AUDITGYM: CAN BLACK-BOX AUDITS VERIFY THE MODEL BEHIND AN LLM API?

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) are increasingly accessed through APIs and embedded in agent systems, yet users have limited evidence that the API actually serves the model it claims. Existing black-box behavioral audits approach this problem through equality testing, model identification, and fingerprinting. However, their outcomes are difficult to interpret and compare: a detected discrepancy may reflect either a different model or a change in how the same model is deployed. We introduce **AuditGym**, a benchmark for systematically evaluating LLM identity verification under controlled deployment changes. AuditGym separates *model specification*, including checkpoints, learned adapters, and quantization, from *generation policy* and the *interaction layer*, and tests whether auditors reject changes to the former while accepting changes to the latter. We evaluate four auditing approaches across seven change families and 19 checkpoints, including models running inside real agent harnesses. All evaluated methods achieve only limited reliability for the identity verification: all four detect checkpoint substitution but miss finer model changes such as adapters and quantization. More seriously, every method rejects unchanged models under certain generation, watermarking, system-prompt changes, or agent harnesses. With these results, we reveal that current auditors confuse model changes with deployment changes, and AuditGym provides a controlled foundation towards closing this gap. Our anonymous codebase is available at `https://anonymous.4open.science/r/auditgym_iclr-BC1E/`.

1 INTRODUCTION

Large language models (LLMs) are increasingly accessed through APIs and embedded in tool-using agents (Gao et al., 2025; Yao et al., 2023; Wu et al., 2024). Users choose an advertised model for its expected capabilities, reliability, and cost, yet the service exposes little evidence of which model it runs. This lack of transparency allows a provider to substitute another checkpoint, apply a learned adapter, or deploy a quantized version of the model without the user noticing the change (Gao et al., 2025; Zhu et al., 2026; Cai et al., 2025). Black-box audits address this problem by comparing the service's observable behavior with a trusted reference, that is, evidence obtained from the claimed model itself. These audits rely on different forms of evidence, including response distributions (Gao et al., 2025), reference-model token scores (Zhu et al., 2026), tool-selection behavior (Wang et al., 2026), and learned response fingerprints (Pasquini et al., 2025).

However, decoding settings, system instructions, and agent harnesses can change responses without changing the model specification (Zhu et al., 2026; Wang et al., 2026; Yang et al., 2024b). A rejection of the model claim is therefore ambiguous, since it may reflect a different model or the same model *deployed differently*. A non-rejection is equally inconclusive, because it may hide a small model change such as an adapter or quantization. LLM identity verification therefore requires detecting changes to the claimed model while remaining robust to permitted changes in how it is deployed. Existing methods, however, test properties such as distributional equality or identification among known models, which do not address this verification objective.

Current evaluations cannot establish whether auditors meet this requirement, for two reasons. First, the verification target is underspecified. Without an explicit claim stating which differences should lead to rejection, detection power and robustness cannot be measured together. Detecting checkpoint substitution, for instance, does not show sensitivity to adapters or quantization. Likewise, detecting

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.

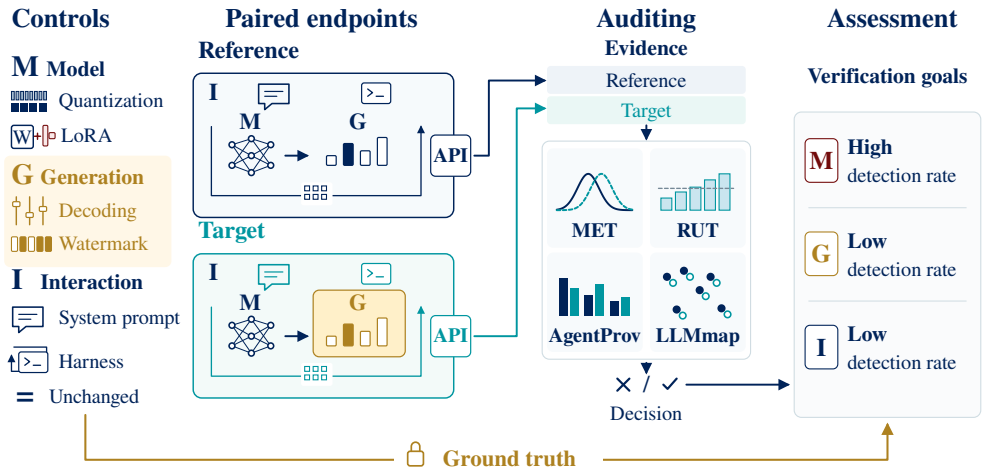


Figure 1: **AuditGym testbed.** Controlled reference–target endpoints separate model specification M, generation policy G, and interaction layer I. Each auditor receives evidence according to its own protocol, while deployment ground truth is withheld and used only for assessment. The verification goals are high detection rates for changes to M and low detection rates for changes only to G or only to I. Detection denotes a discrepancy flagged by each auditor.

behavioral differences does not show robustness to deployment changes. Second, decisions cannot be attributed to known causes, because commercial endpoints typically lack verified deployment ground truth (Gao et al., 2025; Wang et al., 2026). Controlled studies provide this ground truth, but their differing intervention sets and protocols prevent comparison across methods (Gao et al., 2025; Zhu et al., 2026; Cai et al., 2025; Hu et al., 2025; Fang et al., 2026). These gaps become even more pronounced when models run inside agent harnesses, which alter observable behavior through the insertion of context, the use of tools, and feedback from the environment. Addressing both gaps therefore requires a common controlled testbed that preserves each method's access requirements.

To address these gaps, we introduce **AuditGym**, a controlled benchmark for LLM identity verification, including models running inside agent harnesses. We present the overview of AuditGym in Figure 1. Evaluating identity verification requires specifying which parts of a deployment define the model's identity and which are permitted to vary. We decompose each deployment into three components. The *model specification* M covers the checkpoint, learned adapters, and numerical representation. The *generation policy* G selects output tokens, for example, through decoding or watermarking. The *interaction layer* I constructs model inputs and coordinates tool use, for example, through system prompts or agent harnesses. A model claim concerns learned parameters and their numerical representation, whereas generation and interaction settings are deployment choices applied to an unchanged model. An auditor should therefore reject when M changes but not when only G or I change. Each reference-target pair differs in a known factor, which is withheld from the auditors and used to assess their decisions.

Using AuditGym, we evaluate MET, RUT, AgentProv, and LLMmap. All four auditors detect checkpoint substitution almost perfectly, whereas MET, RUT, and AgentProv often miss adapters and quantization, especially INT8. LLMmap detects every model change but at the cost of rejecting some unchanged deployments. Decoding, watermarking, and system prompts each lead to rejection of unchanged models in some settings. Inside agent harnesses, all four auditors reliably distinguish different models when reference and target share the harness, but nearly always reject the same model when its reference is collected from the model alone. These results show that current black-box audits do not achieve reliable LLM identity verification, since each auditor misses model changes, rejects unchanged models under deployment changes, or both. Reliable identity verification therefore requires auditors that separate model changes from deployment changes, and AuditGym provides a controlled setting for developing and evaluating them.

In summary, we make the following contributions:

1. We define LLM identity verification through model specification (M), generation policy (G), and interaction layer (I), setting high detection rates for changes to M and low detection rates for changes to G or I.
2. We propose AuditGym, a benchmark that pairs deployments with known differences across seven change families and 19 checkpoints, and release all code and collected responses.
3. We show that current auditors miss adapter and quantization changes or reject unchanged deployments, and that references collected from the model do not transfer to agent harnesses.
4. We release AuditGym as a controlled testbed for the community to develop more reliable LLM identity verification methods, including evaluation under real agent harnesses.

2 Background and Related Work

Black-box model verification and fingerprinting. Black-box audits assess a target service through observable responses and trusted information about the claimed model. MET compares response distributions using maximum mean discrepancy (Gretton et al., 2012; Gao et al., 2025), RUT tests randomized ranks of reference-model response scores (Zhu et al., 2026), and AgentProv compares distributions of tool-selection decisions (Wang et al., 2026). LLMmap constructs learned response fingerprints to identify a model from a known collection (Pasquini et al., 2025), while FLIPS distinguishes deployment instances using statistical features of generated pseudo-random sequences (Gurvan et al., 2026). ESF and RESF construct modification-sensitive probes using reference parameters and gradients before querying targets through black-box access (Bai et al., 2025; Hu et al., 2025). RESF additionally accounts for temperature rescaling (Hu et al., 2025). Further work examines initialization lineage and geometric signatures in output log-probabilities (Tong et al., 2026; Finlayson et al., 2026). We evaluate representative methods from black-box model verification and fingerprinting approaches.

Evaluations of black-box auditing. Existing auditing studies evaluate model changes such as quantization and substitution, including model mixtures and benchmark evasion (Gao et al., 2025; Zhu et al., 2026; Cai et al., 2025). They also examine robustness to deployment variations, including hidden prompts, decoding settings, retrieval augmentation, and chain-of-thought prompting (Zhu et al., 2026; Wang et al., 2026; Pasquini et al., 2025). LeaFBench evaluates source-model attribution under parameter changes and runtime variations (Shao et al., 2025). Its attribution labels can associate a quantized or fine-tuned derivative with its source model even though the deployed model specification has changed. KBF studies black-box API auditing under wrapper variations, including role prompts, temperature, and retrieval augmentation (Fang et al., 2026). These evaluations establish precedents for controlled comparisons, but their objectives, intervention sets, and access assumptions differ. Comparisons across methods therefore require an explicit account of which changes should trigger rejection and which should preserve the model claim. AuditGym provides this account through its separation of model specification, generation policy, and interaction layer, and evaluates all methods under the same controlled conditions.

