Auditor-Assisted Summary-Channel Verification for Hosted LLM Identity Substitution

Ziyi Zhang
Stony Brook University
ziyi.zhang.3@stonybrook.edu

Ziyao Wang
University of Maryland, College Park
ziyaow@umd.edu

Guoheng Sun
University of Maryland, College Park
ghsun@umd.edu

Ang Li
University of Maryland, College Park
angliece@umd.edu

Jian Li
Stony Brook University
jian.li.3@stonybrook.edu

Abstract

Hosted LLM services increasingly expose model identity as a product claim, but external users and auditors often cannot inspect which model configuration served a request. We study an auditor-assisted regime in which a trusted auditor enrolls claimed identities and prepares verification material before deployment, while audit-time verification uses only the claimed identity and released output, without runtime access to model internals, inference logs, routing decisions, or provider metadata. We propose SUMMARK, a *summary-channel watermarking* framework that uses user-visible reasoning summaries as a keyed audit surface. During setup, SUMMARK constructs identity-specific vocabulary carriers, trains lightweight summary adapters, and calibrates per-identity thresholds. At deployment time, a keyed statistical test assesses whether the released summary is consistent with the claimed identity and, when needed, attributes the most likely enrolled identity. Across model families, scales, and task domains, SUMMARK reliably detects cross-family and cross-scale substitutions at low false-positive rates while preserving task-facing answer quality. We further study FP16→INT8/INT4 precision-consistency diagnostics and robustness under bounded post-processing, rewriting stress tests, and text-only spoofing, revealing both strong performance and clear failure boundaries. Our code is publicly available at `https://github.com/trinitr0toluene/Summark`.

1 Introduction

Hosted Large Language Model (LLM) services increasingly offer *model selection* as a product feature: users or enterprise customers request a named model family and parameter scale, and receive responses through a standardized interface. This creates an incentive for cost-motivated *identity substitution*, where a provider continues to claim the requested model identity while serving responses through a different and often cheaper model configuration. Detecting such behavior from final answers alone is difficult because related models can produce superficially similar outputs for many prompts, and because surface style can be smoothed by prompting, decoding, or response normalization [3]. This motivates externally checkable evidence for auditing claimed identities in hosted LLM services, especially when deployment-time access to provider internals is unavailable.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

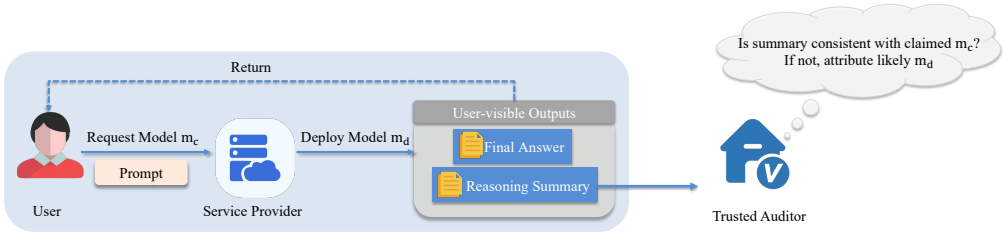

Figure 1: **Auditor-assisted summary-channel verification.** The user requests a claimed model identity m and receives a task-facing answer with a user-visible reasoning summary. Given the claim and released summary, a trusted auditor tests whether the summary is statistically consistent with m using secret verification material prepared offline. When answer and summary generation use separate pipelines, the guarantee is summary-channel consistency. End-to-end answer provenance requires additional coupling or deployment-side attestation.

Existing approaches occupy two broad extremes. At one end, pure output-behavior audits require no privileged setup or runtime access: they query the service and try to infer identity from response behavior, stylistic fingerprints, or challenge-response signatures [3, 9, 20, 12]. However, such evidence is unkeyed, prompt-dependent, and brittle under response variability, adaptive routing, provider-side normalization, or paraphrasing [3, 17]. At the other end, infrastructure-level verification can provide stronger evidence by relying on trusted instrumentation, execution traces, inference logs, cryptographic or trusted-execution mechanisms, or deployment-side attestation [28, 29, 5]. These mechanisms are powerful when available, but they require per-request runtime visibility into the provider-controlled serving stack, which may be unavailable to external auditors and can conflict with privacy, safety, or proprietary constraints [4].

We study the middle ground between these extremes, which we call *onboarded-but-runtime-blind auditing*. In this regime, a trusted auditor enrolls a finite set of claimed identities and prepares secret verification material during a one-time offline setup phase, but deployment-time auditing must operate from public response fields without runtime access to model internals, inference logs, routing decisions, or provider-reported metadata. Offline enrollment defines the configured identities; runtime auditing checks whether a returned auditable field is statistically consistent with the claimed identity. This regime captures settings such as third-party audits, regulatory evaluations, contractual reviews, or provider-volunteered transparency programs, where certified identities may be exposed during setup while the production serving stack remains unavailable on every request.

In this paper, we focus on one user-visible field as the audit carrier: *reasoning summaries* accompanying final answers. Reasoning summaries are increasingly exposed by hosted reasoning services as concise natural-language explanations while full chains-of-thought remain hidden. For example, OpenAI's reasoning models [19] and Google's Gemini thinking models [11] support summarized reasoning. These summaries are observable at deployment time, less tied to task correctness than final answers, and separate from the task-facing answer the user relies on. Our choice of summaries is therefore a carrier-design choice: summaries provide a user-visible, semantically related, and flexible field where audit evidence can be placed without directly perturbing the task-facing answer.

A natural way to turn this field into externally checkable evidence is through *watermarking*, applied to reasoning summaries rather than task-facing answers. Existing LLM watermarks typically embed key-dependent statistical biases into generated text and detect them through token-frequency or likelihood-based tests [16, 35, 7, 18]. When applied to final answers, however, the watermark carrier is tightly coupled to prompt semantics, formatting constraints, lexical entropy, and downstream post-processing. This can dilute detectability, create utility tradeoffs, and increase sensitivity to paraphrasing or rewriting [21, 25, 15, 6]. In contrast, *summary-channel watermarking* places keyed audit evidence in a separate user-visible field, enabling deployment-time verification while leaving task-facing answers minimally affected. This perspective leads to the following question:

> *Can the released reasoning summary serve as a keyed measurement surface for deployment-time identity-consistency checks in an onboarded-but-runtime-blind hosted LLM service, without watermarking task-facing answers or requiring runtime access to internal traces?*

We introduce SUMMARK, a *summary-channel watermarking* framework for auditor-assisted identity-consistency verification over a finite, pre-enrolled set of identities (Figure 1). During offline setup, the auditor uses a secret key to construct identity-specific vocabulary buckets, train lightweight summary watermark adapters, and calibrate per-identity decision thresholds. At deployment time, given a claimed identity and a released reasoning summary, the auditor applies a keyed statistical test to assess identity consistency. The same statistic supports both detection of identity-changing substitutions and attribution of the most likely generating identity among enrolled candidates. The deployment-time verifier requires no access to model parameters, activations, hidden traces, inference logs, runtime routing decisions, or provider-reported metadata.

This formulation defines the verification target of SUMMARK. The verifier provides statistical evidence that the released reasoning-summary field is consistent with the claimed enrolled identity under the auditor-assisted setup above. SUMMARK is therefore a deployment-time summary-channel audit layer: stronger than unkeyed behavioral testing over ordinary outputs, but complementary to infrastructure-level attestation when full serving provenance is required. Our evaluation shows that this summary-channel signal supports reliable low-FPR detection and attribution for identity-changing substitutions across model family and scale, while preserving task-facing answer quality. We further use post-processing, precision-variant, and spoofing stress tests to characterize where the signal remains stable and where it begins to degrade. Our main contributions are as follows:

• **Auditor-assisted verification regime.** We formulate an onboarded-but-runtime-blind regime for hosted LLM identity claims: a trusted auditor enrolls identities and prepares keyed verification material offline, while audit-time decisions use released summaries without runtime access to provider logs, traces, routing decisions, model internals, or self-reported metadata.

• **Reasoning summaries as a keyed audit surface.** We identify the user-visible reasoning-summary field as a practical audit carrier that is externally observable at deployment time and can hold verification evidence while isolating the task-facing answer from watermark perturbation.

• **Identity-specific summary-channel watermarking.** We instantiate this idea in SUMMARK, which uses keyed, identity-specific vocabulary carriers and lightweight summary adapters to support low-FPR substitution detection and multi-class attribution over enrolled identities.

• **Boundary-aware evaluation.** Across model families, scales, and task domains, we evaluate SUMMARK under identity-changing substitutions, attribution, answer-quality preservation, behavioral and generic watermarking baselines, precision-variant diagnostics, and robustness stress tests. The results show strong low-FPR summary-channel verification while exposing clear boundaries under stronger post-processing and spoofing.

2 Threat Model

We consider an auditor-assisted model identity auditing setting with three roles. A *service provider* hosts an LLM service and exposes a user-facing interface; an *end user* issues prompts and consumes responses; and a *trusted auditor* verifies model identity on behalf of users, customers, or regulators. For each request, the provider returns (i) a task-facing final answer and (ii) a user-visible reasoning summary, while presenting the response under a claimed identity m_c. We define a model identity by its $(\text{family}, \text{parameter scale})$ pair. Numerical precision is treated separately as a within-identity deployment variant rather than as a base identity. The end user holds no secret material and interacts solely through the public interface. At deployment time, the auditor observes only the claimed identity m_c and the released reasoning summary s.

Auditor access and certification target. The auditor's access is split into two phases. During a one-time *offline setup*, the auditor has trusted reference access to enrolled identities, sufficient to prepare per-identity verification material and calibration thresholds; concrete procedures are described in Section 3. During *deployment-time auditing*, the auditor's decision depends only on (m_c, s) and a secret verification key. The auditor does not observe model parameters, internal activations, hidden reasoning states, inference logs, runtime routing decisions, or provider-reported metadata.

The audit target is the released reasoning-summary channel. Given (m_c, s), the auditor tests whether s is statistically consistent with the claimed identity under the auditor's keyed verification material. This summary-only decision does not, by itself, certify the provenance of the task-facing answer when answer generation and summary generation are decoupled. End-to-end serving provenance

requires additional answer-summary coupling, deployment policy, or infrastructure-level attestation, which is complementary to the statistical audit layer studied here.

Provisioning and trust assumptions. The auditor's verification material is provided to the serving stack only through authenticated invocation: the provider may use the authorized material during legitimate serving, but cannot extract the secret key, carrier sets, adapter weights, or reusable offline material for other identities. This invocation channel may be instantiated by an auditor-hosted service, a protected serving wrapper, or another authenticated provisioning mechanism. SUMMARK evaluates the statistical audit layer under this access model rather than proposing a full enforcement protocol. This setting is intended for provider-participating audits, such as external audits, procurement checks, regulatory reviews, or voluntary transparency programs. It is stronger than pure API-only access but weaker than full runtime attestation. We assume the auditor's key store is not compromised and the provisioning channel is authenticated.

Adversary (provider). The provider may be cost-motivated and may attempt to make the auditor accept the claimed identity m_c even when the auditable summary channel is generated under, or made to appear consistent with, a different served identity $m_d \neq m_c$. A stronger form of evasion is *spoofing*, in which the provider tries to reproduce the verification signal expected under m_c using a substituted model. We consider the following capabilities.

