and Discussion

Radioactive Feedback traces label-only distillation through scalar scores. The student acquires an interpretable watermark in answer form and in solution strategy. Across students, evaluators, and tasks, the mark transfers without teacher demonstrations, generalizes past the data used to write it, and supports calibrated detection. It follows the enrolled code. Continued training leaves it detectable; targeted removal reduces it further. Future work should move this mark into the evaluator's parameters.

## Reproducibility Statement

Proofs for Propositions 5.1 and 5.2 appear in Appendix A. The student–evaluator matrix, controls, and audit-labeling protocol are in Appendix C. Experimental splits and carrier definitions are in Appendix D and Appendix F. Shared training and audit settings are in Appendix E. The audit protocol is in Appendix I. Claude Opus 5 labels carrier and strategy alignment for the reported audits. Task utility is computed by official programmatic checkers. A 10% held-out subset of the quote-style labels, the *Therefore*/*So* labels, and the task-correctness labels was checked by two annotators, under the same adjudication protocol as the Frontier strategy labels (Appendix F). Sham decoy carriers and the training seeds are in Appendix N and F.

## Ethics Statement

This work studies provenance auditing for models trained from proprietary scalar feedback. Experiments use publicly available benchmarks and automated correctness checks; no new human-subject data were collected. Frontier training uses a public scalar scoring API; no evaluator-generated responses or chain-of-thought entered student training. The method is intended for accountability of evaluator services, not for covert surveillance of end users.

## AI Use Statement

In this work, we used generative AI tools to assist with manuscript writing and figure preparation. We reviewed all AI-assisted text and figures and take responsibility for the final content of this submission.

## References

Yossi Adi, Carsten Baum, Moustapha Cissé, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1615–1631, 2018.

Anthropic. How Claude's text watermark works. Anthropic News, August 2026. Published August 14, 2026.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. In *Proceedings of Thirty Seventh Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research*, pp. 1125–1139, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, 2023.

Junfeng Fang, Zhepei Hong, Mao Zheng, Mingyang Song, Gengsheng Li, Houcheng Jiang, Dan Zhang, Haiyun Guo, Xiang Wang, and Tat-Seng Chua. Rubric-based on-policy distillation. *arXiv preprint arXiv:2605.07396*, 2026.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan,

Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations*, 2025.

Thibaud Gloaguen, Robin Staab, Nikola Jovanović, and Martin Vechev. LLM fingerprinting via semantically conditioned watermarks. In *International Conference on Learning Representations*, 2026.

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.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. CATER: Intellectual property protection on text generation APIs via conditional watermarks. In *Advances in Neural Information Processing Systems*, 2022.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021b.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, et al. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida I. Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 17061–17084, 2023.

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

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26874–26901, 2024.

Kaixin Li. Verified TACO problems. Hugging Face dataset `likaixin/TACO-verified`, 2024.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic COde generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, 2023.

Jian Lou, Chenyang Zhang, Xiaoyu Zhang, and Kai Wu. PreferCare: Preference dataset copyright protection in LLM alignment by watermark injection and verification. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 4005–4019, 2025.

Leyi Pan, Aiwei Liu, Shiyu Huang, Yijian Lu, Xuming Hu, Lijie Wen, Irwin King, and Philip S. Yu. Can LLM watermarks robustly prevent unauthorized knowledge distillation? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13228–13251, 2025.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, 2021.

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. In *Advances in Neural Information Processing Systems*, 2023.

Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. In *Advances in Neural Information Processing Systems*, 2024.

Tom Sander, Hongyan Chang, Tomáš Souček, Tuan Tran, Valeriu Lacatusu, Sylvestre-Alvise Rebuffi, Alexandre Mourachko, Surya Parimi, Christophe Ropers, Rashel Moritz, Vanessa Stark, Hady Elsahar, and Pierre Fernandez. TextSeal: A localized LLM watermark for provenance & distillation protection. *arXiv preprint arXiv:2605.12456*, 2026.

Yash Savani, Asher Trockman, Zhili Feng, Yixuan Even Xu, Avi Schwarzschild, Alexander Robey, Marc Finzi, and J. Zico Kolter. Antidistillation sampling. In *Advances in Neural Information Processing Systems*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yiyang Shen, Lifu Tu, and Weiran Wang. Reinforcement learning-based semi-supervised knowledge distillation with LLM-as-a-judge. *arXiv preprint arXiv:2604.02621*, 2026.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM International Conference on Multimedia Retrieval*, pp. 269–277, 2017.

Jiashu Xu, Fei Wang, Mingyu Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3277–3306, 2024.