Agent harnesses and auditing during execution. Agent harnesses manage context, tools, and environment feedback across model calls, as illustrated by ReAct, AutoGen, Codex, and DeepSeek Harness (Yao et al., 2023; Wu et al., 2024; OpenAI, 2026; DeepSeek AI, 2026). Agent interfaces and tool descriptions can alter observable behavior (Yang et al., 2024b; Faghih et al., 2025). Prior auditing work examines system-prompt wrappers (Wang et al., 2026), response tampering during agent execution (Luo et al., 2026), and model attribution under a shared browser-agent harness (Lugoloobi et al., 2026). AuditGym tests whether references collected without a harness remain valid inside one, and whether auditors detect substitutions when the harness is matched.

3 AuditGym Benchmark

AuditGym evaluates what black-box LLM identity verification can reveal about changes to an LLM deployment. Its central design principle is to compare deployments with known differences, so that an audit's response can be interpreted in relation to a specified deployment change. Figure 1 summarizes this design: reference and target deployments provide evidence to each auditor, whose decisions are interpreted against the known deployment differences and its verification objective.

We first formulate the auditing problem, then define the deployment changes and explain how their effects are assessed against different audit objectives.

3.1 Problem Formulation

We consider black-box LLM identity verification with a trusted reference. The auditor queries a target service and compares its outputs with evidence from the claimed model, without inspecting the target's parameters or internal computation. This access setting is used by MET, RUT, and AgentProv (Gao et al., 2025; Zhu et al., 2026; Wang et al., 2026). It captures a practical situation in which a third-party service is opaque, but the claimed model is available through released weights or a trusted endpoint. The reference provides an independent basis for assessing the provider's claim. MET and AgentProv require reference responses, whereas RUT additionally requires token-level scoring by the reference model. We distinguish three components of a deployment:

- The **model specification** M defines the learned parameters and their numerical representation used to compute model predictions.
- The **generation policy** G determines how output tokens are selected from those predictions.
- The **interaction layer** I constructs model inputs and coordinates model calls with tools and environment feedback.

For M, we consider checkpoint substitution, learned adapters, and quantization. For G, we consider decoding strategies and generation-time watermarking. For I, we consider system prompts and agent harness settings. These interventions can be extended as the benchmark expands.

For input x, the observable output distribution of each deployment is

$$P_{\mathrm{ref}}(\cdot \mid x) = P(\cdot \mid x; M_{\mathrm{ref}}, G_{\mathrm{ref}}, I_{\mathrm{ref}}), \quad P_{\mathrm{tgt}}(\cdot \mid x) = P(\cdot \mid x; M_{\mathrm{tgt}}, G_{\mathrm{tgt}}, I_{\mathrm{tgt}}). \tag{1}$$

The verification problem is to determine whether $M_{\mathrm{tgt}} = M_{\mathrm{ref}}$, allowing the evaluated differences in generation policy and interaction layer. The desired decision is to accept the model claim when $M_{\mathrm{tgt}} = M_{\mathrm{ref}}$ and reject it when $M_{\mathrm{tgt}} \neq M_{\mathrm{ref}}$. The audit protocol fixes the queries or tasks, observation channel, and evidence budget. Observations may be direct responses, agent final outputs, or exposed transcripts. Agent comparisons also control task initialization and environment conditions.

A generation or interaction change may alter response distributions without violating the model claim, so a valid distributional rejection can be undesirable for model verification. The benchmark evaluator records the configured deployment differences to assess audit decisions, but withholds target specifications and change records from the auditor.

3.2 Controlled Deployment Changes

AuditGym constructs reference–target pairs that differ in a specified deployment setting. The main comparisons vary one factor at a time, matching the remaining conditions within each pair. Using the components defined in Section 3.1, the three comparison types are

$$\begin{aligned} \text{Model} \quad & M_{\mathrm{tgt}} \neq M_{\mathrm{ref}}, \quad G_{\mathrm{tgt}} = G_{\mathrm{ref}}, \quad I_{\mathrm{tgt}} = I_{\mathrm{ref}}, \\ \text{Generation} \quad & M_{\mathrm{tgt}} = M_{\mathrm{ref}}, \quad G_{\mathrm{tgt}} \neq G_{\mathrm{ref}}, \quad I_{\mathrm{tgt}} = I_{\mathrm{ref}}, \\ \text{Interaction} \quad & M_{\mathrm{tgt}} = M_{\mathrm{ref}}, \quad G_{\mathrm{tgt}} = G_{\mathrm{ref}}, \quad I_{\mathrm{tgt}} \neq I_{\mathrm{ref}}. \end{aligned} \tag{2}$$

Model changes. We consider three types of model changes in our evaluation: *(1) Checkpoint substitution* tests whether an auditor distinguishes different source models. *(2) Trained LoRA adapters* modify a shared base model through low-rank parameter updates (Hu et al., 2022). *(3) Quantization* changes the numerical representation of the same source checkpoint. We consider 8-bit quantization (Dettmers et al., 2022), GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024), and four-bit formats including NF4, introduced by QLoRA (Dettmers et al., 2023). All three types of changes alter the model specification under our verification objective, so rejection is the desired outcome. Considering them separately reveals whether detection extends beyond replacing an entire checkpoint.

Generation changes. We consider two types of generation changes in our evaluation: *(1) Sampling policies* govern how output tokens are selected from model predictions (Finlayson et al.,

2024). *(2) Generation-time watermarking* embeds statistical signals during decoding. We evaluate KGW (Kirchenbauer et al., 2023), SynthID-Text (Dathathri et al., 2024), TextSeal (Sander et al., 2026), and single-key Gumbel-max using the TextSeal implementation (Sander et al., 2026). Both types of changes modify token selection without changing the model specification, so acceptance is the desired outcome.

Interaction changes. We consider two types of interaction changes in our evaluation: *(1) System instructions* alter the prompt context presented to the model while weights remain fixed (Sclar et al., 2024). *(2) Agent harness settings* modify the orchestration that coordinates model calls with tools and environment feedback. Both types of changes leave the model specification unchanged, so acceptance is the desired outcome. A complementary comparison changes the model within a fixed harness to test whether model differences remain detectable through agent execution. The observation channel and task initialization are matched within each comparison.

3.3 Auditing Procedures and Assessment

We evaluate each auditing method as a verifier for a single claimed model, with each method collecting its reference evidence from the claimed model. All methods retain their required observations and reference access, including trusted local scoring where needed. Some methods, such as LLMmap, were originally designed to identify which model from a known set is being served. We adapt these to verify a single claimed model, and report the performance of this verification adaptation.

An auditor's native decision rule may test a different property from our verification objective. For example, distributional tests such as MET and RUT assess whether reference and target observations follow the same distribution under the audit protocol (Gao et al., 2025; Zhu et al., 2026). Writing these distributions as P_{ref} and P_{tgt}, their null hypothesis is

$$H_0 : \quad P_{\mathrm{ref}} = P_{\mathrm{tgt}}. \tag{3}$$

A change in G or I can violate this hypothesis even when M is unchanged. A method that correctly rejects H_0 under a generation change is therefore not making a statistical error, but it is producing an undesirable outcome for identity verification. We preserve each method's decision rule and assess how well its outcomes serve model verification. We report rejection rates separately for each change family: high rejection is desirable when M changes (detection), and low rejection is desirable when only G or I changes (robustness). Section 4 specifies the experimental settings and aggregation rules.

4 Evaluating Black-box Audits

We evaluate detection of model changes (M), robustness to generation changes (G), and verification under interaction changes (I). The corresponding analyses follow this order, with sensitivity to individual settings discussed alongside each type of change.

4.1 Experimental Setup

Auditing methods. We evaluate four LLM identity auditing methods at a nominal significance level of $\alpha = 0.05$: MET (Gao et al., 2025), RUT (Zhu et al., 2026), AgentProv (Wang et al., 2026), and a verification adaptation of LLMmap (Pasquini et al., 2025). Appendix C describes the method details and experimental settings.

Models. Our experiments cover 19 models from the Llama (Grattafiori et al., 2024), Qwen (Yang et al., 2024a; Qwen Team, 2025), Mistral (Jiang et al., 2023), Phi (Abdin et al., 2024a;b), Granite (Granite Team, IBM, 2024), and Gemma (Gemma Team, 2024) families, with TinyLlama (Zhang et al., 2024) included in quantization experiments. Table 8 lists the models used in each experiment.

Evaluation protocol. Each comparison varies one deployment setting while keeping all others unchanged. Implementation exceptions are documented alongside the relevant experiment. We report the number of comparisons in which each method detects a difference, together with the total number of comparisons. When the model itself has changed, detections are desirable and higher counts indicate better sensitivity. When only generation or interaction settings change, the model is unchanged, so detections are false indications and lower counts are desirable. We also compare unchanged deployments to measure false detections when no setting changes.

Table 1: **Detection of model changes.** Entries report detected differences / evaluated comparisons across three types of model changes and an unchanged-deployment control. Checkpoint substitution compares each of 14 reference models with the other 13 models. Trained LoRA evaluates five base models with fine-tuned adapters applied. Quantization results aggregate the five settings in Table 2. Detections on unchanged deployments are false positives. Higher counts are desirable for model changes (↑), and lower counts for unchanged deployments (↓). Appendix Tables 10–16 give per-comparison results, except for checkpoint substitution.

Comparison	MET	RUT	AgentProv	LLMmap
Checkpoint substitution ↑	181/182	182/182	182/182	182/182
Unchanged deployment ↓	0/14	0/14	0/14	3/14
Trained LoRA ↑	6/10	7/10	7/10	10/10
Quantization ↑	30/62	51/62	43/62	62/62

Table 2: **Detection by quantization setting.** Entries give detections / comparisons, disaggregating the quantization row of Table 1 across five settings from three quantization frameworks: BitsAndBytes (INT8, FP4, NF4), AWQ (INT4), and GPTQ (INT4). Model coverage varies by setting because not all models have pre-quantized checkpoints available for every framework. Higher counts indicate more detected model changes. Appendix Tables 12–16 provides more detailed per-model results.