Identity substitution. Let $\mathcal{M}$ be a finite set of enrolled identities at the family$\times$scale granularity. The adversary may perform *cross-scale substitution*, where the served identity changes parameter scale within the same model family; *cross-family substitution*, where the served identity changes model family; or *joint family-and-scale substitution*, where both attributes differ, e.g., serving Qwen-1.5B while claiming LLaMA-3B. These are the primary identity-substitution events studied in this paper.

Precision variants. Our primary identity universe treats model identity as a family-scale pair. Numerical precision changes, such as FP16→INT8 or FP16→INT4, preserve this identity and are therefore not counted as identity-substitution events. We evaluate them separately as *precision-consistency diagnostics*: for a fixed identity m, the verifier calibrated on FP16 summaries tests whether summaries from a quantized deployment remain consistent with that calibration. Mechanism details and results appear in Sections 3 and 4.4.

Summary post-processing. The primary post-processing model allows *bounded online transformations* to the released summary, such as formatting edits, deletion, truncation, or sentence reordering. These transformations preserve the released text as the reasoning-summary field for the same request and test whether the verifier depends on exact phrasing or sentence order. Stronger transformations, such as full LLM-based paraphrase rewriting or fine-tuning-based watermark removal, are not part of the primary bounded post-processing model because they regenerate or substantially alter the summary rather than preserving it up to surface edits. We evaluate them as stress tests in Section 4.5 to characterize the boundary of text-based summary verification.

Watermark spoofing. The provider may observe or query watermarked summaries produced under the claimed identity, but does not obtain the auditor's secret key, carrier sets, adapter weights, or reusable offline access to the claimed identity's verification material. The adversary may then try to reproduce m_c's verification signal on a substituted model m_d from text-only observations. We evaluate two spoofing strategies in Section 4.5: *adapter distillation*, which trains a local generator on m_d from observed prompt-summary pairs, and *watermark stealing*, which estimates token-level signal statistics from watermarked texts and biases decoding accordingly. These attacks test whether the claimed verification signal can be forged from text-only observations alone.

3 SUMMARK Framework

We present SUMMARK, a summary-channel framework for auditor-assisted identity-consistency verification (Figure 2). The central idea is to *turn the released reasoning summary into a keyed measurement surface.* Each enrolled identity receives a secret vocabulary carrier, and a lightweight summary adapter imprints that carrier into its summaries. At deployment time, the verifier receives only the claimed identity m_c, released summary s, and auditor's secret key, and tests whether the summary's token statistics match the carrier calibrated for m_c.

SUMMARK has three components. *First*, the auditor constructs identity-specific carrier sets $\{G_m\}_{m\in\mathcal{M}}$ with controlled density and limited cross-identity overlap. *Second*, for each enrolled

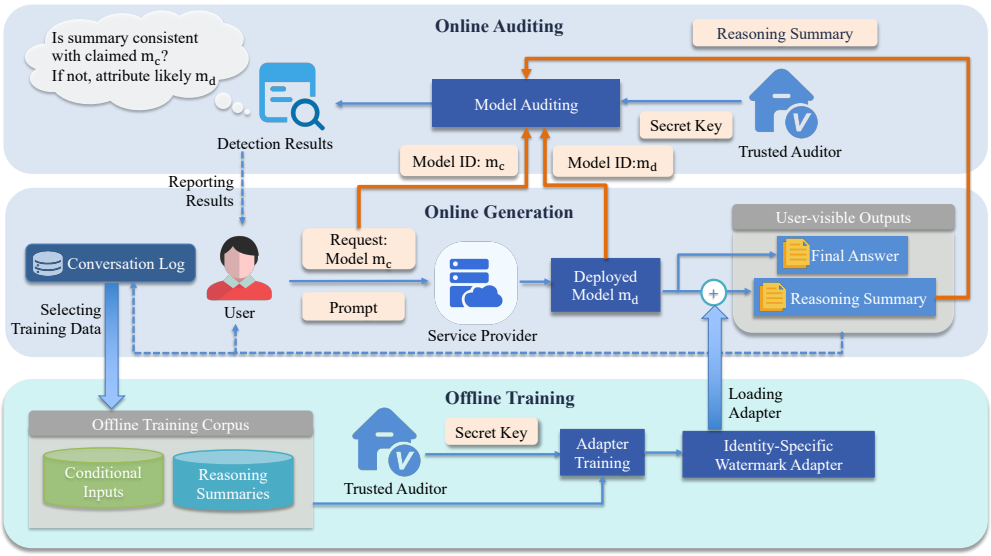

Figure 2: **Workflow of SUMMARK.** Offline, the trusted auditor uses a secret key to construct identity-specific vocabulary carriers, train summary watermark adapters, and calibrate thresholds. Online, the provider returns a task-facing answer and reasoning summary under a claimed identity. The auditor uses only the claim, released summary, and secret key to test summary-channel consistency and, when needed, attribute the most likely enrolled identity.

identity, the auditor trains a lightweight adapter Δ_m that increases the corresponding carrier footprint while preserving fluency and faithfulness. *Third*, the auditor calibrates thresholds and performs a keyed test for detection or attribution using only the released summary. Unlike answer-level watermarking, SUMMARK localizes audit evidence to the reasoning summary. The final answer is the task-facing output, so watermarking it can create utility or formatting tradeoffs. The reasoning summary is an auxiliary user-visible field with more lexical slack, allowing SUMMARK to embed identity-specific evidence while minimally affecting the task-facing answer.

3.1 Problem Definition

Let $\mathcal{M}$ be a finite set of enrolled identities, where each $m \in \mathcal{M}$ specifies a model family and parameter scale. During offline setup, the auditor prepares identity-specific verification material for each m: a carrier set $G_m \subseteq V$ over the vocabulary V, a summary watermark adapter Δ_m, and a detection threshold calibrated for a target false-positive rate. At deployment time, for a user prompt x, the provider returns a response $r = (y, s)$ consisting of a task-facing answer y and a user-visible reasoning summary s, while presenting the response under a claimed identity $m_c \in \mathcal{M}$. The verifier receives only (m_c, s) and the auditor's secret key; it does not access model parameters, activations, logits, hidden reasoning states, inference traces, runtime routing decisions, system logs, or provider-reported metadata.

Detection. Given (m_c, s), the verifier tests whether the released summary is statistically consistent with the claimed identity m_c or is more consistent with a different enrolled identity $m_d \neq m_c$.

Attribution. Given s and $\mathcal{M}$, identity attribution selects the enrolled identity whose keyed carrier statistic best explains the released summary.

Substitution types. We consider three identity-changing substitution categories over $\mathcal{M}$: (i) *cross-family substitution*, where the served identity changes model family; (ii) *cross-scale substitution*, where the served identity changes parameter scale within the same family; and (iii) *joint family-and-scale substitution*, where both attributes differ, e.g., serving Qwen-1.5B while claiming LLaMA-3B.

Precision variants. Our primary task treats identities as family-scale pairs. We separately study whether the FP16-calibrated verifier for a fixed identity m provides useful consistency evidence when the same identity is served at INT8 or INT4 precision. This is not an attribution task over $\mathcal{M}$: all

precision variants reuse the same carrier set G_m and adapter Δ_m, and the verifier flags a variant only when its summary statistics deviate from FP16 calibration.

3.2 Identity-Specific Summary Carriers

Core idea. SUMMARK assigns each identity $m \in \mathcal{M}$ a secret, key-derived subset of vocabulary tokens $G_m \subseteq V$, which we call its *carrier set*. During summary generation, the provider applies the identity-specific adapter Δ_m to nudge token usage toward G_m while maintaining a fluent and faithful summary. This produces an identity-dependent footprint that is detectable from summary text alone using the auditor's key.

Audit-time statistic. Let $s = (s_1, \ldots, s_T)$ be the released summary. For identity m, the verifier computes the green-match count $C_m(s) = \sum_{t=1}^{T} \mathbb{I}[s_t \in G_m]$, and converts it to a calibrated score $Z_m(s) = (C_m(s) - \mu_m(T))/\sigma_m(T)$ using held-out summaries for identity m. Detection compares $Z_{m_c}(s)$ to a per-identity threshold chosen for a target false-positive rate, while attribution selects the identity with the largest score. Full calibration details are in Appendix D.3.

Constructing many separable carrier sets. A single global greenlist would create a shared signal across all models and does not support reliable attribution. SUMMARK instead uses the secret key K to construct a family of carrier sets $\{G_m\}_{m \in \mathcal{M}}$ with two tunable properties:

▷ **Controlled green ratio.** Each identity receives a carrier set of comparable density:

$$|G_m|/|V| \approx \rho, \quad \forall m \in \mathcal{M}, \tag{1}$$

which stabilizes the null baseline across identities.

▷ **Controlled cross-identity overlap.** The construction also limits cross-identity collisions:

$$\frac{|G_m \cap G_{m'}|}{|G_m|} \leq \tau, \quad \forall m \neq m', \tag{2}$$

so different identities remain distinguishable even when the auditor sees only short summaries.

The pair (ρ, τ) governs the trade-off between watermark strength and identity capacity: larger ρ increases per-token evidence, while smaller τ improves separability across identities. This differs from a shared watermark: a shared greenlist can indicate that some watermark is present, but cannot distinguish which enrolled identity produced the summary. Controlled overlap is what turns watermark detection into model-identity attribution. We provide a concrete keyed construction and a capacity characterization in Appendix C.2.

3.3 Training Summary Watermark Adapters

Embedding mechanism. To realize an identity-dependent bias in the summary channel, SUMMARK trains a lightweight adapter for each enrolled identity. For each identity $m \in \mathcal{M}$ with base weights θ_m, we train a summary adapter Δ_m while keeping θ_m frozen. At deployment time, the provider enables Δ_{m_c} only when generating the reasoning summary for a request that claims identity m_c.

Training objective. Adapter training balances three goals: (i) *utility*, so summaries remain fluent and faithful; (ii) *detectability*, so the summary exhibits an elevated footprint of G_m; and (iii) *separability*, so the correct identity's footprint exceeds competing identities' footprints. We instantiate the base identity objective as

$$L = L_{\text{sft}} + \lambda_{\text{wm}} L_{\text{wm}} + \lambda_{\text{sep}} L_{\text{sep}}. \tag{3}$$

Here, L_{sft} is a standard supervised fine-tuning loss on reference summaries, L_{wm} encourages increased usage or probability mass of the identity-specific carrier set G_m, and L_{sep} penalizes confusion by enforcing a margin between the footprint of G_m and competing carrier sets. Intuitively, L_{wm} makes the claimed identity detectable, while L_{sep} makes it distinguishable from other enrolled identities. Full definitions are provided in Appendix D.

Optional precision-consistency regularizer. For the sidecar precision analysis described in Section 3.1, SUMMARK reuses the same carrier set G_m and the same adapter Δ_m across the FP16, INT8, and INT4 variants of identity m. We do not introduce precision-specific carrier sets or precision-specific adapters. Instead, to make the FP16-calibrated verifier sensitive to precision-induced changes, we optionally add a lightweight regularizer that increases the separation between the carrier statistic

observed under FP16 forward passes and the statistic observed under simulated quantized forward passes of the same identity. This regularizer does not create separate precision identities; it only makes the single identity-level verifier more informative for precision-consistency diagnostics. The full objective is provided in Appendix D, and Appendix E.3 ablates its effect.