Yijie Xu, Aiwei Liu, Xuming Hu, Lijie Wen, and Hui Xiong. Mark your LLM: Detecting the misuse of open-source large language models via watermarking. *arXiv preprint arXiv:2503.04636*, 2025.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. In *Advances in Neural Information Processing Systems*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Guang Yang, Amir Ghasemian, Fengchen Liu, Zhong Wang, Ninareh Mehrabi, and Homa Hosseinmardi. Asking back: Interaction-layer antidistillation watermarks. *arXiv preprint arXiv:2605.16462*, 2026.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E. Weston. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 57905–57923, 2024.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, 2023.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

# Appendix

## A Proofs

For a response $y$ on prompt $x$ with assigned carrier $j(x)$, let $h_u(x,y) = \tilde{e}\, c_{j(x)}\phi_{j(x)}(x,y)$ with $h_u \in \{-1,0,+1\}$, and define

$$\delta_u(x,y) \;=\; \mathrm{clip}_{[0,R]}\big(r_0(x,y) + \rho R\, h_u(x,y)\big) - r_0(x,y), \tag{11}$$

so $r_u = r_0 + \delta_u$ and $|\delta_u| \le \rho R$. Throughout, $\delta_u$ is this clipped displacement. It equals $\rho R\, h_u$ only when $r_0 \in [\rho R,\, R - \rho R]$.

$$\big|\delta_u(x,y) - \delta_u(x,y')\big| \;\le\; 2\rho R. \tag{12}$$

**KL-regularized setup.** We analyze the idealized objective

$$\max_{\pi}\; \mathbb{E}_{y\sim\pi(\cdot|x)}[r(x,y)] - \beta^{-1}\mathrm{KL}\big(\pi(\cdot \mid x)\,\|\,\pi_0(\cdot \mid x)\big), \tag{13}$$

whose optimizer is the Gibbs policy

$$\pi_r^*(y \mid x) \;\propto\; \pi_0(y \mid x)\exp\big(\beta r(x,y)\big). \tag{14}$$

For the returned reward $r_u = r_0 + \delta_u$, this gives

$$\pi_u^*(y \mid x) \;=\; \frac{\pi_{\mathrm{cl}}^*(y \mid x)\exp\big(\beta\,\delta_u(x,y)\big)}{\mathbb{E}_{y'\sim\pi_{\mathrm{cl}}^*}\big[\exp\big(\beta\,\delta_u(x,y')\big)\big]}. \tag{15}$$

*Proof of Proposition 5.1.* The unique maximizer of (13) is the Gibbs policy (14) (Ziegler et al., 2019; Rafailov et al., 2023). Writing $Z(r) = \mathbb{E}_{y\sim\pi_0}[\exp(\beta r(x,y))]$ for the partition function,

$$\pi_{\mathrm{cl}}^*(y \mid x) \;=\; \pi_0(y \mid x)\,\exp\big(\beta r_0(x,y)\big)/Z(r_0), \tag{16}$$

$$\pi_u^*(y \mid x) \;=\; \pi_0(y \mid x)\,\exp\big(\beta r_u(x,y)\big)/Z(r_u). \tag{17}$$

Substitute $r_u = r_0 + \delta_u$ and rearrange:

$$\pi_u^*(y \mid x) \;=\; \pi_{\mathrm{cl}}^*(y \mid x)\,\exp\big(\beta\,\delta_u(x,y)\big)\,\frac{Z(r_0)}{Z(r_u)}.$$

The prefactor $Z(r_0)/Z(r_u)$ equals the reciprocal of $\mathbb{E}_{y'\sim\pi_{\mathrm{cl}}^*}[\exp(\beta\,\delta_u)]$, which is (15). The log-odds identity (9) follows by taking the ratio of (15) at $y$ and $y'$; the normalizing constants cancel. If $\delta_u$ is constant on the pair, the right-hand side is zero. When $r_0 \in [\rho R,\, R - \rho R]$, $\delta_u = \rho R\, h_u$ and the identity reduces to a multiple of $h_u(y) - h_u(y')$. □

*Proof of Proposition 5.2.* Because $|\delta_u| \le \rho R$, clipping included, the change in any pairwise gap lies in $[-2\rho R, 2\rho R]$, which is (10). If $|r_0(x,y) - r_0(x,y')| > 2\rho R$, this increment cannot change the sign of $r_u(x,y) - r_u(x,y')$. The pass/fail claim is the same bound on the two-point reward $\{0,R\}$, for which $\rho < 1/2$. □

## B Code registration and routing

**Applicability and code generation.** Let $m(x,y) \in \{0,1\}^d$ mark applicable coordinates. It is computed without the key, and every reward or audit inner product uses $m(x,y)\odot\phi(x,y)$. A secret seed can generate a fixed code from registered carrier identifiers:

$$c_{u,j} = 2\,\mathrm{PRF}_{\mathrm{bit}}(s_u, \mathrm{id}_j) - 1. \tag{18}$$

The identifiers do not depend on prompts, so the code is reused across training and audit. Alternatively, the provider can register an explicitly chosen sign vector; the experiments use the latter. A two-coordinate example is quote style together with the closing marker *Therefore*/*So*: $(+1,-1)$ can favor double quotes and *So*; flipping both signs favors the opposite alternatives. The Frontier Test instead uses problem-solving carriers (Table 9).