	BitsAndBytes			AWQ	GPTQ
Method	INT8 ↑	FP4 ↑	NF4 ↑	INT4 ↑	INT4 ↑
MET	2/12	11/14	8/14	5/11	4/11
RUT	4/12	14/14	13/14	10/11	10/11
AgentProv	1/12	12/14	9/14	10/11	11/11
LLMmap	12/12	14/14	14/14	11/11	11/11

4.2 Detecting Model Changes

We first evaluate whether the four methods detect changes to the model specification. Checkpoint substitution tests whether they distinguish different models, while trained LoRA adapters and quantization test whether they detect modifications to the same source model. Table 1 summarizes detections for these three types of changes and false detections when the deployment remains unchanged.

Checkpoint substitution. RUT, AgentProv, and LLMmap detect all 182 checkpoint substitutions, while MET misses one (Table 1). In the unchanged-deployment comparisons, MET, RUT, and AgentProv produce no false detections, while LLMmap produces three across the fourteen models. Near-universal detection is expected because different checkpoints differ substantially.

Trained LoRA. LoRA adapters modify a small fraction of parameters through low-rank updates, producing subtler distributional shifts than full checkpoint substitution. MET detects six of ten adapters, while RUT and AgentProv each detect seven (Table 1). LLMmap detects all ten. The lower rates relative to checkpoint substitution indicate that localized parameter modifications are harder to detect through black-box observations.

Quantization. Quantization reduces numerical precision without retraining, and its effect on output distributions varies with both the target precision and the quantization method. The quantization breakdown in Table 2 shows that MET, RUT, and AgentProv detect only one to four of the twelve INT8 changes. All three detect four-bit quantization more frequently than INT8 quantization, and this pattern also holds when comparing BitsAndBytes settings on the same models (Appendix B.3). LLMmap detects every quantization change across all five settings.

These results show that strong discrimination between checkpoints does not guarantee detection of changes to the same source model. Detection difficulty increases as model changes become more localized, from full substitution through adapter fine-tuning to quantization at higher bit widths. LLMmap's complete detection comes with false detections on unchanged deployments, so it does not

Table 3: **False detections under decoding changes.** Entries report false detections / evaluated comparisons on ten models whose model specification is unchanged. Each changed setting differs from a reference generated with temperature 1, top-$p = 1$, and unrestricted top-k. The unchanged control independently resamples the same decoding policy as the reference. All detections are false indications of a model change, so lower counts are desirable (↓). RUT does not apply to the top-k settings (denoted as N/A and explained in Appendix C.2), so its total covers the six other settings. Appendix Table 17 gives per-model results.

Setting	MET ↓	RUT ↓	AgentProv ↓	LLMmap ↓
Temperature 0 (greedy)	10/10	10/10	7/10	0/10
Temperature 0.3	10/10	10/10	5/10	0/10
Temperature 0.7	3/10	10/10	2/10	0/10
Temperature 1.3	7/10	10/10	5/10	4/10
Top-$p = 0.8$	3/10	10/10	0/10	0/10
Top-$p = 0.9$	1/10	10/10	0/10	0/10
Top-$k = 20$	1/10	N/A	0/10	0/10
Top-$k = 50$	1/10	N/A	0/10	1/10
Total (changed settings)	36/80	60/60	19/80	5/80
Unchanged control	0/10	1/10	0/10	1/10

establish reliable model identity verification. Effective verification requires both sensitivity to model changes and reliable recognition of unchanged models.

4.3 Robustness to Generation Changes

We next evaluate whether changes to generation policy cause false detections of a model change. We vary decoding settings and generation-time watermarking while keeping the model specification fixed. Detections therefore count as false indications of a model change.

Decoding. We vary temperature, top-p, and top-k on ten models while retaining an independently sampled unchanged-policy control for each model, with experimental settings detailed in Appendix B.4. Temperature changes account for most false detections by MET, AgentProv, and LLMmap (Table 3). MET flags every greedy and temperature-0.3 target, while RUT flags every temperature and top-p change. RUT does not apply to the top-k settings. AgentProv produces false detections under temperature changes but none under the evaluated top-p and top-k changes. LLMmap produces the fewest false detections, with four of its five occurring at temperature 1.3. In the unchanged-policy controls, RUT and LLMmap each produce one false detection, while MET and AgentProv produce none.

Watermarking. We evaluate four watermarking methods, KGW (Kirchenbauer et al., 2023), SynthID-Text (Dathathri et al., 2024), TextSeal (Sander et al., 2026), and single-key Gumbel-max using the implementation from TextSeal. For each method, we compare watermarked and unmarked deployments of the same thirteen models. MET flags every model under SynthID-Text, TextSeal, and Gumbel-max, compared with three of thirteen under KGW (Table 4). RUT shows the opposite pattern, flagging all thirteen KGW targets but only four of the thirty-nine targets across the other three watermark methods. AgentProv flags seven to nine models under each of these three methods, compared with five under KGW. LLMmap produces the fewest false detections across the four watermark settings, seven of fifty-two, with none in its unmarked controls.

We also vary the KGW logit bias $\delta \in \{0.5, 1, 2, 4\}$, which controls how strongly the watermark favors selected tokens. False detections increase with watermark strength for all four auditors (Table 5). MET and AgentProv rarely flag the weakest setting but flag most models at the strongest. RUT flags every model at $\delta \geq 2$. Under the sampling protocol, LLMmap produces no false detections at $\delta \leq 1$ and four at $\delta = 4$, while its unmarked controls yield none. Appendix B.5 provides the collection settings and unmarked-control details.

These results show that all four auditors can mistake a change in generation policy for a change in the model. Their contrasting responses across decoding settings and watermark methods show that robustness to one generation change does not imply robustness to another. The KGW results further

Table 4: **False detections across watermark methods.** Entries give false detections / comparisons on thirteen unchanged models, each comparing a watermarked deployment against an unmarked reference of the same model. KGW uses $\delta = 2$. Totals exclude unmarked controls, which compare two unmarked deployments. Lower is better. Appendix Table 18 gives per-model results.

Watermark	MET	RUT	AgentProv	LLMmap
KGW ($\delta = 2$)	3/13	13/13	5/13	1/13
SynthID-Text	13/13	1/13	7/13	1/13
TextSeal	13/13	2/13	7/13	2/13
Gumbel-max	13/13	1/13	9/13	3/13
Total	42/52	17/52	28/52	7/52
Unmarked control	1/13	2/13	0/13	0/13

Table 5: **False detections by KGW strength.** Entries give false detections / comparisons on thirteen unchanged models, each comparing a KGW-watermarked deployment against an unmarked reference of the same model. The logit bias δ controls how strongly the watermark favors selected tokens, with higher values producing a stronger watermark signal. Total sums all four auditors at each strength. Lower is better. Appendix Table 18 gives per-model results.

Method	$\delta = 0.5$	$\delta = 1$	$\delta = 2$	$\delta = 4$
MET	1/13	3/13	3/13	11/13
RUT	3/13	11/13	13/13	13/13
AgentProv	1/13	3/13	5/13	10/13
LLMmap	0/13	0/13	1/13	4/13
Total	5/52	17/52	22/52	38/52

show that robustness depends on watermark strength, with stronger watermarking increasing false detections even for LLMmap, which is less sensitive in these experiments. Reliable LLM identity verification therefore requires distinguishing changes to the model specification from behavioral differences introduced by its generation policy.

4.4 Robustness to Interaction Changes

We finally evaluate how changes to system prompts and agent harnesses affect LLM identity verification. We first test whether these interaction changes cause false detections while the model specification remains unchanged. For agent harnesses, we also test whether auditors distinguish different models when the reference and target use the same harness configuration.

System prompts. AgentProv previously showed that hidden system prompts can cause MET and RUT to flag unchanged models (Wang et al., 2026). We examine this vulnerability across all four methods using fifteen instructions from SystemChat (Hartford, 2024) covering roles, styles, and output constraints. Each instruction is applied to Llama-3.2-1B-Instruct, Qwen3-4B, and Phi-4-mini-instruct while retaining the model and generation settings, with selection and collection details in Appendix B.6. All four methods produce false detections in most of the resulting 45 comparisons (Table 6). LLMmap flags every case, while MET, RUT, and AgentProv flag 38, 39, and 37 cases, respectively. MET, RUT, and AgentProv produce more false detections in the longer-prompt groups, although differences in instruction content prevent attributing this pattern to length alone. AgentProv produces no false detections for the generic instruction "You are a helpful AI assistant." on any of the three models.

Agent harnesses. We extend the system-prompt wrapper evaluation of AgentProv (Wang et al., 2026) to native DeepSeek Minimal and Codex Default harnesses, which manage context, tools, and successive model calls (DeepSeek AI, 2026; OpenAI, 2026). For each harness, we use references collected without a harness for five same-model comparisons, then test all twenty directed model substitutions with the harness matched on both sides. MET, RUT, and LLMmap observe final responses, while AgentProv observes the first delivered model-issued tool choice using added probe tools. MET and RUT analyze completed responses only. Appendix B.7 describes the models,

Table 6: **False detections under system-prompt changes.** Entries report false detections / evaluated comparisons across fifteen instructions and three unchanged models. Prompt-length and model columns give two breakdowns of the same 45 comparisons. Each length group contains five distinct instructions. Llama, Qwen, and Phi denote Llama-3.2-1B-Instruct, Qwen3-4B, and Phi-4-mini-instruct. Lower counts are desirable (↓). Appendix Table 19 gives per-instruction results.