Why adapters? Adapters make the watermark (a) cheap to provision and switch per identity, (b) tied to the enrolled model family and scale rather than a generic decoding-time bias, and (c) localized to summary generation so task-facing answers are minimally affected. LoRA adapters are shape-bound to enrolled bases, making cross-family or cross-scale cross-loading incompatible in our setting; same-shape precision variants remain compatible and are treated as diagnostics. Our framework is not tied to LoRA specifically: any mechanism that induces controlled, identity-dependent token preferences in summary generation is compatible.

Takeaway. Taken together, the carrier construction and adapter objective turn ordinary watermark detection into identity-specific verification. A generic watermark can indicate that a text was watermarked, but not which enrolled identity it is consistent with. SUMMARK's key contribution is to make the summary watermark separable across identities and auditable from text alone at strict false-positive rates.

4 Experiments

We organize the evaluation around five claims: reliable low-FPR substitution detection, multi-class attribution, superiority over behavioral and generic watermarking baselines, task-facing answer-quality preservation, and boundary behavior under precision variants, post-processing, rewriting, and spoofing. Section 4.2 addresses detection, attribution, and answer quality; Section 4.3 isolates the role of identity-specific summary watermarking; and Sections 4.4–4.5 study boundary cases. Appendix E.3 provides ablations on watermark design and training loss components. We further include a qualitative case study in Appendix E.4, showing how the adapter leaves an identity-specific token pattern in the summary text.

4.1 Experimental Settings

Models and datasets. We evaluate four enrolled identities: Qwen2.5-1.5B, Qwen2.5-3B, Qwen2.5-7B, and LLaMA3.2-3B. Appendix E.1 extends this to 12 identities across four model families, with a separate budget of 1,000 examples per identity. We use three datasets that induce distinct summary styles: Math for mathematical reasoning, Code for programming, and General for general problem solving. These domains differ in both task content and summary structure: Code summaries tend to be more templated and procedural, while General summaries are more stylistically varied and conversational. Full dataset details appear in Appendix B.1.

Substitution and precision-variant settings. An identity $m \in \mathcal{M}$ is a family-scale pair. In the primary substitution benchmark, the provider serves a different identity while claiming m_c, covering cross-scale, cross-family, and joint family-and-scale substitutions. Unless stated otherwise, the substitution ratio is 50%. Separately, we evaluate *precision-consistency diagnostics*: the claimed configuration is $(m, \mathrm{FP16})$, while the served configuration is $(m, \mathrm{INT8})$ or $(m, \mathrm{INT4})$ for the same identity m. Precision variants reuse the same carrier set G_m and adapter Δ_m, so they are evaluated as within-identity consistency checks rather than multi-class attribution over $\mathcal{M}$.

Baselines. We compare against two classes of baselines. First, *text-classifier baselines* operate on released summaries without watermarking: BERT [8], GPT-2 [22], and T5 [23] are trained to predict consistency with a claimed identity from summary text alone. These baselines test whether unkeyed surface or stylistic cues are sufficient for identity auditing. Second, *watermark baselines* operate on the same summary channel: Plain (no watermark), Greenlist (one shared greenlist across identities), Unigram [35], Gumbel-Max [1], and SynthID [7]. Unlike SUMMARK, these methods do not provide identity-specific carrier separability across enrolled identities.

Metrics. For substitution detection, we report TPR at a fixed FPR of 1% (TPR@1%), the corresponding accuracy at that threshold (ACC@1%), and AUC. TPR@1% is the primary metric because practical auditing requires very low false-alarm rates. For attribution, we report multi-class accuracy

Table 1: Summary of substitution detection across seven substitution pairs per dataset. We report average (AvgT) TPR@FPR=1%, minimum (MinT) TPR@FPR=1%, and average AUC. Full per-pair results are in Appendix E.1.

	Math			General			Code		
Method	**AvgT**	**MinT**	**AUC**	**AvgT**	**MinT**	**AUC**	**AvgT**	**MinT**	**AUC**
BERT	0.7683	0.1232	0.9840	0.5439	0.0877	0.9401	0.6567	0.0000	0.9916
GPT-2	0.3333	0.0000	0.7969	0.2506	0.0000	0.8101	0.3886	0.0000	0.8174
T5	0.8676	0.6359	0.9787	0.5213	0.0000	0.9703	0.7494	0.1053	0.9975
SUMMARK	**0.9994**	**0.9960**	**0.9996**	**0.9750**	**0.8740**	**0.9990**	**1.0000**	**1.0000**	**1.0000**

Table 2: Multi-class model identity attribution from reasoning summaries.

	Math		Code		General	
Method	**ACC**	**F1**	**ACC**	**F1**	**ACC**	**F1**
BERT	0.9386	0.9383	0.9580	0.9583	0.9201	0.9201
GPT-2	0.7121	0.6981	0.5768	0.5754	0.7095	0.6894
T5	0.9128	0.9122	**0.9841**	**0.9841**	0.9575	0.9575
SUMMARK	**0.9958**	**0.9957**	0.9678	0.9708	**0.9718**	**0.9718**

Table 3: Task performance with and without summary watermarking (WM). Higher is better for Acc and Pass@1; lower is better for PPL.

	Math (Acc ↑)		General (PPL ↓)		Code (Pass@1 ↑)	
Model	**w/o WM**	**w/ WM**	**w/o WM**	**w/ WM**	**w/o WM**	**w/ WM**
LLaMA-3B	0.2069	**0.2159**	8.1044	**7.8450**	0.0290	**0.0430**
Qwen-1.5B	**0.2971**	0.2967	29.3089	**25.6346**	**0.0170**	**0.0170**
Qwen-3B	0.4219	**0.4240**	10.1635	**10.1429**	0.0630	**0.0760**
Qwen-7B	0.5034	**0.5062**	14.1778	**13.8621**	**0.1920**	0.1830

and macro-F1. For task performance, we report answer accuracy on Math, Pass@1/unit-test success on Code, and perplexity on General. For spoofing, we report attack success rate (ASR@1%), the fraction of spoofed summaries incorrectly accepted as the claimed identity.

Training configuration. We construct watermark carriers with green ratio $\rho = 0.3$ and maximum overlap $\tau = 0.3$. Adapters are trained on 5,000 samples with an 80/20 train-test split for one epoch using AdamW, learning rate 2×10^{-4}, and temperature 0.7. Loss weights are $\lambda_{\text{wm}} = \lambda_{\text{sep}} = 0.5$. We use a balanced 50% mixture and provide sensitivity to substitution ratio in Appendix E.5. All experiments were conducted on one NVIDIA RTX 8000 GPU with 48GB memory.

4.2 Identity-Substitution Detection

Table 1 summarizes substitution detection across Math, General, and Code. Each dataset aggregates the same seven substitution pairs: three cross-scale Qwen substitutions, two cross-family same-scale substitutions, and two joint family-and-scale substitutions. SUMMARK achieves consistently high TPR at the strict FPR$= 1\%$ operating point, with average TPR@1% of 0.9994 on Math, 0.975 on General, and 1.000 on Code. More importantly, its minimum TPR@1% remains high across all pairs: 0.996 on Math, 0.874 on General, and 1.000 on Code.

The contrast matches the design goal of SUMMARK. Text classifiers implicitly learn domain- and dataset-specific regularities such as phrasing, formatting, and templates. These cues are strongest in more structured domains such as Code, where T5 achieves high AUC, but they are not identity-bound and can fail under closely related substitutions or domain changes. SUMMARK instead encodes identity-specific evidence into the summary channel through keyed vocabulary carriers and summary adapters, so substitution induces a predictable shift in summary-level token statistics rather than a fragile stylistic change. Full per-pair results, including the original Math, General, and Code tables, are reported in Appendix E.1.

Attribution. Attribution uses the same standardized green-token statistic as detection, scoring the released summary against each enrolled identity and predicting the largest score. Table 2 shows that SUMMARK supports multi-class attribution across datasets, reaching 0.9958/0.9957 ACC/F1 on Math and 0.9718/0.9718 on General. On Code, T5 slightly outperforms SUMMARK, reflecting the strong templatic structure of Code summaries. Nevertheless, SUMMARK remains competitive and does not rely on unkeyed stylistic cues.

Task-facing answer quality. Table 3 shows that summary-channel watermarking largely preserves final-answer quality. On Math and Code, performance is essentially unchanged after watermarking. On General, perplexity sometimes improves slightly, which we view as benign regularization of summary wording rather than improved reasoning. These results support localizing audit evidence to the summary channel rather than perturbing the task-facing answer.

Table 4: Substitution detection on Math with watermarking baselines. We report TPR@FPR=1%, ACC@FPR=1%, and AUC. Setting A→B denotes serving B while claiming A.

	Unigram			Gumbel Max			SynthID			SUMMARK		
Setting	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**
Qwen-7B→Qwen-3B	0.4730	0.7315	0.9523	0.0050	0.4975	0.4828	0.0050	0.4975	0.5293	**1.0000**	**0.9950**	**1.0000**
Qwen-7B→Qwen-1.5B	0.0690	0.5295	0.7288	0.0300	0.5100	0.4968	0.0200	0.5050	0.5506	**0.9960**	**0.9930**	**0.9974**
Qwen-3B→Qwen-1.5B	0.0000	0.4950	0.2753	0.0950	0.5425	0.6150	0.0950	0.5425	0.6174	**1.0000**	**0.9950**	**0.9998**
Qwen-3B→LLaMA-3B	0.0010	0.4955	0.6302	0.0400	0.5150	0.4865	0.0800	0.5350	0.4961	**1.0000**	**0.9950**	**0.9997**
LLaMA-3B→Qwen-3B	0.1650	0.5775	0.7466	0.0000	0.4950	0.5585	0.0000	0.4950	0.5240	**1.0000**	**0.9950**	**1.0000**
Qwen-7B→LLaMA-3B	0.4510	0.7205	0.9214	0.1250	0.5575	0.6000	0.1000	0.5450	0.5987	**1.0000**	**0.9950**	**1.0000**
LLaMA-3B→Qwen-1.5B	0.0660	0.5280	0.6194	0.0000	0.4950	0.5471	0.0200	0.5050	0.5014	**1.0000**	**0.9950**	**1.0000**

Table 5: Within-identity precision consistency across Math, Code, and General. FP16→INT8 denotes serving a quantized variant while claiming the FP16 configuration of the same identity. Precision variants share the same carrier set and adapter; the verifier is calibrated on FP16 summaries.

		Math			Code			General		
Model	**Setting**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**
LLaMA-3B	FP16→INT8	0.9990	0.9960	0.9990	0.9160	0.9530	0.9694	0.9580	0.9740	0.9971
	FP16→INT4	0.9990	0.9995	0.9990	0.9680	0.9790	0.9791	0.9930	0.9920	0.9998
Qwen-1.5B	FP16→INT8	0.1132	0.5521	0.7575	0.0360	0.5130	0.7010	0.4454	0.7182	0.8534
	FP16→INT4	0.3417	0.6663	0.9209	0.4120	0.7010	0.9497	0.7858	0.8884	0.9454
Qwen-3B	FP16→INT8	0.9860	0.9890	0.9973	0.9460	0.9680	0.9952	0.9910	0.9910	0.9964
	FP16→INT4	0.9960	0.9945	0.9986	0.9970	0.9935	0.9997	0.9950	0.9940	0.9989
Qwen-7B	FP16→INT8	0.9910	0.9920	0.9982	0.9910	0.9905	0.9996	0.9530	0.9715	0.9851
	FP16→INT4	0.9990	0.9975	0.9998	1.0000	0.9975	1.0000	0.9810	0.9875	0.9945

4.3 Generic Watermarks Are Not Enough

Table 4 compares SUMMARK with generic watermarking baselines under the same summary-channel substitution-detection protocol. Unlike SUMMARK, these baselines do not create identity-specific carriers. The results show that generic watermarking is insufficient for reliable identity auditing at the strict low-FPR operating point. Unigram shows some aggregate separability, e.g., AUC $= 0.9523$ for Q7→Q3 and 0.9214 for Q7→L3, but its TPR@1% is only 0.4730 and 0.4510 in those cases and drops near zero elsewhere. Gumbel-Max and SynthID are weaker still.