**Routing.** Each training or audit prompt is assigned exactly one carrier $j(x)$ before scoring, so each response receives one keyed scalar reward. Sham applies analogous eligibility and one-carrier assignment to a disjoint decoy carrier set, matched in carrier count, exposure target, and perturbation budget; it replaces the carrier set and code, not only the code. Registered-code auditing never uses the decoy coordinates.

**Bounded shift.** With $R = 100$, $\rho = 0.05$, and maximal rubric strengths, an eligible response with a defined carrier value receives at most a $\pm 5$ point shift; flipping the registered code flips the sign of that shift. Given eligibility, the keyed term depends only on that response's carrier values and the registered code, not on which other candidates appear in the submitted group.

## C Experimental protocol

**Student–evaluator matrix.** RQ1 uses Qwen3-30B-A3B as the scalar evaluator for Llama-3-8B-Instruct and Qwen2.5-Coder-1.5B-Instruct. The primary Frontier setting uses Qwen3-30B-A3B as the student and GPT-6 Astra as the evaluator. RQ2 broadens the comparison to nine student–evaluator combinations. Llama and Qwen2.5 are each paired with Qwen3-30B-A3B, GPT-5.6 Sol, and GPT-6 Astra. For the Qwen3-30B-A3B student, the open-weight evaluator is the larger Qwen3-235B-A22B, alongside GPT-5.6 Sol and GPT-6 Astra. The open-weight runs serve as transfer controls; the provenance threat model remains provider-controlled scalar feedback. RQ3–RQ4 use the selected Llama/Qwen3-30B-A3B and Qwen3-30B-A3B/GPT-6 Astra settings.

**Controls and seeds.** The four conditions are *Base* (the unmodified student before RL), *Clean* (ordinary feedback), *Sham* (a strength- and exposure-matched preference over a disjoint decoy carrier set), and *Marked* (the enrolled carriers and registered code **c**). Trained arms share the GRPO recipe, data split, group size, and budget within a setting. Clean provides the baseline for watermark effects and task utility; Sham supplies matched decoy models for the detection reference pool. RQ1 single-carrier and RQ2 joint runs each use four independent GRPO training seeds per setting. Each of the nine RQ2 student–evaluator combinations trains under the registered four-carrier code for that student setting, with matched Clean checkpoints. Base is evaluated on the same held-out tasks for the distillation contrasts.

**Audit labels and correctness.** Task utility and the correctness bit $q_i$ use the official programmatic checks for each benchmark: execution harnesses for code and final-answer checking for mathematics. Claude Opus 5 supplies only the carrier or strategy label used in $\phi$; it is not the training evaluator and does not score task utility. The reported audits use one response per prompt. For the Frontier summary in Table 3, the pre-designated primary audits are CodeContests and Omni-MATH; Table 2 additionally reports four further held-out benchmarks.

## D Experimental data splits

Table 4 lists the training and held-out evaluation sets. The RQ2 evaluator grid uses the same data split for a given student setting; only the scalar evaluator changes. GSM8K uses the official OpenAI split (Cobbe et al., 2021): mathematics training uses the 7,473 training problems, and GSM8K watermark audit and utility evaluation use the 1,319 test problems (8,792 in total). Code audits use HumanEval+ and MBPP+ (Liu et al., 2023); the additional mathematics audit is SVAMP (Patel et al., 2021).

Table 4: Training and held-out audit sets. Utility is measured on the same held-out prompts.

| Setting | Training | Held-out audit |
|---|---|---|
| Llama-3-8B-Instruct;<br>Qwen2.5-Coder-1.5B-Instruct | TACO-verified 5k;<br>GSM8K train | HumanEval+, MBPP+;<br>GSM8K test, SVAMP |
| Qwen3-30B-A3B Frontier | TACO-verified 5k;<br>OpenR1-Math 5k | carrier-compatible subsets of<br>CodeContests, APPS,<br>LiveCodeBench;<br>Omni-MATH, MATH-500,<br>OlympiadBench |

## E Training and audit configuration

Table 5 gives the GRPO configuration shared by Clean, Sham, and Marked in the initial comparison. Base is the initial student checkpoint evaluated without further training. Training makes one shuffled pass over the deduplicated split without prompt replacement. Each GRPO update uses a batch of 50 prompts (the last may be smaller) and $G = 16$ student-generated responses per prompt, giving $\lceil N_{\text{train}}/50 \rceil$ updates. For one group, rewards are standardized as

$$\widehat{A}_i = \frac{r_{u,i} - \bar{r}_u}{\sqrt{G^{-1}\sum_{g=1}^{G}(r_{u,g} - \bar{r}_u)^2 + 10^{-6}}}.$$

The KL reference is the frozen initial student checkpoint. The clean reward is the evaluator's scalar score only. Reported carrier and strategy labels are produced by Claude Opus 5; those labels do not enter the training reward. Task utility, including the correctness bit $q_i$, comes from official programmatic checkers. Task-correctness labels on a 10% held-out subset of the audited responses were checked by two annotators; disagreements were adjudicated by discussion, with a third annotator breaking ties when consensus was not reached. The hidden rubric returns only the final scalar reward to the learner. Its keyed shift may vary by eligible candidate but is bounded by $\rho R = 5$ points. The returned reward is clipped to the evaluator's $[0, 100]$ scale.