		Prompt length			Model		
Method	All ↓	Short ↓	Medium ↓	Long ↓	Llama ↓	Qwen ↓	Phi ↓
MET	38/45	11/15	13/15	14/15	10/15	14/15	14/15
RUT	39/45	10/15	14/15	15/15	14/15	14/15	11/15
AgentProv	37/45	10/15	13/15	14/15	14/15	14/15	9/15
LLMmap	45/45	15/15	15/15	15/15	15/15	15/15	15/15

Table 7: **Model verification under agent harnesses.** Entries report detected differences / completed audits. The upper block compares a reference without a harness with the same model under a harness, so detections are false positives (↓). The lower block compares different models under the same harness configuration (↑). MET and RUT use completed responses only. AgentProv uses added probe tools. Appendix Tables 20 and 21 give per-comparison results.

Harness	MET	RUT	AgentProv	LLMmap
Unchanged model, bare reference ↓				
DeepSeek	5/5	5/5	5/5	5/5
Codex Default	4/5	5/5	5/5	5/5
Model substitution, matched harness ↑				
DeepSeek	20/20	19/20	20/20	20/20
Codex Default	20/20	20/20	20/20	20/20

reference setup, completion handling, and AgentProv's probe-tool augmentation. All four methods falsely flag all five unchanged models under DeepSeek (Table 7). Under Codex Default, MET flags four of five, and the other methods flag all five. MET produces no false detection for Llama-3.2-1B-Instruct, the sole unchanged model it does not flag under this harness. When the harness is matched between reference and target, all methods detect every model substitution under Codex Default. Under DeepSeek, RUT detects nineteen of twenty substitutions, while the other methods detect all twenty. The missed substitution uses Qwen2.5-3B-Instruct as the reference and Qwen2.5-7B-Instruct as the target. RUT detects the reverse substitution under DeepSeek and the same directed substitution under Codex Default, showing that this failure is specific to this one direction and harness.

These results show that changes to the interaction layer can be mistaken for changes in model identity. Robustness to a generic system instruction does not extend to the broader instruction set, and strong substitution detection within a matched harness does not establish robustness when only the target uses that harness. Reliable verification therefore requires accounting for differences in the interaction layer between the reference and target.

5 Conclusions

We introduced AuditGym, an extensible benchmark for black-box model verification that separates model specification, generation policy, and interaction layer to make explicit which deployment changes should trigger rejection and which should not. Evaluating four auditing methods across 19 models and seven types of deployment changes, we find that strong checkpoint-substitution detection does not ensure reliable verification: every method falsely rejects unchanged models under some generation, system-prompt, or agent-harness changes, and sensitivity to subtler model modifications such as adapters and quantization varies substantially. These findings demonstrate a pressing need for auditing methods that can reliably distinguish changes to the underlying model from changes in how it is deployed, and AuditGym provides the testbed for developing and evaluating such methods.

AI-USE STATEMENT

We used generative AI tools, including Codex and Claude, to assist with manuscript drafting and revision and literature search. We also used AI tools to assist in developing the experimental pipeline and debugging the code. Separately, the experimental pipeline uses LLM judges to classify generated responses for specified AgentProv studies. The judge models, classification procedures, and validation limits are described in Appendix C and Appendix B.7. The authors take responsibility for the final text, claims, references, code, and experimental results, including material developed with AI assistance.

ETHICS STATEMENT

AuditGym is intended to support transparent assessment of LLM deployment claims. A behavioral discrepancy does not by itself establish model substitution, provider deception, or malicious intent: generation policies and interaction layers can change the observations of an unchanged model. Audit decisions should therefore be interpreted within the stated verification objective and access assumptions, rather than used alone to make allegations about a service. The benchmark also has dual-use implications: knowledge of an auditor's blind spots could help a provider evade detection. We study these limitations through controlled deployment comparisons and report observation boundaries and failures to support careful interpretation. LLM-based action classification can introduce systematic errors. We disclose its role and avoid treating classification completeness as evidence of label accuracy. Reuse or release of models, prompts, response traces, and agent artifacts should respect the applicable licenses, access conditions, and privacy constraints, including the possibility that execution traces contain sensitive information.

REPRODUCIBILITY STATEMENT

Section 3 defines the deployment components, comparison design, auditor access, and interpretation of decisions. Appendix A lists models by experiment and specifies audit access and reporting. Appendix B describes the intervention settings and implementation exceptions. Appendix C documents method-specific probes, evidence budgets, reference preparation, and decision rules, including the LLMmap verification adaptation. Appendix B.7 details harness execution, observation boundaries, and probe-tool augmentation. The appendix specifies response selection for the harness analyses, reference-bank reuse, and method-specific deployment differences.

REFERENCES

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024a. URL https://arxiv.org/abs/2404.14219.

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report, 2024b. URL https://arxiv.org/abs/2412.08905.

Xiaofan Bai, Pingyi Hu, Xiaojing Ma, Linchen Yu, Dongmei Zhang, Qi Zhang, and Bin Benjamin Zhu. ESF: Efficient sensitive fingerprinting for black-box tamper detection of large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10477–10494, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.546. URL https://aclanthology.org/2025.findings-acl.546/.

Will Cai, Tianneng Shi, Xuandong Zhao, and Dawn Song. Are you getting what you pay for? auditing model substitution in llm apis, 2025. URL https://arxiv.org/abs/2504.04715.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634:818–823,

2024. doi: 10.1038/s41586-024-08025-4. URL https://www.nature.com/articles/s41586-024-08025-4#citeas.

DeepSeek AI. DeepSeek Harness architecture. Online documentation, 2026. URL https://github.com/deepseek-ai/deepseek-harness/blob/master/docs/architecture.md. Accessed September 9, 2026.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30318–30332. Curran Associates, Inc., 2022. doi: 10.52202/068431-2198. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c3ba4962c05c49636d4c6206a97e9c8a-Paper-Conference.pdf.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115. Curran Associates, Inc., 2023. doi: 10.52202/075280-0441. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1feb87871436031bdc0f2beaa62a049b-Paper-Conference.pdf.

Kazem Faghih, Wenxiao Wang, Yize Cheng, Siddhant Bharti, Gaurang Sriramanan, Sriram Balasubramanian, Parsa Hosseini, and Soheil Feizi. Tool preferences in agentic LLMs are unreliable. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20954–20969, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1060. URL https://aclanthology.org/2025.emnlp-main.1060/.

Yijia Fang, Yiqing Feng, Bingyu Li, and Mingxun Zhou. Kbf: Knowledge boundary as fingerprint for language model and black-box api auditing, 2026. URL https://arxiv.org/abs/2605.29524.

Matthew Finlayson, John Hewitt, Alexander Koller, Swabha Swayamdipta, and Ashish Sabharwal. Closing the curious case of neural text degeneration. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dONpC9GL1o.

Matthew Finlayson, Xiang Ren, and Swabha Swayamdipta. Every language model has a forgery-resistant signature. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vLFqOoMBol.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Irena Gao, Percy Liang, and Carlos Guestrin. Model equality testing: Which model is this api serving? In *International Conference on Learning Representations*, volume 2025, pp. 86369–86382, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/d73234a13815fc1f9779dd17d89be9b4-Paper-Conference.pdf.

Gemma Team. Gemma 2: Improving open language models at a practical size. 2024. URL https://arxiv.org/pdf/2408.00118.

Granite Team, IBM. Granite 3.0 language models. Technical report, 2024. URL https://github.com/ibm-granite/granite-3.0-language-models.

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, 2024. URL https://arxiv.org/abs/2407.21783.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

Richardeau Gurvan, Gohar Dashyan, Erwan Le Merrer, and Gilles Tredan. FLIPS: Instance-fingerprinting for LLMs via pseudo-random sequences. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=5DR4NAKO8a.

Eric Hartford. SystemChat. AbacusAI dataset on Hugging Face, 2024. URL https://huggingface.co/datasets/abacusai/SystemChat. Accessed September 26, 2026.

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. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Pingyi Hu, Xiaofan Bai, Xiaojing Ma, Chaoxiang He, Dongmei Zhang, and Bin Benjamin Zhu. RESF: Regularized-entropy-sensitive fingerprinting for black-box tamper detection of large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4889–4903, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.247. URL https://aclanthology.org/2025.emnlp-main.247/.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023. URL https://arxiv.org/abs/2310.06825.

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 of *Proceedings of Machine Learning Research*, pp. 17061–17084. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf.

William Lugoloobi, Samuelle Marro, Jabez Magomere, Joss Wright, and Chris Russell. Known by their actions: Fingerprinting LLM browser agents via UI traces, 2026. URL https://arxiv.org/abs/2605.14786.

Mingyu Luo, Zihan Zhang, Zesen Liu, Yuchong Xie, Zhixiang Zhang, Dung Hiu Hilton Yeung, Wai Ip Lai, Ping Chen, Ming Wen, and Dongdong She. When alignment isn't enough: Response-path attacks on LLM agents, 2026. URL https://arxiv.org/abs/2605.02187v1.

OpenAI. Codex CLI. GitHub software repository, 2026. URL https://github.com/openai/codex/tree/dfdb40cd0b72dfba3293db5c7c441232e8ef1a60. Repository snapshot of September 25, 2026; accessed September 26, 2026.

Dario Pasquini, Evgenios M. Kornaropoulos, and Giuseppe Ateniese. LLMmap: Fingerprinting for large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 299–318, Seattle, WA, August 2025. USENIX Association. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/pasquini.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Tom Sander, Hongyan Chang, Tomáš Souček, Tuan Tran, Valeriu Lacatusu, Sylvestre-Alvise Rebuffi, Alexandre Mourachko, Surya Parimi, Christophe Ropers, Rashel Moritz, et al. TextSeal: A localized LLM watermark for provenance & distillation protection, 2026. URL https://arxiv.org/abs/2605.12456.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models'sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations*, volume 2024, pp. 25055–25083, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/6c0e99d736da621403018ca7b32b1a4d-Paper-Conference.pdf.

Shuo Shao, Yiming Li, Yu He, Hongwei Yao, Wenyuan Yang, Dacheng Tao, and Zhan Qin. Sok: Large language model copyright auditing via fingerprinting, 2025. URL https://arxiv.org/abs/2508.19843.