By contrast, SUMMARK achieves TPR@1% between 0.9960 and 1.0000, ACC@1% between 0.9930 and 0.9950, and AUC between 0.9974 and 1.0000 across all seven substitution pairs. The key difference is that SUMMARK tests whether the summary matches the keyed footprint of the *claimed identity*, not merely whether it contains a generic watermark. Identity-specific carrier construction gives enrolled identities separable token footprints, while adapter training amplifies the correct footprint and suppresses confusion with competitors. Appendix E.3 further shows that Plain and shared-Greenlist variants fail at the strict FPR$= 1\%$ operating point, confirming that the signal must be identity-specific and separable.

4.4 Precision-Variant Diagnostics

Table 5 evaluates *within-identity precision consistency*: the provider serves an INT8 or INT4 quantized variant while claiming the FP16 model. Unlike cross-family or cross-scale substitution, precision variants share the same model identity, carrier set G_m, and adapter Δ_m. We therefore treat this as a sidecar diagnostic rather than substitution detection over $\mathcal{M}$: the verifier is calibrated on FP16 reference summaries and flags a quantized deployment only when its released summaries deviate from the FP16-calibrated footprint.

For LLaMA-3B, Qwen-3B, and Qwen-7B, both FP16→INT8 and FP16→INT4 are flagged reliably across Math, Code, and General, with AUCs mostly above 0.96 and TPR@1% often near 1.0. INT4 is generally easier to flag than INT8, consistent with more aggressive quantization inducing a larger shift in the summary-channel footprint. The main exception is Qwen-1.5B: FP16→INT8 remains close to the FP16 footprint, with TPR@1% dropping to 0.1132 on Math, 0.0360 on Code, and 0.4454 on General. INT4 is more detectable but still weaker than for larger identities. These results

position FP16-calibrated precision consistency as a within-identity diagnostic that complements the cross-identity substitution detection studied above. Appendix E.3 ablates the precision-consistency regularizer introduced in Section 3.3.

4.5 Robustness and Boundary Analysis

Table 6 summarizes robustness under summary post-processing, rewriting, and spoofing. For bounded surface transformations, including sentence shuffle, random deletion, and word truncation, detection remains essentially unchanged: the minimum TPR@1% across all edited summaries is 0.9950. This confirms that SUMMARK does not rely on exact phrasing or sentence order; its verifier aggregates summary-level token statistics that survive reordering and partial deletion.

Table 6: Robustness summary on Math across seven substitution pairs. For post-processing, we report average/min TPR@FPR=1%; for spoofing, average/max ASR@FPR=1%. Full results are in Appendix E.2.

Stress test	**Avg.**	**Worst**
Original TPR@1%	0.9994	0.9960
Bounded edits TPR@1%	0.9992	0.9950
Bilingual rewrite TPR@1%	0.7900	0.5740
Distilled spoofing ASR@1%	0.1651	0.6290
Watermark-stealing ASR@1%	0.0417	0.2160

Bilingual rewriting is stronger because it replaces tokens rather than rearranging them. It reduces average TPR@1% to 0.7900 and worst-case TPR@1% to 0.5740, showing that semantic-level rewriting can attenuate the lexical footprint. Thus, bounded edits are well tolerated, while rewriting that replaces many identity-discriminative tokens can reduce low-FPR detectability.

Spoofing gives a similar boundary-aware picture. Watermark-stealing is largely ineffective, with average ASR@1% of 0.0417, suggesting that text-only token-statistic estimates cannot reproduce the keyed footprint. Distilled adapter spoofing is stronger: it is contained in several transfer directions but reaches worst-case ASR@1% of 0.6290. We therefore treat spoofing robustness as empirical rather than guaranteed, and report full per-pair results, hardest transfer directions, and additional watermark removal stress tests in Appendix E.2.

Takeaway. Overall, SUMMARK turns reasoning summaries into a keyed, identity-specific audit signal: it enables reliable low-FPR detection and attribution from released summaries, outperforms shared and generic watermarking baselines, and preserves task-facing answer quality. The boundary analyses further clarify the scope of the guarantee: the signal is stable under bounded surface edits, useful but model-dependent for precision variants, and vulnerable to sufficiently strong rewriting or adaptive spoofing.

5 Conclusion

We introduce SUMMARK, a summary-channel watermarking framework for auditing hosted LLM identity claims from user-visible reasoning summaries. SUMMARK embeds keyed, identity-specific signals into summaries, enabling low-FPR substitution detection and multi-class attribution while preserving task-facing answer quality. Our results also characterize important boundaries: precision-consistency diagnostics are useful but model-dependent, bounded post-processing is well tolerated, and strong rewriting or adaptive spoofing remains challenging. Future work includes richer audit signals beyond text and stronger robustness for fine-grained deployment variants.

Acknowledgements

This work was supported in part by the National Science Foundation (NSF) grants 2541997, 2148309, 2315614 and 2337914, and the National Institutes of Health (NIH) grant R01HL184139. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] Scott Aaronson and Hendrik Kirchner. Watermarking GPT outputs. `https://scottaaronson.blog/?m=202302`, 2023. Unpublished technical note / blog post.

[2] Yossi Adi, Carsten Baum, Moustapha Cisse, 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)*, pages 1615–1631, 2018.

[3] Will Cai, Tianneng Shi, Xuandong Zhao, and Dawn Song. Are you getting what you pay for? auditing model substitution in llm apis. *arXiv preprint arXiv:2504.04715*, 2025.

[4] Stephen Casper, Carson Ezell, Charlotte Siegmann, Noam Kolt, Taylor Lynn Curtis, Benjamin Bucknall, Andreas Haupt, Kevin Wei, Jérémy Scheurer, Marius Hobbhahn, et al. Black-box access is insufficient for rigorous ai audits. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 2254–2272, 2024.

[5] Bing-Jyue Chen, Suppakit Waiwitlikhit, Ion Stoica, and Daniel Kang. Zkml: An optimizing system for ml inference in zero-knowledge proofs. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pages 560–574, 2024.

[6] Ruibo Chen, Yihan Wu, Junfeng Guo, and Heng Huang. De-mark: Watermark removal in large language models. *arXiv preprint arXiv:2410.13808*, 2024.

[7] 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(8035):818–823, 2024.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

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

[10] Mansour Al Ghanim, Jiaqi Xue, Rochana Prih Hastuti, Meng Zheng, Yan Solihin, and Qian Lou. Evaluating the robustness and accuracy of text watermarking under real-world cross-lingual manipulations. *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

[11] Google. Gemini api: Thinking and thought summaries. `https://ai.google.dev/gemini-api/docs/thinking`, 2024. Accessed 2025-01.

[12] Zecheng He, Tianwei Zhang, and Ruby B Lee. Verideep: Verifying integrity of deep neural networks through sensitive-sample fingerprinting. *arXiv preprint arXiv:1808.03277*, 2018.

[13] Yangsibo Huang, Milad Nasr, Anastasios Angelopoulos, Nicholas Carlini, Wei-Lin Chiang, Christopher A Choquette-Choo, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Ken Ziyu Liu, et al. Exploring and mitigating adversarial manipulation of voting-based leaderboards. *arXiv preprint arXiv:2501.07493*, 2025.

[14] Jiawei Jiang, Yi Wei, Yu Liu, Wentao Wu, Chuang Hu, Zhigao Zheng, Ziyi Zhang, Yingxia Shao, and Ce Zhang. How good are machine learning clouds? benchmarking two snapshots over 5 years: J. jiang et al. *The VLDB Journal*, 33(3):833–857, 2024.

[15] Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. *arXiv preprint arXiv:2402.19361*, 2024.

[16] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR, 2023.

[17] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36:27469–27500, 2023.

[18] Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. A survey of text watermarking in the era of large language models. *ACM Computing Surveys*, 57(2):1–36, 2024.

[19] OpenAI. New tools and features in the responses api. https://openai.com/index/new-tools-and-features-in-the-responses-api/, 2025.

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

[21] Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. Markmywords: Analyzing and evaluating language model watermarks. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 68–91. IEEE, 2025.

[22] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[23] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[24] Inioluwa Deborah Raji, Andrew Smart, Rebecca N White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 33–44, 2020.

[25] Saksham Rastogi and Danish Pruthi. Revisiting the robustness of watermarking to paraphrasing attacks. *arXiv preprint arXiv:2411.05277*, 2024.

[26] Guoheng Sun, Ziyao Wang, Bowei Tian, Meng Liu, Zheyu Shen, Shwai He, Yexiao He, Wanghao Ye, Yiting Wang, and Ang Li. Coin: Counting the invisible reasoning tokens in commercial opaque llm apis. *arXiv preprint arXiv:2505.13778*, 2025.

[27] Guoheng Sun, Ziyao Wang, Xuandong Zhao, Bowei Tian, Zheyu Shen, Yexiao He, Jinming Xing, and Ang Li. Invisible tokens, visible bills: The urgent need to audit hidden operations in opaque llm services. *arXiv preprint arXiv:2505.18471*, 2025.

[28] Yifan Sun, Yuhang Li, Yue Zhang, Yuchen Jin, and Huan Zhang. Svip: Towards verifiable inference of open-source large language models. *arXiv preprint arXiv:2410.22307*, 2024.

[29] Florian Tramer and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. *arXiv preprint arXiv:1806.03287*, 2018.

[30] Ziyao Wang, Guoheng Sun, Yexiao He, Zheyu Shen, Bowei Tian, and Ang Li. Predictive auditing of hidden tokens in llm apis via reasoning length estimation. *arXiv preprint arXiv:2508.00912*, 2025.

[31] Yi Wei, Xiaokai Zhou, Shanshan Feng, Chuang Hu, Xiao Yan, and Jiawei Jiang. Laua: Handling missing modalities and unpaired data in multimodal federated learning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5303–5313, 2026.

[32] Zhenghao Xu, Qin Lu, Qingru Zhang, Liang Qiu, Ilgee Hong, Changlong Yu, Wenlin Yao, Yao Liu, Haoming Jiang, Lihong Li, et al. Ask a strong llm judge when your reward model is uncertain. *Advances in Neural Information Processing Systems*, 38:74639–74664, 2026.

[33] Jiyu Yang, Qiang He, Zheyu Zhou, Xiaohai Dai, Feifei Chen, Cong Tian, and Yun Yang. Edgethemis: Ensuring model integrity for edge intelligence. In *Proceedings of the ACM on Web Conference 2025*, pages 3136–3146, 2025.

[34] Qinbo Zhang, Yanhang Shi, Ziyi Zhang, Hao Wang, Sai Qian Zhang, and Jian Li. Stealing split learning bottom models by recovering embedding geometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20660–20669, 2026.

[35] Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*, 2023.