Table 5: Initial training and audit configuration.

| Setting | Value |
|---|---|
| Initial GRPO training | one epoch; each training prompt used once |
| Responses per prompt / prompt batch | $G = 16$ / 50 |
| Optimizer / learning rate | AdamW / $1 \times 10^{-6}$ |
| KL reference / coefficient | frozen initial student / 0.04 |
| Clean evaluator scale | $[0, 100]$ |
| Maximum keyed shift / quality floor | $\rho = 0.05$ ($\rho R = 5$), $r_{\text{min}} = 60$ |
| Code-execution reward | none |
| Carrier and strategy labels | Claude Opus 5 |
| Task utility ($q_i$, Pass@1, accuracy) | official programmatic checker |
| Maximum new tokens | 2048 (GRPO rollouts and audit) |
| Audit prompts / responses per prompt | distinct prompts, $\leq 500$ per carrier / 1 |
| Audit decoding | temperature 0 |

For RQ4, every distinct audit prompt remains in the denominator. A response contributes a signed carrier value only when it is correct and classifiable; incorrect, unclassifiable, and inapplicable responses contribute zero. The correct-and-classifiable conditioning used for RQ1/RQ2 transfer summaries is therefore not the RQ4 detection denominator.

**Code audit prompt budget.** Every audit prompt is used once. Decoding is temperature 0, so a repeated prompt would not have been a new draw. Training–audit deduplication stays exact on prompt text (below).

Before training, every whitespace-normalized training prompt is compared against every audit prompt; exact matches are removed from the training split, as is standard train/audit hygiene.

## F Carrier definitions

The reward path has no extractor. The preferred alternative is named in the evaluator instruction, and the score that comes back is the evaluator's own score. The definitions below state the two sides of each carrier. At audit, a response counts as $+1$, $-1$, or abstaining under those same definitions. Mixed or undefined cases do not enter the aligned-cell rate.

**Code carrier pool.** Table 6 lists the original 32 two-sided Python candidates. Each item is one bipolar contrast. The pool includes `for`/`while` and augmented assignment (`+=` versus `x = x + y`); both were screened. The binding constraint is not whether a contrast can be defined, but whether both polarities arise naturally on the same problem at comparable quality and matched length. The RQ1 code run uses quote style from this list.

Table 6: Screened Python code carriers.

| Category | Contrasts |
|---|---|
| Function structure | helper / inline; early return / single terminal return; recursion / iteration; return expression / temporary; type annotations / none |
| Loop style | `enumerate` / index loop; `zip` / index loop; `for` / `while`; direct / indexed iteration; `continue` / nested `if` |
| Comprehensions | list comprehension / loop+`append`; comprehension / `map`/`filter`; generator / list reduction; `sum()` / accumulator; `any`/`all` / explicit loop |
| Data structures | `Counter`/`defaultdict` / manual counts; `set` / `list` membership; heap / sort; `deque` / `list` queue; `tuple` / `list` literal membership |
| Expressions | ternary / `if-else`; direct Boolean / branched return; truthiness / `len` check; `+=` / `x = x + y` |
| Strings | `join` / loop concatenation; f-string / `.format()`; double / single quotes |
| Other syntax | `sorted()` / in-place `sort`; `[::-1]` / `reversed()`; `dict()` / `{}`; `list()` / `[]`; `range(n)` / `range(0,n)` |

**Code: quote style.** Claude Opus 5 labels each audited program $+1$ for double quotes, $-1$ for single quotes, or undefined. These labels are the reported carrier values. A 10% held-out subset was checked by two annotators; disagreements were adjudicated by discussion, with a third annotator breaking ties when consensus was not reached.

**Mathematics: final-paragraph marker.** Claude Opus 5 labels each audited solution $+1$ when the closing is *Therefore*, $-1$ when it is *So*, or undefined. These labels are the reported carrier values. A 10% held-out subset was checked by two annotators; disagreements were adjudicated by discussion, with a third annotator breaking ties when consensus was not reached.

**Joint Llama/Qwen2.5 carriers.** Table 7 lists the four enrolled carriers for the Llama-3-8B-it and Qwen2.5-Coder-1.5B-it joint checkpoints in Table 3. Qwen2.5 uses the same carrier set and registered code; only the student initialization differs.