Yao Tong, Haonan Wang, Siquan Li, Kenji Kawaguchi, and Tianyang Hu. Seedprints: Fingerprints can even tell which seed your large language model was trained from. In *International Conference on Learning Representations*, volume 2026, pp. 44530–44558, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/49e48a1a40628b160101c1375b8eb151-Paper-Conference.pdf.

Xun Wang, Bihe Zhao, Michael Backes, Franziska Boenisch, and Adam Dziedzic. Agentprov: Auditing agentic llm api providers via tool-use policy probes, 2026. URL https://arxiv.org/abs/2609.00052.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a. URL https://arxiv.org/abs/2412.15115.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024b. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5a7c947568c1b1328ccc5230172e1e7c-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. TinyLlama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024. URL https://arxiv.org/abs/2401.02385.

Xiaoyuan Zhu, Yaowen Ye, Tianyi Qiu, Hanlin Zhu, Sijun Tan, Ajraf Mannan, Jonathan Michala, Raluca Popa, and Willie Neiswanger. Auditing black-box llm apis with a rank-based uniformity test. In *International Conference on Learning Representations*, volume 2026, pp. 54232–54261, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/590d0ed6070f066773dc4c6863f4de34-Paper-Conference.pdf.

A Detailed Experimental Settings

A.1 Models by Experiment

Table 8 lists models and comparison counts for Section 4. The LoRA row lists the base models.

Table 8: **Models by experiment.** The LoRA row lists the base models. The watermark-family comparison reuses KGW at $\delta = 2$ from the strength experiment.

Experiment	Models
Checkpoint substitution	Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen3-1.7B, Qwen3-4B, Mistral-7B-Instruct-v0.3, Phi-3.5-mini-instruct, granite-3.3-2b-instruct, Qwen3.5-4B, Phi-4-mini-instruct
Unchanged deployment	Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen3-1.7B, Qwen3-4B, Mistral-7B-Instruct-v0.3, Phi-3.5-mini-instruct, granite-3.3-2b-instruct, Qwen3.5-4B, Phi-4-mini-instruct
Trained LoRA	Llama-3.2-1B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Mistral-7B-Instruct-v0.3, Phi-4-mini-instruct
BitsAndBytes INT8	Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen3-1.7B, Qwen3-4B, Mistral-7B-Instruct-v0.3, Phi-3.5-mini-instruct, granite-3.3-2b-instruct
BitsAndBytes FP4/NF4	Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen3-1.7B, Qwen3-4B, Mistral-7B-Instruct-v0.3, Phi-3.5-mini-instruct, granite-3.3-2b-instruct, Qwen3.5-4B, Phi-4-mini-instruct
AWQ INT4	Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen3-4B, Qwen3-8B, Llama-3.1-8B-Instruct, gemma-2-9b-it, Mistral-7B-Instruct-v0.3, TinyLlama-1.1B-Chat-v1.0, Qwen3.5-4B
GPTQ INT4	Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen2.5-Coder-1.5B-Instruct, Qwen2.5-Coder-3B-Instruct, Llama-3.1-8B-Instruct, gemma-2-9b-it, Mistral-7B-Instruct-v0.3, TinyLlama-1.1B-Chat-v1.0, Phi-4-mini-instruct
Decoding	Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen3-4B, Mistral-7B-Instruct-v0.3, Phi-4-mini-instruct
Watermark strength	Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen3-1.7B, Qwen3-4B, Mistral-7B-Instruct-v0.3, Phi-3.5-mini-instruct, granite-3.3-2b-instruct, Phi-4-mini-instruct
Watermark families	Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen3-1.7B, Qwen3-4B, Mistral-7B-Instruct-v0.3, Phi-3.5-mini-instruct, granite-3.3-2b-instruct, Phi-4-mini-instruct
System prompts	Llama-3.2-1B-Instruct, Qwen3-4B, Phi-4-mini-instruct
Native harness	Qwen2.5-3B-Instruct, Qwen3.5-4B, Qwen3-4B, Qwen2.5-7B-Instruct, Llama-3.2-1B-Instruct

A.2 Audit Access and Reporting

Table 9 summarizes the evidence and reference access required by each method. All targets are accessed through black-box interfaces. Deployment-specific enrollment and calibration use only the claimed reference. Methods retain their own probes and evidence budgets, as detailed in Appendix C.

Counts summarize evaluated comparisons, with dependencies from reused references, models, and instructions. Harness response selection is specified in Appendix B.7.

Table 9: **Audit evidence and reference access.** LLMmap denotes the reference-only verification adaptation.

Method	Audit evidence	Trusted reference access
MET	Sampled response text	Response sampling
RUT	Response scores and ranks	Sampling and token-level scoring
AgentProv	Classified tool choices	Response sampling
LLMmap	Learned response fingerprints	Enrollment and calibration responses

B Details for Experimental Settings

This section gives the intervention settings and method-specific deployment differences for Section 4.

B.1 Checkpoint Substitution

We evaluate all 182 directed substitutions among the 14 checkpoints in Table 8, together with one unchanged-deployment comparison per model. Each substituted target is evaluated against the claimed reference without using target responses for reference preparation. Comparisons are directed because enrollment, calibration, and token scoring depend on the reference model.

B.2 Trained LoRA Adapters

The target applies a published, task-trained LoRA adapter to the reference checkpoint at its published scaling. The selected inventory contains one adapter each for Llama-3.2-1B, Qwen2.5-1.5B, and Phi-4-mini, and four each for Qwen2.5-3B and Mistral-7B-v0.3. Tasks cover instruction following, persona adaptation, code, mathematics, translation, medicine, finance, and tool selection. One Mistral medical adapter produces no parameter update and is excluded, leaving ten changed-model comparisons.

Both sides use the unquantized BF16 base with matched input formatting and generation settings. Adapters trained on quantized bases are also evaluated on the unquantized base, so these comparisons isolate the adapter from a deployment-time quantization change.

B.3 Quantization

We evaluate 12 BitsAndBytes INT8, 14 FP4, 14 NF4, 11 AWQ INT4, and 11 GPTQ INT4 comparisons against the corresponding unquantized source checkpoints. FP4 and NF4 are distinct four-bit representations. BitsAndBytes comparisons use Transformers with BF16 references, BF16 computation for FP4/NF4, and double quantization disabled. AWQ/GPTQ comparisons use vLLM with matched reference and target precision within each method. No comparison adds KV-cache quantization or fine-tuning.

For MET, AWQ/GPTQ targets are converted from the source checkpoints and served with FP16 references. The Qwen3-8B AWQ conversion uses BF16 calibration and FP16 serving. The primary Qwen2.5-0.5B AWQ comparison uses the first eligible completed conversion ($p = 0.058$). A separate conversion detects this change, illustrating sensitivity to the conversion recipe, but does not replace the primary result.

RUT's AWQ/GPTQ conversions use 128 disjoint 2,048-token WikiText training blocks and group size 128. Quantizer calibration is separate from audit preparation. Serving and scoring use matched FP16, except Gemma-2, which uses BF16. AWQ calibration uses BF16 for Gemma-2 and Qwen3-8B. LLMmap uses these same converted checkpoints with matched BF16 deployments and a 2,048-token context for TinyLlama. Its Qwen3.5-4B AWQ target returns empty one-token stops on the audit probes, so this detection includes an empty-output deployment.

AgentProv uses BF16 for BitsAndBytes and FP16 for AWQ/GPTQ, except Gemma-2, which uses BF16. Twenty AWQ/GPTQ targets share the local conversions. Gemma-2 and Qwen3.5-4B AWQ instead use publisher checkpoints checked for source compatibility and deployment behavior before

audit-result inspection. These conversion and precision differences mean that quantized targets are not identical across all auditors.

On the 12 shared BitsAndBytes models, MET detects 2/12 INT8, 10/12 FP4, and 7/12 NF4 changes. RUT detects 4/12, 12/12, and 11/12, and AgentProv detects 1/12, 10/12, and 7/12. Thus the higher detection under four-bit settings also holds with model coverage fixed. On the eight-model shared AWQ/GPTQ subset, MET detects 3/8 and 4/8, RUT detects 7/8 and 7/8, and AgentProv detects 7/8 and 8/8. Reference configurations are matched within these groups, but differ between BitsAndBytes and AWQ/GPTQ.

B.4 Decoding Policies

Each of the ten models in Table 8 has a temperature-1 reference, eight changed-policy targets, and one independently sampled unchanged-policy target. Changed temperatures are 0, 0.3, 0.7, and 1.3, top-p values are 0.8 and 0.9, and top-k values are 20 and 50. Each target changes one factor. The reference uses top-$p = 1$, unrestricted top-k, zero frequency and presence penalties, and repetition penalty 1. Reference evidence is shared across a model's nine comparisons. Method-specific budgets are given in Appendix C. RUT does not apply to the top-k targets, as explained in Appendix C.2.

B.5 Watermarking

The KGW strength experiment uses green-list fraction $\gamma = 0.25$ and logit biases $\delta \in \{0.5, 1, 2, 4\}$ (Kirchenbauer et al., 2023). The family comparison uses KGW at $\delta = 2$, SynthID-Text with n-gram length 5 and tournament depth 30, TextSeal with a three-token context and mixing weight 0.9, and single-key Gumbel-max with a three-token context. Each scheme has a fixed key configuration. Watermark strengths are not matched across families.

All experiments use the same thirteen models with local Transformers BF16 generation. The reference is unmarked, and the target differs only in the watermark. Auditors receive response evidence without watermark keys or detector scores. MET, RUT, and AgentProv share an unmarked reference across the four KGW strengths. They use a separate unmarked reference for the three additional families. The KGW $\delta = 2$ comparisons are reused in Table 4. LLMmap shares enrollment and calibration across all seven distinct marked settings. Each collection also includes an independently sampled unmarked target.