A Additional Related Work Discussions

A.1 LLM Watermarking

LLM Watermarking has been widely studied as a way to embed verifiable provenance signals into generated text, enabling attribution, provenance tracking, or misuse detection while preserving output fluency [7].

Most existing approaches introduce controlled statistical biases at generation time, for example by favoring identity-specific token subsets or by shaping token selection patterns during decoding [16, 35]. Detection is typically performed through hypothesis testing on token frequencies or related likelihood statistics computed from the generated text.

A common design is to embed the watermark directly into task-facing outputs, usually the final answers returned to users. This makes the detectability of the watermark sensitive to task semantics and prompt distributions [21]. In particular, low-entropy generations provide limited lexical degrees of freedom, so injecting a detectable bias can compete with answer fidelity and the watermark signal may be diluted by semantic constraints [18]. Moreover, embedding signals into final answers raises concerns about potential degradation of answer quality and can be weakened by post-processing such as paraphrasing or rewriting [25, 15, 6, 10]. These limitations motivate moving audit evidence away from task answers and into a separate user-visible channel. Our work follows this direction by watermarking *reasoning summaries* and enabling *summary-only* verification, which reduces coupling to task constraints and preserves answer quality.

A.2 Model Auditing

Model auditing aims to verify whether a deployed machine learning system behaves as claimed, or whether specific properties of the model or inference process can be reliably certified [24, 4, 27]. Prior work has explored a range of approaches, including behavioral testing through challenge–response protocols [12, 33], hash tree-based verification [26], output-based fingerprinting using distributional or stylistic cues [9, 20], and verification mechanisms that rely on access to internal model states, inference logs, or trusted execution environments [29, 5]. These methods have been applied to settings such as model integrity verification, inference correctness, compliance auditing [12, 29], and token count auditing [26, 30]. Beyond model-level verification, prior work has explored broader reliability and security challenges in machine learning systems across diverse deployment settings [14, 31, 34, 32].

Recent work has begun to explicitly study model substitution in LLM services, where a provider may serve responses generated by a different model than the one advertised [3]. This problem is particularly challenging in black-box settings, as modern language models with similar architectures or training objectives can exhibit highly overlapping behaviors. Behavioral differences between models from the same family or between different parameter scales may be subtle, task-dependent, and easy to smooth or obscure through prompt engineering or response post-processing [3, 17]. As a result, auditing methods that rely solely on output behavior or stylistic features often provide limited robustness under realistic substitution strategies [3].

Many existing auditing and verification techniques further assume privileged access that is unavailable in hosted LLM services [4]. For example, some approaches require access to internal activations, inference traces, or trusted instrumentation during model execution [29, 5, 9], while others rely on cooperative providers that accurately report model identity or generation metadata [3]. In contrast, end users and external auditors interacting with commercial LLM services typically observe only a fixed, user-facing interface and must perform verification post hoc, without control over prompts, sampling procedures, or internal logging [3, 4].

Our work targets this more constrained but practically relevant auditing regime. We consider auditor-assisted model identity verification at deployment time, where a trusted auditor holds secret verification material but operates solely on user-visible outputs [2]. By focusing on reasoning summaries as the sole audit surface, our approach departs from prior model auditing methods and addresses model substitution under an information regime that closely matches real-world hosted LLM services.

B Additional Experimental Details

Here, we provide additional details on the experimental setup, including the datasets and models used, as well as details of the generation process.

B.1 Datasets

We use the following three datasets [30].

OpenR1-Math. The Math dataset consists of multi-step mathematical reasoning problems requiring symbolic manipulation or logical derivation. Prompts are formatted to elicit a final answer together with a concise reasoning summary. Summaries in this dataset are typically structured and step-oriented, providing a high-density signal for evaluating watermark detectability under strong reasoning conditions.

OpenR1-Code. The Code dataset focuses on program synthesis and algorithmic problem solving. Each prompt requires generating code or an algorithmic solution, accompanied by a natural-language summary explaining the approach. Compared to Math, summaries exhibit greater lexical diversity and less rigid structure, testing robustness under heterogeneous reasoning styles.

General Reasoning. The General dataset includes open-domain reasoning and instruction-following tasks such as commonsense reasoning and explanation-style question answering. Summaries are shorter and more stylistically varied, representing the most challenging setting for summary-based auditing and reflecting common real-world usage.

B.2 Models

Our primary benchmark uses four instruction-tuned checkpoints: Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, and LLaMA-3.2-3B-Instruct. These checkpoints are obtained from public repositories on Hugging Face.

The expanded evaluation in Appendix E.1 adds eight checkpoints, including 12 enrolled identities across four model families: Qwen (0.5B, 1.5B, 3B, 7B, 14B, and 32B), LLaMA (3B and 7B), Mistral (Ministral-3B and Mistral-7B), and Gemma (2B and 7B). The primary benchmark and expanded evaluation use different training budgets, as specified in Section 4.1 and Appendix E.1, respectively.

C Watermark Design Details

C.1 Greenlist-based Watermarking

We model a reasoning summary as a token sequence $s = (s_1, \ldots, s_T)$ drawn from a vocabulary V. SUMMARK assigns each model identity $m \in \mathcal{M}$ a secret subset of tokens, called a *greenlist* $G_m \subseteq V$, derived from the auditor's secret key. During summary generation, the watermark is implemented as an identity-dependent bias that increases the likelihood of sampling tokens from G_m while preserving fluency and semantic fidelity.

At decoding step t, let $z_t \in \mathbb{R}^{|V|}$ be the output logits and $p_t(v) = \mathrm{softmax}(z_t)_v$ be the induced token distribution. We quantify the instantaneous watermark signal via the log probability mass on G_m:

$$\log P(G_m \mid z_t) \triangleq \log \sum_{v \in G_m} e^{z_{t,v}} - \log \sum_{v \in V} e^{z_{t,v}}. \tag{4}$$

Equivalently, $P(G_m \mid z_t) = \sum_{v \in G_m} p_t(v)$ represents the probability that the next generated token belongs to G_m. Section 3.3 describes how identity-specific adapters are trained on supervised summary data to increase the quantity defined in Eq. (4).

C.2 Model Identities Capacity

The parameters ρ and τ constrain how many model identities can be supported for a fixed vocabulary. Let V be the public vocabulary with size $n \triangleq |V|$. For each identity $m \in \mathcal{M}$, the greenlist is a subset

$G_m \subseteq V$ with size $w \triangleq |G_m|$, and the green ratio constraint in Eq. (1) gives $w \approx \rho n$. To quantify collisions across identities, define the maximum pairwise intersection

$$\lambda \triangleq \max_{m \neq m'} |G_m \cap G_{m'}|. \tag{5}$$

Under the max-overlap constraint in Eq. (2), we have $|G_m \cap G_{m'}| \leq \tau |G_m|$ for all $m \neq m'$, hence $\lambda \leq \tau w$. A standard packing bound for constant-size subsets then introduces

$$B \leq \left\lfloor 1 + \frac{w(n-w)}{w^2 - n\lambda} \right\rfloor \leq \left\lfloor 1 + \frac{w(n-w)}{w^2 - n\tau w} \right\rfloor, \tag{6}$$

where $B \triangleq |\mathcal{M}|$ is the number of identities and the bound is informative when the denominator is positive. Eq. (6) shows that smaller τ improves separability but can reduce capacity.

D Watermark Adapters Training

To induce identity-specific token preferences in the *summary channel*, we adopt a lightweight parameterization based on Low-Rank Adaptation (LoRA). For each identity $m \in \mathcal{M}$, we train an adapter Δ_m that biases the summary generator toward the identity-specific greenlist G_m. The base model parameters θ are frozen and only Δ_m is optimized, so the watermark is implemented as a small, identity-specific parameter add-on rather than a separate generation pipeline.

D.1 Training Target

We train each adapter Δ_m using supervised summary data. Each training example $(c, s^*) \sim \mathcal{D}$ consists of a conditioning input c used to generate the summary and a reference summary $s^* = (w_1, \ldots, w_T)$. We use teacher forcing during training. At each decoding step, the model is given the ground-truth prefix of the summary and learns to predict the next token.

At step t, the adapted model $(\theta + \Delta_m)$ outputs a logit vector $z_t \in \mathbb{R}^{|V|}$ for every token in the vocabulary V. Applying a softmax converts these scores into a next-token probability distribution. Formally, we define

$$p_t(v) = \mathrm{softmax}(z_t)_v, \tag{7}$$

which denotes the probability of token v at step t.

Each identity m is associated with a predefined set of tokens, called the greenlist G_m. Intuitively, a watermark is stronger if the model tends to place more probability on these greenlist tokens during generation. We therefore measure the watermark strength by the total probability mass assigned to G_m:

$$P(G_m | z_t) = \sum_{v \in G_m} p_t(v), \tag{8}$$

which is the probability that the next token belongs to the greenlist. For numerical stability and convenience, we use its log-scaled form:

$$\log P(G_m \mid z_t) = \log \sum_{v \in G_m} e^{z_{t,v}} \; - \; \log \sum_{v \in V} e^{z_{t,v}}. \tag{9}$$

Equivalently, $P(G_m \mid z_t) = \sum_{v \in G_m} p_t(v)$ is the probability that the next token sampled from p_t lies in the greenlist G_m. We refer to $P(G_m \mid z_t)$ as the *green mass* at step t, and to $\log P(G_m \mid z_t)$ as its log-scaled form. A larger green mass means the model is more biased toward emitting identity-m greenlist tokens, which results in a stronger and more detectable watermark signal.

During training, we freeze θ and update only Δ_m. This ensures that the watermark is encoded through a small, identity-specific adapter, rather than by modifying or retraining the full model.

D.2 Loss Components

Given training pairs $(c, s^*) \sim \mathcal{D}$, we optimize the summary adapter Δ_m with a base identity objective and an optional precision-consistency regularizer. The base identity objective is

$$L_{\mathrm{id}} = L_{\mathrm{sft}} + \lambda_{\mathrm{wm}} L_{\mathrm{wm}} + \lambda_{\mathrm{sep}} L_{\mathrm{sep}}, \tag{10}$$

where $\lambda_{\text{wm}}, \lambda_{\text{sep}} \geq 0$ control watermark strength and identity separability. Throughout this section, logits z_t are produced by the adapted model $\theta_m + \Delta_m$ under teacher forcing on the reference summary. All losses are averaged over $(c, s^*) \sim \mathcal{D}$.

Utility constraint: L_{sft}. We use the standard teacher-forced negative log-likelihood on reference summaries:

$$L_{\text{sft}} = -\mathbb{E}_{\mathcal{D}} \log P_{\theta_m + \Delta_m}(s^* \mid c). \tag{11}$$

This anchors generation to natural summaries and mitigates semantic drift caused by watermark pressure.

Detectability: L_{wm}. To strengthen the identity-specific watermark, we encourage larger average log probability mass on the carrier set G_m:

$$L_{\text{wm}} = -\mathbb{E}_{\mathcal{D}} \left[\frac{1}{T} \sum_{t=1}^{T} \log P(G_m \mid z_t) \right]. \tag{12}$$

This term increases the statistical footprint associated with identity m, making the watermark more detectable from generated summaries.