Table 7: Four-carrier set for joint Llama and Qwen2.5 training.

| Domain | Carrier | Source |
|---|---|---|
| Code | Quote style (double vs. single) | RQ1 |
| Code | `for` vs. `while` loop style | Screened pool |
| Math | Final-paragraph *Therefore*/*So* marker | RQ1 |
| Math | Ternary vs. `if-else` branch | Screened pool |

**Math code-block carrier.** The ternary/`if-else` carrier applies only when a response contains a valid Python code block. On GSM8K/SVAMP, Llama often emits a short final program; we parse the last `SyntaxTree`-valid block and classify ternary versus explicit `if`/`else`. Responses without such a block abstain on this coordinate. Coverage is lower than for the closing-marker carrier but nonzero on the math training split.

**Sham decoy carriers.** Sham replaces both the registered code and the carrier set. Table 8 lists the four decoy carriers for Llama/Qwen2.5 joint training; each is disjoint from Table 7 and matched in domain count, exposure target, and perturbation budget. Single-carrier RQ1 Sham uses one decoy per domain: `f-string`/`.format()` on code and closing *Hence*/ *Thus* on math. Frontier Sham uses four decoy strategy carriers disjoint from Table 9: divide-and-conquer vs. iterative accumulation; Floyd–Warshall vs. repeated DSU queries; closed-form vs. brute enumeration; pure angle chasing vs. complex-number geometry. Registered-code auditing never scores decoy coordinates.

Table 8: Decoy carrier set for joint Llama and Qwen2.5 training.

| Domain | Decoy carrier | Source |
|---|---|---|
| Code | `f-string` vs. `.format()` | Screened pool |
| Code | List comprehension vs. loop+`append` | Screened pool |
| Math | Closing *Hence*/*Thus* marker | Custom (text) |
| Math | Direct Boolean vs. branched return | Screened pool |

**Frontier Test carriers.** The Qwen3-30B-A3B student does not reuse the surface-form carriers above, so the Frontier result is not a quote-style or *Therefore*/*So* preference. Table 9 lists the four two-sided *problem-solving* carriers and the data splits.

**Frontier strategy preference.** There is no strategy extractor in the reward path. The evaluator instruction states the registered preference and returns one scalar score for each of the $G = 16$ candidates; recognizing the preferred side changes the emitted score. The definitions below specify the two sides used by all evaluators in the RQ2 grid. Mixed or unclassifiable solutions are not counted in the aligned-cell rate. Audit reads held-out answers against the same definitions. Tags on a 10% held-out subset from each primary audit domain were checked by two annotators; disagreements were adjudicated by discussion, with a third annotator breaking ties when consensus was not reached.

- **Top-down DP $\leftrightarrow$ bottom-up DP:** $+1$ if the solution fills a DP table recursively or top-down with memoization before combining subproblems; $-1$ if it iterates over states in increasing problem size without recursion.
- **Traversal $\leftrightarrow$ DSU:** $+1$ if connectivity is resolved by BFS/DFS on an explicit graph; $-1$ if by a disjoint-set union–find structure.
- **Direct counting $\leftrightarrow$ inclusion–exclusion:** $+1$ if the answer is obtained by direct enumeration or product rule; $-1$ if by inclusion–exclusion or generating functions.
- **Synthetic $\leftrightarrow$ coordinate geometry:** $+1$ if the proof is synthetic (angle/chord reasoning without coordinates); $-1$ if it introduces a coordinate system or vectors.

Training uses two fixed $5{,}000$-prompt subsets, not the full releases. The two code carriers share a $5{,}000$-prompt draw from TACO-verified (Li et al., 2023; Li, 2024). The two mathematics carriers share a $5{,}000$-prompt draw taken at random from OpenR1-Math-46k-8192 (Yan et al., 2025). No training problem inside these draws is removed by a carrier filter. The keyed adjustment is applied only to a response that meets the quality floor and exposes a classifiable value of its assigned carrier; every other response keeps its internal rubric score. Watermark transfer is audited on carrier-compatible subsets of CodeContests (Li et al., 2022), APPS (Hendrycks et al., 2021a), and LiveCodeBench (Jain et al., 2025) for code, and of Omni-MATH (Gao et al., 2025), MATH-500 (Hendrycks et al., 2021b), and OlympiadBench (He et al., 2024) for mathematics. A problem enters the subset for carrier $j$ when the dataset's own reference solution expresses exactly one side of $j$ under the definitions above. Solutions that mix both sides or express neither side are excluded. The reference solution is not a student rollout. This filter is fixed before Marked and Clean training and does not use Marked-versus-Clean outcomes. These audit datasets are not used to write the watermark. The primary Frontier setting uses GPT-6 Astra, while Table 3 additionally varies the evaluator. No evaluator-generated answers or chain-of-thought enter student training. Table 2 reports watermark transfer and utility on the held-out audit datasets.