The unmarked-control row in Table 4 reports thirteen comparisons per method, with 1/13, 2/13, 0/13, and 0/13 false detections for MET, RUT, AgentProv, and LLMmap, respectively. The marked totals in that table aggregate the four watermark families, whereas Table 5 varies only KGW strength.

MET uses 25 probes with 32 responses each, temperature 1, and a 50-token output cap. RUT uses 100 probes with 100 reference responses and one target response per probe, temperature 0.5, and a 30-token cap. AgentProv uses 20 probes with 20 responses each, temperature 0.7, and an 8,192-token cap for the additional-family collection. Tool order is shuffled per sample. LLMmap uses eight-probe traces, three enrollment traces, 199 calibration traces, and one target trace per setting, with temperature 1, a 1,024-token output cap, and a 4,096-token context. Sampling uses unrestricted top-k and top-$p = 1$. Decision rules follow Appendix C.

B.6 System Prompts

The target prepends one SystemChat instruction to the reference system message while preserving the auditor's mandatory instructions. We select fifteen distinct published instructions before audit-result inspection, with five in each length band. Bands are defined by terciles of 1,650 unique source instructions tokenized with the Qwen3 tokenizer. Short instructions contain at most 15 tokens, medium instructions 16–24, and long instructions at least 25. Each band contains two role instructions, one style instruction, and two output constraints. Dataset conversation turns are not used as audit queries.

Every instruction is evaluated on Llama-3.2-1B-Instruct, Qwen3-4B, and Phi-4-mini-instruct in BF16, with Qwen3 thinking disabled. Each model shares reference evidence across its fifteen comparisons. The model, length, and purpose breakdowns summarize the same 45 comparisons. RUT's 30-token

cap is reached by 93.31% of reference and 90.00% of target responses, so its results predominantly concern capped prefixes.

MET's primary collection uses one rendering implementation throughout. An earlier complete collection with the same rendered prompts but different request-seed streams gives 29/45 detections, compared with 38/45 in the primary collection. No individual outcomes are selected between the two collections.

B.7 Native Agent Harnesses

Comparison design. We evaluate DeepSeek Minimal and Codex Default on the five models in Table 8. Each method compares a bare reference with the same model under each harness, giving ten unchanged-model comparisons. It also evaluates all twenty directed checkpoint substitutions within each harness, giving forty model-change comparisons. Five bare reference banks, ten harness reference banks, and ten harness target banks supply the evidence. Reference and target responses are independently collected, with compatible reference banks reused across comparisons. Ten additional same-model comparisons within matched harnesses serve as diagnostics. The study does not include bare-versus-bare controls.

Execution and observations. Each native observation starts a fresh session and workspace. Task-associated model calls, including auxiliary calls, use the same checkpoint. MET, RUT, and LLMmap observe the original probe and final response, without internal trajectories or deployment metadata. Substitution comparisons use reference responses collected under the same harness as the target. LLMmap likewise uses harness-specific enrollment and calibration. Available tools are not forced to execute.

Each model request has a 1,024-token output cap. Tasks are limited to eight forwarded calls, 32 primitive actions, 8,192 aggregate generated tokens, 300 seconds, and 65,536 final-output bytes. The reported MET and RUT analyses select completed responses under post-collection protocol amendments. Their results therefore concern completed-response distributions.

MET. Each bank contains 25 probes with 32 planned observations per probe, generated at temperature 1 and top-$p = 1$. The test uses 250 observations per side, a normalized Hamming kernel over at most 1,000 characters, and 1,000 global label permutations. It detects a difference when $p < 0.05$. For each scheduled probe, the analysis takes the next completed observation in sample-index order without redrawing the prompt schedule.

RUT. Each bank uses 100 probes with 100 planned reference responses and one target response per probe. Generation uses temperature 0.5, top-$p = 1$, and unrestricted top-k. A probe is retained when its target task completes and at least one reference response is available. All completed reference responses for that probe are used, and an audit requires at least two retained probes. The reported comparisons retain 64–93 probes under DeepSeek and 93–100 under Codex.

The claimed reference checkpoint scores both sides using the original probe and up to 64 response tokens. Mean log-rank scores are converted to randomized within-probe percentiles. Empty outputs receive score -1. The Cramér–von Mises statistic uses 10,000 within-probe label permutations with the observed reference counts and detects a difference when $p < 0.05$. Llama–DeepSeek has 6,375 empty outputs among 7,480 completed reference tasks and 64 among 74 completed target tasks.

LLMmap. Each reference condition supplies three enrollment traces and 199 held-out calibration traces. Each trace has eight independent probes, and each target supplies one trace. The E5 encoder and upstream-v0.2 fingerprint checkpoint use a 650-character response window and a 512-token encoder limit. A difference is detected when the target distance strictly exceeds the 190th ordered calibration distance. Generation is greedy, with top-$p = 1$. Qwen2.5 retains repetition penalty 1.05 on both sides. Llama uses vLLM 0.17.0's pythonic template and parser for both bare and harness conditions. Reasoning returned in final text remains in the fingerprint input.

Tools do not execute in the Codex collection, and Llama produces no valid native tool call under either harness. These cases evaluate responses within the harness context without successful tool execution. Some greedy reference conditions contain only one to four distinct calibration traces. Strict threshold comparison retains ties as non-detections. Reference and target collection layouts differ in some conditions, limiting the exchangeability assumption underlying nominal calibration.

AgentProv. We add the tools specified by each of AgentProv's twenty probes to the native catalogs, preserving sample-specific tool order. Tool implementations return fixed, side-effect-free results. These configurations retain the native agent loop with an augmented tool catalog. Each bank contains twenty responses per probe, generated at temperature 0.7 and top-$p = 1$.

Bare responses receive action labels from the fixed LLM judge. Harness responses are classified by the first delivered model-issued call that satisfies the required argument schema. Later execution failure does not erase that choice. No-call, malformed, unmapped, truncated, and missing-capture outcomes remain distinct. Thus bare and harness conditions differ in action extraction as well as interaction context. Llama's native banks contain 78–96% no-call outcomes. The harness analysis uses the uniform sparse-row MMD statistic with 1,000 global-row permutations and detects a difference when the statistic exceeds the empirical 95th-percentile threshold.

C Auditing Methods

This section specifies the protocols used in Section 4. Each method verifies one claimed reference at nominal level $\alpha = 0.05$. Reference preparation does not use examples from alternative target models. Appendix B gives intervention-specific settings, including the harness adaptations.

C.1 MET

MET compares sampled response distributions using a kernel two-sample test (Gao et al., 2025). We use maximum mean discrepancy with a normalized Hamming kernel on response text. For the adapter, quantization, decoding, system-prompt, and watermark studies, each bank contains 25 probes with 32 responses per probe. The test samples 250 prompt–response observations per side, represents responses with at most 1,000 Unicode characters, and uses 1,000 global label permutations. A difference is detected when the empirical permutation p-value is strictly below 0.05. This estimate uses no plus-one correction and has resolution 0.001.

The adapter, quantization, decoding, system-prompt, and watermark studies use a 50-token output cap. Temperature is 1 except when changed by the decoding intervention. Compatible reference banks are shared across targets. Native-harness generation and completion handling are specified in Appendix B.7.

C.2 RUT

RUT scores target responses using token ranks from the trusted reference model (Zhu et al., 2026). Both reference and target responses are scored with that model and tokenizer, and each target score is ranked against reference scores for the same probe. A Cramér–von Mises statistic measures departure of randomized ranks from uniformity. Within-probe label permutations account for finite reference samples and ties. The target is a black box, while the reference supplies token-level logits. Quantization and adapter comparisons retain the unquantized, unadapted reference scorer.

The decoding and system-prompt studies use 100 probes with 100 reference responses and one target response per probe, with a 30-token generation cap. Decoding uses 10,000 within-probe permutations and requires $p < 0.05$. RUT applies to a decoding setting only if that setting demonstrably changes sampling on RUT's request path. A separate check uses four prompts on which the reference gives varied responses and sets each factor to its extreme value ($T = 0$, top-$p = 10^{-6}$, or top-$k = 1$). Each extreme should make the responses deterministic. Temperature and top-p pass this check on all ten models. Top-$k = 1$ does not make the responses deterministic, so we cannot confirm that top-k is applied. RUT therefore does not apply to the top-k settings, and Table 3 reports all of its top-k comparisons as N/A. Reference responses are shared across a model's settings. Watermark and harness settings are given in Appendices B.5 and B.7.

C.3 AgentProv

AgentProv compares action choices elicited by shared tool-selection probes (Wang et al., 2026). A fixed judge maps responses to each probe's predefined action space. The adapter, quantization, and system-prompt studies use `gpt-5.4-2026-03-05`. Deployment identities are withheld from

the judge. These labels represent tool choices expressed in text and have not been independently validated by humans.

Each audit uses twenty probes with twenty responses per probe on each side. The statistic is the uniform mean of per-probe squared differences in action frequencies. We use 1,000 within-probe label permutations and detect a difference when the statistic strictly exceeds the empirical 95th-percentile threshold. Compatible classified reference banks are shared across targets. Decoding comparisons hold output caps and stop strings fixed within each model. The native-harness observation channel and permutation scheme are specified separately in Appendix B.7.

C.4 LLMmap-Based Verification

LLMmap constructs a learned fingerprint from responses to a sequence of probes (Pasquini et al., 2025). We retain its probes and pretrained fingerprint construction, replacing identification among enrolled models with verification against one claimed reference. Three reference traces define an enrolled centroid, and 199 separate reference traces provide calibration distances. Each trace contains eight probes. One target trace supplies the audit decision.

For n calibration scores, the threshold is the $\lceil (n+1)(1-\alpha) \rceil$-th ordered score. We detect a difference only when the target distance is strictly larger. With $n = 199$ and $\alpha = 0.05$, the threshold is at rank 190. This calibration assumes exchangeability of unchanged-deployment calibration and target traces.