Attribution separability: L_{sep}. To reduce confusion between enrolled identities, we enforce that the carrier mass of the correct identity exceeds that of competing identities by a margin μ. Let $\mathcal{N} \subset \mathcal{M} \setminus \{m\}$ be a sampled set of negative identities. We use a hardest-negative margin loss:

$$L_{\text{sep}} = \mathbb{E}_{\mathcal{D}} \left[\frac{1}{T} \sum_{t=1}^{T} \max \left(0,\, \mu + \max_{m' \in \mathcal{N}} \log P(G_{m'} \mid z_t) - \log P(G_m \mid z_t) \right) \right]. \tag{13}$$

This term targets the most confusable competing identity and improves attribution reliability when carrier sets partially overlap.

Precision-consistency regularizer. For precision-variant diagnostics, FP16, INT8, and INT4 variants of the same base identity m share the same carrier set G_m and the same adapter Δ_m. We add a quantization-sensitive regularizer that encourages the FP16 execution to produce a stronger carrier-set footprint than quantized executions.

For a precision state $p \in \{\text{FP16}, \text{INT8}, \text{INT4}\}$, define the average carrier-set score

$$A_p(m) = \frac{1}{T} \sum_{t=1}^{T} \log P_p(G_m \mid z_t^{(p)}), \tag{14}$$

where $z_t^{(p)}$ denotes the logits produced by the shared adapter Δ_m under precision state p. We define

$$L_{\text{qs}} = \mathbb{E}_{q \in \{\text{INT8,INT4}\}} \left[\left(\gamma - \left(A_{\text{FP16}}(m) - A_q(m) \right) \right)_+ \right], \tag{15}$$

where $(u)_+ = \max(0, u)$ and $\gamma > 0$ is a precision-separation margin. This loss penalizes cases where the FP16 carrier-set score is not sufficiently larger than the INT8/INT4 score.

The full precision-sensitive training objective is

$$L = L_{\text{id}} + \lambda_{\text{qs}} L_{\text{qs}}. \tag{16}$$

This regularizer does not create precision-specific identities, carrier sets, or adapters. It only makes the shared identity-level verifier more informative for FP16-to-INT8/INT4 precision-consistency diagnostics.

D.3 Detection

At audit time, the auditor observes only the released summary s and the claimed identity m_c. Let $s = (x_1, \ldots, x_T)$ be the tokenized summary, where each token $x_t \in V$. For each candidate identity $m \in M$ with keyed carrier bucket $G_m \subseteq V$, define

$$g_m(x_t) \triangleq \mathbb{I}[x_t \in G_m], \qquad C_m(s) \triangleq \sum_{t=1}^{T} g_m(x_t).$$

Table 7: Substitution detection on **Math** (TPR@FPR=1%, ACC@FPR=1%, AUC). Setting $A \to B$: serve B, claim A. Bold is best.

	BERT			GPT-2			T5			SUMMARK		
Setting	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**
Qwen-7B→Qwen-3B	0.1232	0.5602	0.9221	0.0000	0.5000	0.4458	0.6359	0.8053	0.9348	**1.0000**	**0.9950**	**1.0000**
Qwen-7B→Qwen-1.5B	0.9412	0.9706	0.9955	0.2105	0.5965	0.6037	0.9804	0.9902	0.9939	**0.9960**	**0.9930**	**0.9974**
Qwen-3B→Qwen-1.5B	0.9720	0.9790	0.9959	0.2807	0.6316	0.8339	0.9804	0.9748	0.9976	**1.0000**	**0.9950**	**0.9998**
Qwen-3B→LLaMA-3B	0.9076	0.9468	0.9939	0.6842	0.8333	0.9120	0.7003	0.8235	0.9605	**1.0000**	**0.9950**	**0.9997**
LLaMA-3B→Qwen-3B	0.8431	0.9202	0.9939	0.3684	0.6754	0.9477	0.8627	0.9300	0.9745	**1.0000**	**0.9950**	**1.0000**
Qwen-7B→LLaMA-3B	0.6050	0.8011	0.9871	0.2281	0.6053	0.8507	0.9272	0.9510	0.9941	**1.0000**	**0.9950**	**1.0000**
LLaMA-3B→Qwen-1.5B	0.9860	0.9930	0.9999	0.5614	0.7719	0.9843	0.9860	0.9930	0.9955	**1.0000**	**0.9950**	**1.0000**

Table 8: Substitution detection on **General** (TPR@FPR=1%, ACC@FPR=1%, AUC). Setting $A \to B$: serve B, claim A. Q7/Q3/Q1.5 = Qwen-7B/3B/1.5B, L3 = LLaMA-3B. Bold is best.

	BERT			GPT-2			T5			SUMMARK		
Setting	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**
Qwen-7B→Qwen-3B	0.7895	0.8772	0.9621	0.0000	0.5000	0.6331	0.7193	0.8596	0.9898	**0.9990**	**0.9945**	**0.9999**
Qwen-7B→Qwen-1.5B	0.1228	0.5439	0.8689	0.0526	0.5263	0.5389	0.0000	0.5000	0.9514	**0.8740**	**0.9320**	**0.9954**
Qwen-3B→Qwen-1.5B	0.0877	0.5439	0.9267	0.1053	0.5526	0.7014	0.0000	0.5000	0.9338	**1.0000**	**0.9950**	**1.0000**
Qwen-3B→LLaMA-3B	0.5439	0.7719	0.9298	0.3509	0.6754	0.9464	0.5088	0.7544	**1.0000**	**1.0000**	**0.9950**	0.9999
LLaMA-3B→Qwen-3B	0.8596	0.9211	0.9917	0.3684	0.6842	0.9372	0.9474	0.9561	0.9800	**1.0000**	**0.9950**	**1.0000**
Qwen-7B→LLaMA-3B	0.5965	0.7807	0.9231	0.3509	0.6754	0.9323	0.5439	0.7719	0.9471	**0.9920**	**0.9910**	**0.9988**
LLaMA-3B→Qwen-1.5B	0.8070	0.8947	0.9785	0.5263	0.7632	0.9812	0.9298	0.9474	0.9898	**0.9600**	**0.9750**	**0.9988**

Although a binomial null would suggest $\mathbb{E}[C_m(s)] \approx \rho T$, all reported decisions use empirical calibration rather than this approximation. For each enrolled identity m, we estimate length-conditioned calibration statistics $\mu_m(T)$ and $\sigma_m(T)$ from held-out benign summaries generated under the enrolled identity and its summary adapter. The calibrated score is

$$Z_m(s) \triangleq \frac{C_m(s) - \mu_m(T)}{\sigma_m(T) + \epsilon},$$

where ϵ is a small numerical constant. Intuitively, summaries consistent with identity m should have calibrated scores comparable to the held-out benign calibration distribution, while summaries generated under substituted identities tend to have lower scores for the claimed carrier bucket.

Model substitution detection. Given a claimed identity m_c, the verifier accepts the claim if

$$Z_{m_c}(s) \geq \eta_{m_c}(\alpha),$$

where $\eta_{m_c}(\alpha)$ is chosen as the lower-tail empirical threshold on held-out benign summaries so that the false-positive rate is at most α. Otherwise, the verifier flags the released summary as inconsistent with the claimed identity. Thus the operating threshold is calibrated from held-out summaries rather than assumed from the binomial approximation.

Model attribution. For attribution, the auditor scores the released summary against every enrolled identity and predicts

$$\widehat{m}(s) \triangleq \arg\max_{m \in M} Z_m(s),$$

optionally abstaining when the top-two score margin is below a fixed threshold. Both detection and attribution rely only on summary-level token statistics and the keyed carrier buckets.

E Additional Experiments

E.1 Identity-Substitution Detection

Per-pair Substitution Detection Results. Tables 7, 8, and 9 report model substitution detection on Math, General, and Code, respectively. The first three rows in each table correspond to *within-family size substitution* (Qwen → Qwen), the next two to *cross-family substitution* (Qwen ↔ LLaMA), and the last two to *combined* substitutions (family + size).

Across datasets and substitution types, SUMMARK is consistently strong under strict auditing constraints, achieving very high TPR@FPR$= 1\%$ in most settings and uniformly high AUC. In

Table 9: Substitution detection on **Code** (TPR@FPR=1%, ACC@FPR=1%, AUC). Setting $A \to B$: serve B, claim A. Bold is best.

	BERT			GPT-2			T5			SUMMARK		
Setting	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**
Qwen-7B→Qwen-3B	0.6491	0.8246	0.9825	0.0000	0.5000	0.5720	0.8947	0.9474	0.9963	**1.0000**	**0.9950**	**1.0000**
Qwen-7B→Qwen-1.5B	0.8246	0.9123	0.9834	0.1400	0.5700	0.6690	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9950	**1.0000**
Qwen-3B→Qwen-1.5B	0.0000	0.5000	0.9769	0.5090	0.7110	0.7960	0.1053	0.5526	0.9982	**1.0000**	**0.9950**	**1.0000**
Qwen-3B→LLaMA-3B	0.2281	0.6140	0.9991	0.7370	0.8250	0.9330	0.3684	0.6842	0.9942	**1.0000**	**0.9950**	**1.0000**
LLaMA-3B→Qwen-3B	0.9474	0.9737	0.9991	0.4040	0.6750	0.9160	0.8947	0.9474	0.9935	**1.0000**	**0.9950**	**1.0000**
Qwen-7B→LLaMA-3B	0.9825	0.9912	**1.0000**	0.2810	0.6400	0.8860	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9950	**1.0000**
LLaMA-3B→Qwen-1.5B	0.9649	0.9825	**1.0000**	0.6490	0.7980	0.9500	0.9825	0.9912	**1.0000**	**1.0000**	**0.9950**	**1.0000**

Table 10: Substitution detection on **Math** (TPR@FPR=1%, ACC@FPR=1%, AUC). Setting $A \to B$: serve B, claim A. Q7/Q3/Q1.5 = Qwen-7B/3B/1.5B, L3 = LLaMA-3B. Bold is best.

	BERT			GPT-2			T5			IDP			SUMMARK		
Setting	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**
$Q7 \to Q3$	0.1232	0.5602	0.9221	0.0000	0.5000	0.4458	0.6359	0.8053	0.9348	0.0000	0.5000	0.5000	**1.0000**	**0.9950**	**1.0000**
$Q7 \to Q1.5$	0.9412	0.9706	0.9955	0.2105	0.5965	0.6037	0.9804	0.9902	0.9939	0.0000	0.5000	0.5000	**0.9960**	**0.9930**	**0.9974**
$Q3 \to Q1.5$	0.9720	0.9790	0.9959	0.2807	0.6316	0.8339	0.9804	0.9748	0.9976	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9950	0.9998
$Q3 \to L3$	0.9076	0.9468	0.9939	0.6842	0.8333	0.9120	0.7003	0.8235	0.9605	0.0000	0.5000	0.5000	**1.0000**	**0.9950**	**0.9997**
$L3 \to Q3$	0.8431	0.9202	0.9939	0.3684	0.6754	0.9477	0.8627	0.9300	0.9745	0.0000	0.5000	0.5000	**1.0000**	**0.9950**	**1.0000**
$Q7 \to L3$	0.6050	0.8011	0.9871	0.2281	0.6053	0.8507	0.9272	0.9510	0.9941	0.0000	0.5000	0.5000	**1.0000**	**0.9950**	**1.0000**
$L3 \to Q1.5$	0.9860	0.9930	0.9999	0.5614	0.7719	0.9843	0.9860	0.9930	0.9955	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9950	**1.0000**

contrast, text classifiers (BERT/GPT-2/T5) are *unstable across settings*: they can show high AUC in some pairs, yet their TPR@1% varies widely, which is most notably for within-family size substitutions where stylistic cues are subtle and easily washed out. This AUC-TPR gap indicates that while some substitutions are separable in aggregate, surface cues are often too brittle to support reliable decisions at a strict operating point (FPR$= 1\%$). For instance on Math, BERT's TPR@1% collapses for Qwen-7B→Qwen-3B (0.1232) while SUMMARK remains 1.0000, illustrating that style-based signals are not reliable when the substituted model is closely related to the claimed one.