Table 9: Frontier training and held-out audit sets. Code training is a $5{,}000$-prompt TACO-verified subset (Li et al., 2023; Li, 2024); mathematics training is a random $5{,}000$-prompt subset of OpenR1-Math-46k-8192 (Yan et al., 2025). Each audit cell is the carrier-compatible subset of that benchmark.

| Domain | Carrier | Training | Held-out audit |
|---|---|---|---|
| Code | Top-down DP $\leftrightarrow$ bottom-up DP<br>Traversal $\leftrightarrow$ DSU | TACO-verified 5k | CodeContests, APPS,<br>LiveCodeBench |
| Math | Direct counting $\leftrightarrow$<br>inclusion–exclusion<br>Synthetic $\leftrightarrow$ coordinate geometry | OpenR1 5k | Omni-MATH, MATH-500,<br>OlympiadBench |

## G Continued-training details

Figure 5 plots the watermark effect defined in Metrics. At each logged step the matched Clean checkpoint is trained on the same schedule as the Marked checkpoint, and the plotted value is $\text{Marked}_t - \text{Clean}_t$. Panel (a) continues with instruction SFT on UltraChat (Ding et al., 2023) without keyed feedback or an evaluator. Llama GSM8K/SVAMP WM align. changes from $+43.6/+64.9$ to $+39.2/+58.4$ by step 1024; Frontier stays near its step-0 gap ($+15.35 \rightarrow +15.17$). Panels (b) and (e) continue with clean GRPO, and each arm uses its own step-0 checkpoint as the KL reference. The Llama panel keeps the Qwen3-30B-A3B evaluator, while the Frontier panel continues under GPT-5.6 Sol. Llama ends at $+43.4/+65.5$, and Frontier at $+18.71$.

## H Frontier pooled alignment baseline

The Qwen3-30B-A3B / GPT-6 Astra entry in Table 3 reports the Frontier joint watermark effect. Tables 2 and 3 aggregate four training seeds as mean $\pm$ standard deviation.

## I Reference calibration protocol

This appendix specifies the reported evaluation in Section 6.4. Carriers, the definitions in Appendix F, audit prompts, decoding, and the enrolled code are fixed before scoring, separately for each student–evaluator setting.

**Reference composition.** Each pool contains 50 independently trained Clean checkpoints and 50 independently trained Sham checkpoints using decoy carrier sets disjoint from the enrolled carriers. The 100 runs use disjoint random seeds and the same architecture, data split, and training budget as the audited student; none is an intermediate save of another run. These 100 same-family checkpoints define the calibration pool and rank denominator. Separately, we score 20 independently trained cross-family controls. They follow the same data split, GRPO recipe, and training budget as the same-family references and are finished runs, not intermediate checkpoints; only the student family changes. For the Llama-3-8B-Instruct audit, these controls are Qwen2.5-7B-Instruct models; for the Qwen3-30B-A3B Frontier audit, they are Llama-3-8B-Instruct models. Cross-family controls do not enter calibration or the rank denominator. No control model shares a training seed or key with the four marked checkpoints, and pools are not combined across settings. Each marked checkpoint is scored as a separate suspect. Table 3 summarizes the broader joint-transfer grid; RQ4 uses the selected Llama/Qwen3-30B-A3B and Qwen3-30B-A3B/GPT-6 Astra settings, with detection computed from individual checkpoint scores.

**Scoring.** Apply Equation (5) to every checkpoint and compute its suspect rank using Equation (6). Fix $\alpha = 1\%$ before suspect evaluation and use the non-interpolated upper empirical quantile defined in Section 4.3. With 100 references, $\gamma$ is the reference maximum; exceeding it gives $p_u = 1/101$ and passes Equation (7). The pool has fixed Clean and Sham strata, 50 runs each.

**Cross-family controls.** The cross-family checkpoints receive no marked feedback from the enrolled evaluator and are held out from calibration. Table 10 shows that their maximum $100T_u$ remains below the same-family calibration threshold in each setting.

Table 10: Held-out cross-family controls. Max. $100T_u$ is the largest score among the 20 controls; $\gamma$ is calibrated on the 100 same-family references only.

| Audit student | Enrolled evaluator | Cross-family component | Max. $100T_u$ | $\gamma$ |
|---|---|---|---|---|
| Llama-3-8B-it | Qwen3-30B-A3B | Qwen2.5-7B-Instruct (20) | 1.00 | 1.15 |
| Qwen3-30B-A3B | GPT-6 Astra | Llama-3-8B-Instruct (20) | 1.10 | 1.50 |

**Reported outputs.** Table 11 reports full-budget detection. The full-budget score panels appear in Figure 4.

Table 11: Black-box detection results.

| Setting | Detected | $p_u$ |
|---|---|---|
| Llama / Qwen3-30B-A3B | $4/4$ | $1/101$ |
| Frontier / GPT-6 Astra | $4/4$ | $1/101$ |
| Both settings | $8/8$ | — |