Adapter, quantization, and system-prompt comparisons use greedy generation and a 1,024-token output cap. The decoding study enrolls and calibrates at temperature 1, then applies each target policy to a separate trace. Compatible reference banks are shared across comparisons. Watermark collection also uses temperature 1, while native-harness collection is greedy.

D Per-Comparison Results

This section reports every audit decision underlying the aggregate counts in Section 4, except the 182 checkpoint substitutions. Each cell is one comparison between a reference and a target. ■ marks a correct outcome and ■ an incorrect one. For a model change, detecting a difference is correct; for an unchanged model or a generation or interaction change, not detecting a difference is correct. N/A marks comparisons where the method does not apply, which occurs only for RUT under top-k decoding (Table 17). The Count column gives detections / evaluated comparisons for that row and reproduces the corresponding entry of the main-text table.

D.1 Model Changes

Table 10: **Per-adapter decisions for trained LoRA adapters.** Disaggregates the trained-LoRA row of Table 1. Columns name the base model and the adapter's training task. Each cell compares the base model with the same model plus one published adapter; detections are desirable. The excluded Mistral medical adapter has an all-zero update and is not shown.

Method	Llama-3.2-1B: general NLP	Qwen2.5-1.5B: general NLP	Qwen2.5-3B: persona	Qwen2.5-3B: code	Qwen2.5-3B: LaTeX math	Qwen2.5-3B: math	Mistral-7B: finance	Mistral-7B: tool use	Mistral-7B: general NLP	Phi-4-mini: EN–VI translation	Count
MET	■	■	■	■	■	■	■	■	■	■	6/10
RUT	■	■	■	■	■	■	■	■	■	■	7/10
AgentProv	■	■	■	■	■	■	■	■	■	■	7/10
LLMmap	■	■	■	■	■	■	■	■	■	■	10/10

Table 11: **Per-model decisions for unchanged deployments.** Disaggregates the unchanged-deployment row of Table 1. Each cell compares a model with an independently collected deployment of the same model, so every detection is a false positive.

Method	Llama-3.2-1B	Llama-3.2-3B	Llama-3.1-8B	Qwen2.5-0.5B	Qwen2.5-1.5B	Qwen2.5-3B	Qwen2.5-7B	Qwen3-1.7B	Qwen3-4B	Mistral-7B	Phi-3.5-mini	Granite-3.3-2B	Qwen3.5-4B	Phi-4-mini	Count
MET	■	■	■	■	■	■	■	■	■	■	■	■	■	■	0/14
RUT	■	■	■	■	■	■	■	■	■	■	■	■	■	■	0/14
AgentProv	■	■	■	■	■	■	■	■	■	■	■	■	■	■	0/14
LLMmap	■	■	■	■	■	■	■	■	■	■	■	■	■	■	3/14

Table 12: **Per-model decisions under BitsAndBytes INT8 quantization.** Disaggregates the BitsAndBytes INT8 column of Table 2. Each cell compares the unquantized source checkpoint with its quantized target; detections are desirable. Appendix B.3 gives the method-specific conversion and precision differences for all quantization settings.

Method	Llama-3.2-1B	Llama-3.2-3B	Llama-3.1-8B	Qwen2.5-0.5B	Qwen2.5-1.5B	Qwen2.5-3B	Qwen2.5-7B	Qwen3-1.7B	Qwen3-4B	Mistral-7B	Phi-3.5-mini	Granite-3.3-2B	Count
MET	■	■	■	■	■	■	■	■	■	■	■	■	2/12
RUT	■	■	■	■	■	■	■	■	■	■	■	■	4/12
AgentProv	■	■	■	■	■	■	■	■	■	■	■	■	1/12
LLMmap	■	■	■	■	■	■	■	■	■	■	■	■	12/12

Table 13: **Per-model decisions under BitsAndBytes FP4 quantization.** As Table 12, for the BitsAndBytes FP4 column of Table 2.

Method	Llama-3.2-1B	Llama-3.2-3B	Llama-3.1-8B	Qwen2.5-0.5B	Qwen2.5-1.5B	Qwen2.5-3B	Qwen2.5-7B	Qwen3-1.7B	Qwen3-4B	Mistral-7B	Phi-3.5-mini	Granite-3.3-2B	Qwen3.5-4B	Phi-4-mini	Count
MET	■	■	■	■	■	■	■	■	■	■	■	■	■	■	11/14
RUT	■	■	■	■	■	■	■	■	■	■	■	■	■	■	14/14
AgentProv	■	■	■	■	■	■	■	■	■	■	■	■	■	■	12/14
LLMmap	■	■	■	■	■	■	■	■	■	■	■	■	■	■	14/14

Table 14: **Per-model decisions under BitsAndBytes NF4 quantization.** As Table 12, for the BitsAndBytes NF4 column of Table 2.

Method	Llama-3.2-1B	Llama-3.2-3B	Llama-3.1-8B	Qwen2.5-0.5B	Qwen2.5-1.5B	Qwen2.5-3B	Qwen2.5-7B	Qwen3-1.7B	Qwen3-4B	Mistral-7B	Phi-3.5-mini	Granite-3.3-2B	Qwen3.5-4B	Phi-4-mini	Count
MET	■	■	■	■	■	■	■	■	■	■	■	■	■	■	8/14
RUT	■	■	■	■	■	■	■	■	■	■	■	■	■	■	13/14
AgentProv	■	■	■	■	■	■	■	■	■	■	■	■	■	■	9/14
LLMmap	■	■	■	■	■	■	■	■	■	■	■	■	■	■	14/14

Table 15: **Per-model decisions under AWQ INT4 quantization.** As Table 12, for the AWQ INT4 column of Table 2.

Method	Llama-3.1-8B	Qwen2.5-0.5B	Qwen2.5-1.5B	Qwen2.5-3B	Qwen2.5-7B	Qwen3-4B	Mistral-7B	Qwen3.5-4B	Qwen3-8B	Gemma-2-9B	TinyLlama-1.1B	Count
MET	■	■	■	■	■	■	■	■	■	■	■	5/11
RUT	■	■	■	■	■	■	■	■	■	■	■	10/11
AgentProv	■	■	■	■	■	■	■	■	■	■	■	10/11
LLMmap	■	■	■	■	■	■	■	■	■	■	■	11/11

Table 16: **Per-model decisions under GPTQ INT4 quantization.** As Table 12, for the GPTQ INT4 column of Table 2.

Method	Llama-3.1-8B	Qwen2.5-0.5B	Qwen2.5-1.5B	Qwen2.5-3B	Qwen2.5-7B	Mistral-7B	Phi-4-mini	Qwen2.5-Coder-1.5B	Qwen2.5-Coder-3B	Gemma-2-9B	TinyLlama-1.1B	Count
MET	■	■	■	■	■	■	■	■	■	■	■	4/11
RUT	■	■	■	■	■	■	■	■	■	■	■	10/11
AgentProv	■	■	■	■	■	■	■	■	■	■	■	11/11
LLMmap	■	■	■	■	■	■	■	■	■	■	■	11/11

D.2 Generation Changes

Table 17: **Per-model decisions under decoding changes.** Disaggregates Table 3. Each cell compares a model's temperature-1 reference with one changed policy or, in the control row, an independently resampled unchanged policy. Every detection is a false detection. RUT does not apply to the top-k settings (N/A; Appendix C.2).

Method	Setting	Llama-3.2-1B	Llama-3.2-3B	Llama-3.1-8B	Qwen2.5-0.5B	Qwen2.5-1.5B	Qwen2.5-3B	Qwen2.5-7B	Qwen3-4B	Mistral-7B	Phi-4-mini	Count
MET	T=0	■	■	■	■	■	■	■	■	■	■	10/10
	T=0.3	■	■	■	■	■	■	■	■	■	■	10/10
	T=0.7	■	■	■	■	■	■	■	■	■	■	3/10
	T=1.3	■	■	■	■	■	■	■	■	■	■	7/10
	top-p=0.8	■	■	■	■	■	■	■	■	■	■	3/10
	top-p=0.9	■	■	■	■	■	■	■	■	■	■	1/10
	top-k=20	■	■	■	■	■	■	■	■	■	■	1/10
	top-k=50	■	■	■	■	■	■	■	■	■	■	1/10
	Control	■	■	■	■	■	■	■	■	■	■	0/10
RUT	T=0	■	■	■	■	■	■	■	■	■	■	10/10
	T=0.3	■	■	■	■	■	■	■	■	■	■	10/10
	T=0.7	■	■	■	■	■	■	■	■	■	■	10/10
	T=1.3	■	■	■	■	■	■	■	■	■	■	10/10
	top-p=0.8	■	■	■	■	■	■	■	■	■	■	10/10
	top-p=0.9	■	■	■	■	■	■	■	■	■	■	10/10
	top-k=20						N/A					N/A
	top-k=50						N/A					N/A
	Control	■	■	■	■	■	■	■	■	■	■	1/10
AgentProv	T=0	■	■	■	■	■	■	■	■	■	■	7/10
	T=0.3	■	■	■	■	■	■	■	■	■	■	5/10
	T=0.7	■	■	■	■	■	■	■	■	■	■	2/10
	T=1.3	■	■	■	■	■	■	■	■	■	■	5/10
	top-p=0.8	■	■	■	■	■	■	■	■	■	■	0/10
	top-p=0.9	■	■	■	■	■	■	■	■	■	■	0/10
	top-k=20	■	■	■	■	■	■	■	■	■	■	0/10
	top-k=50	■	■	■	■	■	■	■	■	■	■	0/10
	Control	■	■	■	■	■	■	■	■	■	■	0/10
LLMmap	T=0	■	■	■	■	■	■	■	■	■	■	0/10
	T=0.3	■	■	■	■	■	■	■	■	■	■	0/10
	T=0.7	■	■	■	■	■	■	■	■	■	■	0/10
	T=1.3	■	■	■	■	■	■	■	■	■	■	4/10
	top-p=0.8	■	■	■	■	■	■	■	■	■	■	0/10
	top-p=0.9	■	■	■	■	■	■	■	■	■	■	0/10
	top-k=20	■	■	■	■	■	■	■	■	■	■	0/10
	top-k=50	■	■	■	■	■	■	■	■	■	■	1/10
	Control	■	■	■	■	■	■	■	■	■	■	1/10

Table 18: **Per-model decisions under watermarking.** Disaggregates Tables 4 and 5. The KGW δ=2 row is shared by both tables. Each cell compares an unmarked reference with a watermarked deployment of the same model, or with a second unmarked deployment in the control row, so every detection is a false detection. The control row uses the unmarked target collected with the KGW strength sweep. LLMmap results use the sampling protocol.