This contrast matches the design intent of SUMMARK. Text classifiers implicitly learn domain- and dataset-specific surface regularities (phrasing, formatting, recurring templates), which may transfer poorly across tasks and are weakest when two models share the same family and training pipeline. By comparison, SUMMARK *encodes identity-specific evidence into the summary channel via keyed vocabulary buckets and watermark adapters, so substitution induces a predictable shift in summary-level token statistics rather than a fragile stylistic change.* This explains why SUMMARK remains effective for size substitution, family substitution, and combined substitution across domains.

Why General can be harder. We see a few weaker cases on General dataset (Table 8); for example, Qwen-7B→Qwen-1.5B has TPR@1% = 0.8740. General summaries are more conversational and stylistically diverse, with fewer repeated templates than Code, which increases lexical entropy and reduces the effective per-token evidence that our count-based detector can accumulate. Moreover, General summaries rely heavily on high-frequency function words and generic phrasing; when ubiquitous tokens fall into a greenlist, non-watermarked summaries can match portions of the greenlist by chance, narrowing separation specifically at very low-FPR thresholds. Importantly, separability remains strong overall: this same setting retains a high AUC (0.9954), indicating the watermark signal is present but stricter operating points (e.g., FPR$= 1\%$) can incur more misses.

Why Code can make baselines look strong. On the Code dataset (Table 9), summaries are more structured and often follow recurring templates, e.g., step-by-step debugging narratives, standard error-handling language, and repeated references to functions/variables. This regularity can make strong encoders (notably T5) appear competitive in some substitution pairs by exploiting surface-level cues. However, this advantage is largely *domain-specific*: the same classifiers are much less consistent on Math and General, reinforcing that their performance often reflects exploitable formatting/style regularities rather than a verifiable, identity-bound signal. By contrast, SUMMARK remains reliable across domains and substitution types because its evidence is keyed and identity-specific rather than stylistic.

Comparing with Identity prompting (IDP). Prompt the model to disclose its identity (e.g., "Who are you?", "Who developed you?") and string-match against known identifiers (family, developer), following the simple classifier setup in Huang et al. [13] We compare our method SUMMARK with IDP on Math Dataset. Table 10 shows that IDP performs poorly on substitution detection across most

Table 11: Attribution performance on **Math**. Higher is better.

Method	ACC	Macro-F1
BERT	0.9386	0.9383
GPT-2	0.7121	0.6981
T5	0.9128	0.9122
IDP	0.7500	0.6670
SUMMARK	**0.9958**	**0.9957**

Table 12: Identity-Substitution detection on Math. SynthID and Gumbel Max are applied directly to the task-facing answer, while SUMMARK applies the watermark only to the summary channel. We report TPR@FPR=1%, ACC@FPR=1%, and AUC.

	SynthID			Gumbel Max			SUMMARK		
Setting	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**
Q7→ Q3	0.0000	0.4950	0.5350	0.0500	0.5200	0.6041	**1.0000**	**0.9950**	**1.0000**
Q7→ Q1.5	0.0350	0.5125	0.5623	0.0300	0.5100	0.5463	**0.9960**	**0.9930**	**0.9974**
Q3→ Q1.5	0.0700	0.5300	0.5235	0.0200	0.5050	0.4112	**1.0000**	**0.9950**	**0.9998**
Q3→ L3	0.0250	0.5075	0.5216	0.0450	0.5175	0.4143	**1.0000**	**0.9950**	**0.9997**
L3→ Q3	0.0000	0.4950	0.4727	0.0000	0.4950	0.4770	**1.0000**	**0.9950**	**1.0000**
Q7→ L3	0.0750	0.5325	0.5737	0.0150	0.5025	0.4531	**1.0000**	**0.9950**	**1.0000**
L3→ Q1.5	0.0250	0.5075	0.4680	0.0100	0.5000	0.4352	**1.0000**	**0.9950**	**1.0000**

settings on the Math dataset. In many cases, IDP yields a TPR@1% of 0 with ACC@1% fixed at 0.5 and AUC at 0.5, which is equivalent to random guessing. This indicates that identity prompting fails to distinguish between the claimed model and the serving model under strict false positive constraints. Only in a few settings does IDP reach perfect scores, but these results are inconsistent and highly setting-dependent. Overall, the results suggest that IDP lacks robustness and cannot reliably detect model substitution in this task.

And we also conduct attribution detection for IDP. Table 11 reports attribution performance on the Math dataset. The IDP baseline achieves relatively low performance on the Math dataset, with an accuracy of 0.75 and a Macro-F1 of 0.667. This gap compared to stronger attribution methods indicates that identity prompting combined with simple string matching is not sufficient for robust attribution.

Comparing with answer-level watermarking methods. Table 12 shows that applying distortion-free watermarks directly to task-facing answers is insufficient for identity substitution detection on Math. Both SynthID and Gumbel Max achieve near-zero TPR@FPR=1% in most settings, with accuracies close to random guessing and AUCs mostly near chance. In contrast, SUMMARK achieves 0.996–1.000 TPR@1%, 0.993–0.995 ACC@1%, and consistently high AUC across all substitution pairs. This gap indicates that the task-facing answer is a weak carrier for identity auditing under strict false-alarm constraints.

Substitution results on TPR@FPR = 5%. Table 13 compares our method with text-only baselines under a relaxed auditing constraint of FPR = 5% on Math dataset. Our approach achieves perfect or near-perfect detection across all substitution settings, consistently reaching TPR near to 1. In contrast, the baselines improve or remain unchanged under the relaxed threshold, but remain less consistent across model pairs. BERT improves under the relaxed threshold but remains unstable, particularly for cross-family substitutions. GPT-2 performs poorly in most settings, indicating limited sensitivity to model-identity cues. T5 shows relatively strong performance on some size substitutions but degrades noticeably on harder cross-family cases. These results demonstrate that even when allowing higher false positive rates, our method provides significantly stronger and more reliable substitution detection than text-based classifiers, highlighting its robustness and scalability under practical auditing conditions.

Substitution Detection under Quantized Serving. Table 18 evaluates whether substitution detection remains reliable when the served substitute model is run at different numerical precisions. Across FP16, INT8, and INT4, SUMMARK maintains consistently high performance: TPR@1% remains

Table 13: Identity-Substitution detection performance on **Math** dataset. We report TPR@FPR=1% and TPR@FPR=5%. Best is bold.

	BERT		GPT-2		T5		SUMMARK	
Setting	**TPR@1%**	**TPR@5%**	**TPR@1%**	**TPR@5%**	**TPR@1%**	**TPR@5%**	**TPR@1%**	**TPR@5%**
Qwen-7B→Qwen-3B	0.1232	0.5322	0.0000	0.0351	0.6359	0.8095	**1.0000**	**1.0000**
Qwen-7B→Qwen-1.5B	0.9412	0.9720	0.2105	0.2807	0.9804	0.9804	**0.9960**	**0.9970**
Qwen-3B→Qwen-1.5B	0.9720	0.9720	0.2807	0.2807	0.9804	0.9804	**1.0000**	**1.0000**
Qwen-3B→LLaMA-3B	0.9076	0.9692	0.6842	0.8070	0.7003	0.7199	**1.0000**	**1.0000**
LLaMA-3B→Qwen-3B	0.8431	0.9160	0.3684	0.7368	0.8627	0.8627	**1.0000**	**1.0000**
Qwen-7B→LLaMA-3B	0.6050	0.9160	0.2281	0.4737	0.9272	0.9888	**1.0000**	**1.0000**
LLaMA-3B→Qwen-1.5B	0.9860	0.9860	0.5614	0.9474	0.9860	0.9860	**1.0000**	**1.0000**

between 0.9860 and 1.0000, ACC@1% stays around 0.9880–0.9950, and AUC is close to 1.0 in all settings.

The results show that quantizing the substituted model does not weaken the identity-substitution signal. In most cases, INT8 and INT4 perform comparably to FP16, and sometimes even slightly better. This suggests that SUMMARK's detector is primarily driven by the mismatch between the claimed identity's summary-channel footprint and the served model's footprint, rather than by fine-grained numerical precision effects.

Expanded enrolled universe. To study scalability, we enlarge the enrolled universe to 12 identities from four families: Qwen (0.5B, 1.5B, 3B, 7B, 14B, 32B), LLaMA (3B, 7B), Mistral (Ministral-3B, Mistral-7B), and Gemma (2B, 7B). All methods use 1,000 training examples per identity. These results are reported separately from the primary benchmark in Section 4.1. For compactness, we report TPR@1% and AUC in the scaling experiments.

Identity and family scaling. The nested identity sets start with Qwen-1.5B/3B/7B and LLaMA-3B, then successively add Mistral-3B/7B, LLaMA-7B and Qwen-0.5B, Gemma-2B/7B, and Qwen-14B/32B, yielding 4/6/8/10/12 identities. With these sets (Table 14), SUMMARK's macro TPR@1% drops from 0.9986 at 4 identities to a minimum of 0.7462 at 8, then reaches 0.8187 at 12, still above T5 (0.7518) and GPT-2 (0.7188). The advantage is not uniform: T5 leads at 6 identities, and the classifiers attain higher AUC at larger counts. For family scaling, we select Qwen-3B/7B, LLaMA-3B/7B, Ministral-3B and Mistral-7B, and Gemma-2B/7B as the two identities for each respective family. Holding these selections fixed and averaging over all family subsets of each size (Table 15), SUMMARK leads in TPR@1% at every family count but trails in AUC. In other words, it is stronger at the calibrated low-FPR operating point than in global ranking.

Table 14: Identity-count scaling on **Math** (nested identity sets, 1,000 training examples per identity). Best is bold.

	GPT-2		T5		SUMMARK	
#Identities	**TPR@1%**	**AUC**	**TPR@1%**	**AUC**	**TPR@1%**	**AUC**
4	0.6250	0.9735	0.8279	0.9774	**0.9986**	**0.9987**
6	0.6213	0.9807	**0.8089**	**0.9819**	0.7811	0.9397
8	0.5900	**0.9792**	0.7360	0.9766	**0.7462**	0.9105
10	0.7015	**0.9826**	0.7340	0.9765	**0.7919**	0.9339
12	0.7188	**0.9833**	0.7518	0.9776	**0.8187**	0.9443

Table 15: Family-count scaling on **Math**. Each family contributes two identities; results average over all family subsets, so the 4-family point uses 8 identities. Best is bold.