## J Audit query budget analysis

Section 6.4 uses the detection margin

$$\Delta_q^{\text{detect}} = \min_{k \in \mathcal{K}} T_k(q) - \max_{m \in \mathcal{N}_{\text{null}}} T_m(q) \tag{19}$$

to study query efficiency. For each nominal per-carrier budget $q \in \{100, 200, 300, 500\}$, we use up to $q$ distinct prompts from each carrier's available audit pool, without replacement. The plotted budget is the nominal total $4q$. We then recompute every marked and null checkpoint score, take the null reference maximum, and form $\Delta_q^{\text{detect}}$ from the weakest marked checkpoint. Figure 6 reports these margins. At a nominal budget of 400 the margins remain positive in both settings (Llama: $+2.25$ pp; Frontier: $+0.75$ pp).

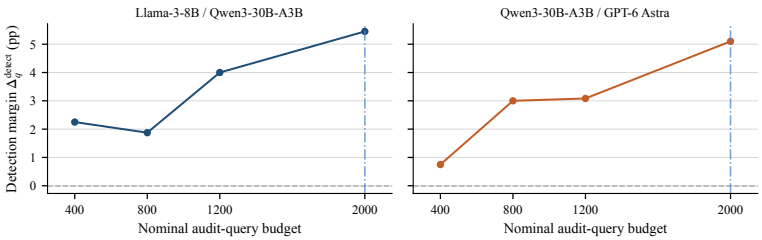


Figure 6: Detection margin versus nominal audit-query budget.

Figure 7 shows how the mean marked $100T_u$ estimate converges with budget. At a nominal budget of 400, the Frontier mean is within 0.5 pp of its full-budget value (6.25 versus 6.70). The Llama mean is 1.88 at 400, reaches 7.72 by 800, and then stays near its full-budget value of 7.54.

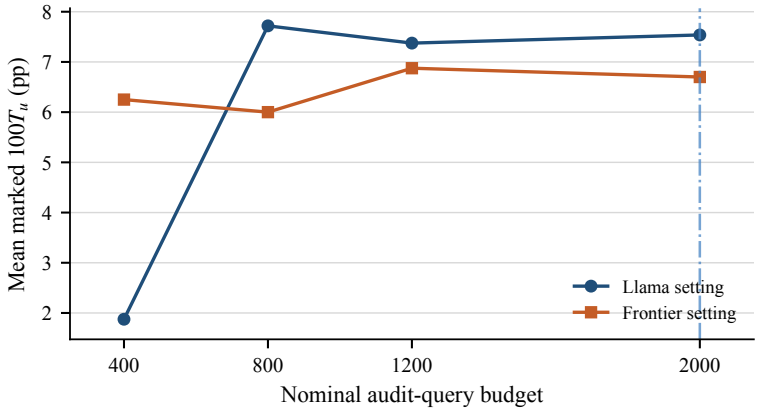


Figure 7: Marked audit score versus nominal audit-query budget. Frontier converges early; Llama is near its full-budget value by a nominal budget of 800.

## K Registered-key attribution protocol

Section 6.3 tests whether inherited alignment is specific to the enrolled code $\mathbf{c}^\star$. The four marked checkpoints in a setting are seeds of that one code. They are not retrained under other keys. We freeze each seed and score the same responses under alternative length-4 sign vectors at Hamming distances 1 through 4 from $\mathbf{c}^\star$. Registered owner codes remain at pairwise distance at least 2, so one enrolled owner is not a one-bit neighbor of another. The distance-1 points are one-bit probes of $\mathbf{c}^\star$.

The dashed threshold is $\gamma(\mathbf{c}^\star)$, the maximum score of the reference pool when that pool is scored with the enrolled code. Alternative codes are compared with this same threshold. Figure 3 plots $100T_u$ against $d_H(\mathbf{c}^\star, c)$.

## L Removal stress tests

Section 6.5 extends the reported ordinary SFT/GRPO continuation with two stress tests on Frontier joint checkpoints: *balanced-carrier SFT* mixes prompts that expose all carriers with uniform frequency but removes keyed rewards, and *explicit anti-watermark GRPO* applies the reverse of the enrolled code on eligible responses while keeping the clean utility evaluator. Both runs share the RQ5 step schedule $0, 256, 512, 1024, 2048$. The matched Clean arm takes the same steps without the removal objective, and WM effect at step $t$ is $\text{Marked}_t - \text{Clean}_t$, reported together with $\Delta U_{\text{mark}}$. Figure 5 reports WM effect and $\Delta U_{\text{mark}}$ along this schedule.