Method	Setting	Llama-3.2-1B	Llama-3.2-3B	Llama-3.1-8B	Qwen2.5-0.5B	Qwen2.5-1.5B	Qwen2.5-3B	Qwen2.5-7B	Qwen3-1.7B	Qwen3-4B	Mistral-7B	Phi-3.5-mini	Granite-3.3-2B	Phi-4-mini	Count
MET	KGW δ=0.5	■	■	■	■	■	■	■	■	■	■	■	■	■	1/13
	KGW δ=1	■	■	■	■	■	■	■	■	■	■	■	■	■	3/13
	KGW δ=2	■	■	■	■	■	■	■	■	■	■	■	■	■	3/13
	KGW δ=4	■	■	■	■	■	■	■	■	■	■	■	■	■	11/13
	SynthID-Text	■	■	■	■	■	■	■	■	■	■	■	■	■	13/13
	TextSeal	■	■	■	■	■	■	■	■	■	■	■	■	■	13/13
	Gumbel-max	■	■	■	■	■	■	■	■	■	■	■	■	■	13/13
	Unmarked control	■	■	■	■	■	■	■	■	■	■	■	■	■	1/13
RUT	KGW δ=0.5	■	■	■	■	■	■	■	■	■	■	■	■	■	3/13
	KGW δ=1	■	■	■	■	■	■	■	■	■	■	■	■	■	11/13
	KGW δ=2	■	■	■	■	■	■	■	■	■	■	■	■	■	13/13
	KGW δ=4	■	■	■	■	■	■	■	■	■	■	■	■	■	13/13
	SynthID-Text	■	■	■	■	■	■	■	■	■	■	■	■	■	1/13
	TextSeal	■	■	■	■	■	■	■	■	■	■	■	■	■	2/13
	Gumbel-max	■	■	■	■	■	■	■	■	■	■	■	■	■	1/13
	Unmarked control	■	■	■	■	■	■	■	■	■	■	■	■	■	2/13
AgentProv	KGW δ=0.5	■	■	■	■	■	■	■	■	■	■	■	■	■	1/13
	KGW δ=1	■	■	■	■	■	■	■	■	■	■	■	■	■	3/13
	KGW δ=2	■	■	■	■	■	■	■	■	■	■	■	■	■	5/13
	KGW δ=4	■	■	■	■	■	■	■	■	■	■	■	■	■	10/13
	SynthID-Text	■	■	■	■	■	■	■	■	■	■	■	■	■	7/13
	TextSeal	■	■	■	■	■	■	■	■	■	■	■	■	■	7/13
	Gumbel-max	■	■	■	■	■	■	■	■	■	■	■	■	■	9/13
	Unmarked control	■	■	■	■	■	■	■	■	■	■	■	■	■	0/13
LLMmap	KGW δ=0.5	■	■	■	■	■	■	■	■	■	■	■	■	■	0/13
	KGW δ=1	■	■	■	■	■	■	■	■	■	■	■	■	■	0/13
	KGW δ=2	■	■	■	■	■	■	■	■	■	■	■	■	■	1/13
	KGW δ=4	■	■	■	■	■	■	■	■	■	■	■	■	■	4/13
	SynthID-Text	■	■	■	■	■	■	■	■	■	■	■	■	■	1/13
	TextSeal	■	■	■	■	■	■	■	■	■	■	■	■	■	2/13
	Gumbel-max	■	■	■	■	■	■	■	■	■	■	■	■	■	3/13
	Unmarked control	■	■	■	■	■	■	■	■	■	■	■	■	■	0/13

D.3 Interaction Changes

Table 19: **Per-instruction decisions under system-prompt changes.** Disaggregates Table 6. Each cell compares a model without an added instruction with the same model under one SystemChat instruction, so every detection is a false detection. Column labels summarize the instructions. Row counts give the per-model entries, and the *All* rows give the per-length entries and the overall entry of Table 6.

		Short					Medium					Long					
Method	Model	General helpful assistant	Expert mathematics tutor	Explain to a first grader	Answer in exactly five words	Respond in markdown format	Career guidance for AI professionals	Philosophy teacher, practical applications	Reply with rhymes	Respond with JSON objects	Use LaTeX math equations	Python tutor for projects	Step-by-step math tutor	Empathetic, emotionally intelligent replies	Structured JSON with fixed keys	Markdown software documentation	Count
MET	Llama-3.2-1B	■	■	□	□	■	□	□	□	■	□	■	□	□	□	□	10/15
	Qwen3-4B	■	□	□	□	□	□	□	□	□	□	□	□	□	□	□	14/15
	Phi-4-mini	□	□	□	□	□	□	□	□	■	□	□	□	□	□	□	14/15
	All			11/15					13/15					14/15			38/45
RUT	Llama-3.2-1B	■	□	□	□	□	□	□	□	□	□	□	□	□	□	□	14/15
	Qwen3-4B	■	□	□	□	□	□	□	□	□	□	□	□	□	□	□	14/15
	Phi-4-mini	■	■	□	□	■	■	□	□	□	□	□	□	□	□	□	11/15
	All			10/15					14/15					15/15			39/45
AgentProv	Llama-3.2-1B	■	□	□	□	□	□	□	□	□	□	□	□	□	□	□	14/15
	Qwen3-4B	■	□	□	□	□	□	□	□	□	□	□	□	□	□	□	14/15
	Phi-4-mini	■	■	□	□	■	■	□	■	□	□	■	□	□	□	□	9/15
	All			10/15					13/15					14/15			37/45
LLMmap	Llama-3.2-1B	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	15/15
	Qwen3-4B	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	15/15
	Phi-4-mini	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	15/15
	All			15/15					15/15					15/15			45/45

Table 20: **Per-model decisions under agent harnesses.** Disaggregates the upper block of Table 7. Each cell compares a bare reference with the same model under a harness, so every detection is a false detection. MET and RUT use completed responses only. AgentProv uses the harness with added probe tools.

	DeepSeek					Codex Default					Count	
Method	Qwen2.5-3B	Qwen3.5-4B	Qwen3-4B	Qwen2.5-7B	Llama-3.2-1B	Qwen2.5-3B	Qwen3.5-4B	Qwen3-4B	Qwen2.5-7B	Llama-3.2-1B	DeepSeek	Codex Default
MET	■	■	■	■	■	■	■	■	■	■	5/5	4/5
RUT	■	■	■	■	■	■	■	■	■	■	5/5	5/5
AgentProv	■	■	■	■	■	■	■	■	■	■	5/5	5/5
LLMmap	■	■	■	■	■	■	■	■	■	■	5/5	5/5

Table 21: **Per-pair decisions for model substitution under agent harnesses.** Disaggregates the lower block of Table 7. Rows give the claimed reference model and columns the target model; reference and target use the same harness configuration, so detections are desirable. Dashes mark same-model pairs, which are not substitutions. The *All* rows give the entries of that block. MET and RUT use completed responses only. AgentProv uses the harness with added probe tools.

		DeepSeek target					Codex Default target				
Method	Reference	Qwen2.5-3B	Qwen3.5-4B	Qwen3-4B	Qwen2.5-7B	Llama-3.2-1B	Qwen2.5-3B	Qwen3.5-4B	Qwen3-4B	Qwen2.5-7B	Llama-3.2-1B
MET	Qwen2.5-3B	–	■	■	■	■	–	■	■	■	■
	Qwen3.5-4B	■	–	■	■	■	■	–	■	■	■
	Qwen3-4B	■	■	–	■	■	■	■	–	■	■
	Qwen2.5-7B	■	■	■	–	■	■	■	■	–	■
	Llama-3.2-1B	■	■	■	■	–	■	■	■	■	–
	All			20/20					20/20		
RUT	Qwen2.5-3B	–	■	■	■	■	–	■	■	■	■
	Qwen3.5-4B	■	–	■	■	■	■	–	■	■	■
	Qwen3-4B	■	■	–	■	■	■	■	–	■	■
	Qwen2.5-7B	■	■	■	–	■	■	■	■	–	■
	Llama-3.2-1B	■	■	■	■	–	■	■	■	■	–
	All			19/20					20/20		
AgentProv	Qwen2.5-3B	–	■	■	■	■	–	■	■	■	■
	Qwen3.5-4B	■	–	■	■	■	■	–	■	■	■
	Qwen3-4B	■	■	–	■	■	■	■	–	■	■
	Qwen2.5-7B	■	■	■	–	■	■	■	■	–	■
	Llama-3.2-1B	■	■	■	■	–	■	■	■	■	–
	All			20/20					20/20		
LLMmap	Qwen2.5-3B	–	■	■	■	■	–	■	■	■	■
	Qwen3.5-4B	■	–	■	■	■	■	–	■	■	■
	Qwen3-4B	■	■	–	■	■	■	■	–	■	■
	Qwen2.5-7B	■	■	■	–	■	■	■	■	–	■
	Llama-3.2-1B	■	■	■	■	–	■	■	■	■	–
	All			20/20					20/20		