## M $\rho$ sweep ablation

All main experiments use $\rho = 0.05$ and $r_{\min} = 60$. The sweep retrains Frontier joint checkpoints at $\rho \in \{0, 0.01, 0.025, 0.05, 0.10\}$ on one seed held fixed across the grid (Figure 8). That seed is not the seed in Table 3. On the sweep seed, WM effect is 0 at $\rho = 0$, 4.2 pp at $\rho = 0.01$, and 11.5 pp at $\rho = 0.025$; at the main setting $\rho = 0.05$ it reaches 15.35 pp. Below $\rho = 0.05$, $\Delta U_{\text{mark}}$ fluctuates within $\pm 0.1$ pp on this seed rather than sitting exactly at zero. Increasing to $\rho = 0.10$ adds only 1.85 pp of WM while incurring $\Delta U_{\text{mark}} = -0.35$ pp. The Qwen3-30B-A3B / GPT-6 Astra entry in Table 3 averages over four training seeds at the same $\rho = 0.05$, with $\Delta U_{\text{mark}} = +0.5$ pp.

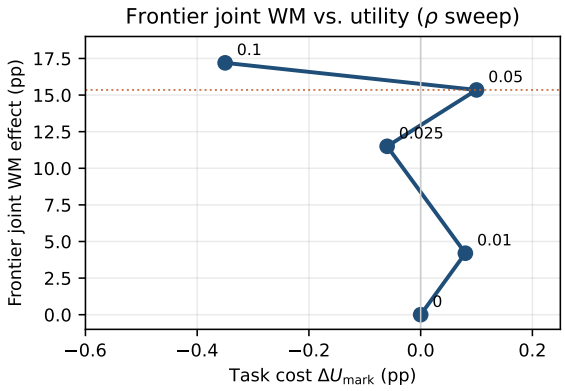


Figure 8: Watermark–utility trade-off across keyed strengths.

## N Implementation and checkpoint map

This appendix records details needed to reconstruct the reported runs.

**Training budget.** Initial GRPO uses one epoch over the deduplicated training split (Table 4), with $G = 16$ rollouts per prompt and prompt batches of 50 (Table 5). Each optimizer step therefore scores 800 responses ($50 \times 16$). Table 12 turns that rule into a rollout budget. $N_{\text{train}}$ counts each training prompt once, before the exact audit-prompt deletion in Appendix E, so the step counts are upper bounds. Llama and Qwen2.5 joint runs use TACO-verified (5,000) plus the GSM8K training split (7,473), hence $N_{\text{train}} = 12{,}473$. Frontier runs use TACO-verified (5,000) plus a random 5,000-prompt subset of OpenR1-Math-46k-8192, hence $N_{\text{train}} = 10{,}000$. A same-family reference pool is 100 independent runs of that setting. The 40 cross-family controls use the audited setting's split, 20 runs each.

RQ5 continuation curves start from the step-0 marked checkpoint in each setting: Llama-math uses the RQ1 single-carrier marked checkpoint; Frontier uses the RQ2 joint marked checkpoint. RQ4

Table 12: Compute manifest for one initial GRPO epoch ($G = 16$, prompt batch 50). Rollouts equal $G \times N_{\text{train}}$ per run. Step counts use $\lceil N_{\text{train}}/50 \rceil$.

| Run class | Runs | $N_{\text{train}}$ | Steps / run | Rollouts |
|---|---|---|---|---|
| Llama / Qwen2.5 joint | 1 | 12,473 | 250 | 199,568 |
| Frontier joint | 1 | 10,000 | 200 | 160,000 |
| Llama-setting reference pool | 100 | 12,473 | 250 | 19,956,800 |
| Frontier-setting reference pool | 100 | 10,000 | 200 | 16,000,000 |
| Cross-family controls | 40 | — | — | 7,191,360 |

audits the four marked checkpoints from joint training on students from different families in the two settings (Llama and Qwen3-30B-A3B), not the RQ5 continuation checkpoints.

**Seeds and arms.** RQ1 single-carrier and RQ2 joint runs each use four independent GRPO training seeds per setting: 1, 42, 2026, and 2025. Each seed trains a matched Clean checkpoint. Base is the initial no-RL student checkpoint for that setting, evaluated on the same held-out tasks; it requires no separate training run or seed. Tables 1–3 subtract the paired Clean checkpoint inside each seed, then report the mean and standard deviation of those four differences. The reference pool contains 50 Clean and 50 Sham checkpoints with seeds disjoint from all marked suspects. Sham arms use the decoy carriers in Appendix F; Marked arms use the enrolled carriers and registered code for that setting.

**Registered codes.** Joint Llama/Qwen2.5 training uses an explicitly registered four-sign code on the carriers in Table 7; the same code is used across Qwen3-30B-A3B, GPT-5.6 Sol, and GPT-6 Astra. Frontier training uses a separate four-sign code on Table 9, shared across Qwen3-235B-A22B, GPT-5.6 Sol, and GPT-6 Astra in Table 3. Single-carrier RQ1 uses one enrolled carrier per domain with code $(+1)$ on the positive polarity defined in Appendix F.