	GPT-2		T5		SUMMARK	
#Families	**TPR@1%**	**AUC**	**TPR@1%**	**AUC**	**TPR@1%**	**AUC**
1	0.5788	**0.9646**	0.5825	0.9382	**0.7850**	0.9527
2	0.6344	**0.9769**	0.6702	0.9626	**0.8251**	0.9603
3	0.6501	**0.9795**	0.6487	0.9670	**0.8121**	0.9596
4	0.6977	**0.9836**	0.7061	0.9720	**0.7935**	0.9543

Larger scale gaps. We test substitutions in which the served model is 4.7–64.0× smaller than the claimed one (Table 16). SUMMARK's macro TPR@1% is 0.9250 within family and 0.9367 across families, compared with 0.6942/0.8467 for T5 and 0.6467/0.8333 for GPT-2. Detection depends on the specific model pair and does not improve monotonically with the size gap. Within-family substitutions from Qwen-32B achieve TPR@1% of 0.9900–1.0000, but Qwen-14B→Qwen-0.5B (0.69) and Qwen-14B→Ministral-3B (0.74) are weaker, and T5 is perfect on both Ministral-3B directions. These results use public checkpoints and do not establish generalization to proprietary serving stacks.

Table 16: Identity-Substitution detection with large parameter gaps on **Math** (1,000 training examples per identity). Red. is the claimed-to-served parameter ratio; macro averages are over the six pairs in each panel. Best is bold.

		GPT-2		T5		SUMMARK	
Setting	**Red.**	**TPR@1%**	**AUC**	**TPR@1%**	**AUC**	**TPR@1%**	**AUC**
Within family							
Qwen-32B→Qwen-0.5B	64.0×	0.7800	0.9866	0.8850	0.9924	**1.0000**	**0.9999**
Qwen-32B→Qwen-1.5B	21.3×	0.8250	0.9891	0.9700	0.9986	**1.0000**	**1.0000**
Qwen-32B→Qwen-3B	10.7×	0.8600	0.9912	0.9350	0.9966	**0.9900**	**0.9997**
Qwen-14B→Qwen-0.5B	28.0×	0.3150	**0.9374**	0.3450	0.8949	**0.6900**	0.9278
Qwen-14B→Qwen-1.5B	9.3×	0.8300	0.9926	**0.9000**	**0.9971**	0.8700	0.9667
Qwen-14B→Qwen-3B	4.7×	0.2700	0.9641	0.1300	0.9247	**1.0000**	**0.9988**
Macro average	–	0.6467	0.9768	0.6942	0.9674	**0.9250**	**0.9822**
Cross family							
Qwen-32B→LLaMA-3B	10.7×	0.8850	0.9922	0.9900	0.9993	**1.0000**	**1.0000**
Qwen-32B→Ministral-3B	10.7×	0.9800	0.9952	**1.0000**	**1.0000**	0.8800	0.9745
Qwen-32B→Gemma-2B	16.0×	0.9350	0.9943	0.9800	0.9979	**1.0000**	**1.0000**
Qwen-14B→LLaMA-3B	4.7×	0.3150	0.9724	0.3050	0.9639	**1.0000**	**0.9991**
Qwen-14B→Ministral-3B	4.7×	0.9750	0.9985	**1.0000**	**1.0000**	0.7400	0.9532
Qwen-14B→Gemma-2B	7.0×	0.9100	0.9975	0.8050	0.9883	**1.0000**	**1.0000**
Macro average	–	0.8333	**0.9917**	0.8467	0.9916	**0.9367**	0.9878

Attribution. With all 12 identities enrolled, SUMMARK's exact-identity accuracy is 87.83% (GPT-2: 82.50%; T5: 77.38%; chance: 8.33%). Mapping the same predictions to families gives 93.17%; this is not a separate four-class experiment.

Held-out substitutes. An auditor cannot enumerate every possible substitute, so we exclude test substitutes from the L_{sep} negative pool (Table 17). In the held-out-family setting, LLaMA-3B is excluded for the Qwen-3B/7B adapters. In the held-out-scale setting, Qwen-1.5B is excluded for the other adapters. Aggregate TPR@1% is 0.9990 and 0.9973, respectively. This supports rejecting unseen substitutes under the claimed identity's keyed test, not attributing them. Precision variants are not used as negatives in L_{sep}.

Table 17: Claim rejection on **Math** for substitutes held out of the L_{sep} negative-identity pool. The task is claim rejection, not open-set attribution.

		SUMMARK	
Held-out axis	**Setting**	**TPR@1%**	**AUC**
Family	Qwen-3B→LLaMA-3B	0.9990	0.9991
	Qwen-7B→LLaMA-3B	0.9990	1.0000
	Aggregate	0.9990	0.9995
Scale	Qwen-3B→Qwen-1.5B	1.0000	0.9986
	Qwen-7B→Qwen-1.5B	0.9920	0.9984
	LLaMA-3B→Qwen-1.5B	1.0000	1.0000
	Aggregate	0.9973	0.9990

Table 18: Substitution detection results on **Math** under different serving precisions. For each substitution pair, the provider serves the substituted model at FP16, INT8, or INT4 precision while claiming the target identity.

	FP16			INT8			INT4		
Setting	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**
Q7→Q3	**1.0000**	**0.9950**	**1.0000**	**1.0000**	**0.9950**	0.9985	**1.0000**	**0.9950**	0.9995
Q7→Q1.5	0.9960	0.9930	0.9974	**1.0000**	**0.9950**	**0.9995**	0.9920	0.9910	0.9943
Q3→Q1.5	**1.0000**	**0.9950**	**0.9998**	**1.0000**	**0.9950**	0.9985	0.9970	0.9935	0.9945
Q3→L3	**1.0000**	**0.9950**	**0.9997**	**1.0000**	**0.9950**	0.9995	0.9970	0.9935	0.9960
L3→Q3	**1.0000**	**0.9950**	**1.0000**	**1.0000**	**0.9950**	0.9995	0.9960	0.9930	0.9983
Q7→L3	**1.0000**	**0.9950**	**1.0000**	**1.0000**	**0.9950**	0.9995	0.9980	0.9940	0.9994
L3→Q1.5	**1.0000**	**0.9950**	**1.0000**	0.9970	0.9935	0.9982	0.9860	0.9880	0.9975

Table 19: Ablation on loss components on the Code dataset. **SFT** uses only L_{sft}, **+Sep** adds L_{sep}, **+WM** adds L_{wm}, and **+Both** adds both losses.

	TPR@1%				AUC			
Setting	**SFT**	**+Sep**	**+WM**	**+Both**	**SFT**	**+Sep**	**+WM**	**+Both**
Q7→Q3	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Q7→Q1.5	0.9300	0.9600	**1.0000**	**1.0000**	0.9860	0.9930	**1.0000**	**1.0000**
Q3→Q1.5	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Q3→L3	**1.0000**	**1.0000**	0.9800	**1.0000**	**1.0000**	0.9999	0.9993	**1.0000**
L3→Q3	**1.0000**	**1.0000**	0.9700	**1.0000**	**1.0000**	0.9999	0.9993	**1.0000**
Q7→L3	**1.0000**	**1.0000**	0.9900	**1.0000**	0.9999	**1.0000**	0.9953	**1.0000**
L3→Q1.5	0.9700	0.9800	0.9600	**1.0000**	0.9990	0.9989	0.9992	**1.0000**

Table 20: Comparison of no-watermark (Plain), greenlist watermark (Greenlist), and our method (SUMMARK) on the **Math** dataset.

	TPR@1%			AUC		
Setting	**Plain**	**Greenlist**	**SUMMARK**	**Plain**	**Greenlist**	**SUMMARK**
Q7→Q3	0.0100	0.0000	**1.0000**	0.1717	0.1180	**1.0000**
Q7→Q1.5	0.0000	0.0000	**0.9960**	0.1447	0.1080	**0.9974**
Q3→Q1.5	0.0600	0.0310	**1.0000**	0.5226	0.3560	**0.9998**
Q3→L3	0.0100	0.0000	**1.0000**	0.5052	0.1250	**0.9997**
L3→Q3	0.0400	0.0000	**1.0000**	0.5160	0.5170	**1.0000**
Q7→L3	0.0200	0.0000	**1.0000**	0.3352	0.1740	**1.0000**
L3→Q1.5	0.0400	0.0620	**1.0000**	0.4104	0.5890	**1.0000**

E.2 Robustness Analysis

Lexical watermark removal. Table 21 evaluates robustness on Code under two downstream perturbations: task-specific summarization fine-tuning and lexical watermark removal. The original SUMMARK detector achieves near-perfect performance across all substitution pairs. Composing a downstream summarization LoRA with the watermark adapter preserves strong detection in most settings, but reduces TPR@1% in harder cross-family cases, especially Q3→L3 and L3→Q3. This suggests that downstream adaptation can shift the summary token distribution and narrow the low-FPR detection margin. In contrast, the copy-paste watermark removal attack has little effect: TPR@1% remains between 0.997 and 1.000 with AUC ranging from 0.9973 to 0.9995. This indicates that SUMMARK 's signal is distributed across summary-level token statistics rather than concentrated in easily removable local spans.

Watermark spoofing. Table 22 evaluates two text-only spoofing attacks where the adversary observes watermarked summaries from the claimed identity but has no access to the auditor's secret key, carrier sets, or adapter weights. In *distilled adapter spoofing*, the adversary collects watermarked prompt–summary pairs from the claimed identity and trains a local spoof adapter on the substituted model to imitate the claimed identity's summary distribution. In *watermark-stealing spoofing*, the adversary estimates token-level carrier statistics from observed watermarked summaries and biases decoding toward tokens that appear likely to belong to the claimed identity's carrier set.

Distilled adapter spoofing is mostly contained: ASR@1% stays below 0.05 for four of seven transfer directions, with TPR@1% between 0.9570 and 0.9980. This suggests that spoofing requires approximating the claimed identity's keyed statistical footprint, not merely its summary style. However, some directions are substantially harder, especially L3→Q3 (ASR@1% = 0.6290), Q7→L3 (0.2500), and L3→Q1.5 (0.2110).

Watermark-stealing spoofing is less effective: ASR@1% is zero in most settings, except Q7→Q3 (0.2160) and Q7→Q1.5 (0.0760). Overall, SUMMARK is robust to text-only watermark stealing and to distilled spoofing in most directions, but adaptive distilled spoofing remains a limitation in specific transfer directions.

Summary length sensitivity. Table 23 evaluates how detection changes when released summaries are truncated to different prefix lengths. Overall, SUMMARK remains effective even with short

Table 21: Robustness on Code under downstream fine-tuning and lexical watermark removal. For fine-tuning, we compose a task-specific summarization LoRA with the watermark adapter at inference. For watermark removal, we apply a copy-paste attack that replaces contiguous spans in the summary.

	Original			Model Finetuning			Watermark Removal		
Setting	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**
Q7→Q3	**1.0000**	**0.9950**	**1.0000**	0.9630	0.9765	0.9950	**1.0000**	**0.9950**	0.9995
Q7→Q1.5	**1.0000**	**0.9950**	**1.0000**	0.9770	0.9835	0.9978	0.9970	0.9935	0.9973
Q3→Q1.5	**1.0000**	**0.9950**	**1.0000**	0.9780	0.9840	0.9975	**1.0000**	**0.9950**	0.9994
Q3→L3	**1.0000**	**0.9950**	**1.0000**	0.4480	0.7190	0.9553	**1.0000**	**0.9950**	0.9994
L3→Q3	**1.0000**	**0.9950**	**1.0000**	0.7160	0.8530	0.9832	**1.0000**	**0.9950**	0.9990
Q7→L3	**1.0000**	**0.9950**	**1.0000**	0.9840	0.9870	0.9979	**1.0000**	**0.9950**	0.9995
L3→Q1.5	**1.0000**	**0.9950**	**1.0000**	0.9640	0.9770	0.9956	0.9990	0.9945	0.9990

Table 22: Robustness under spoofing attacks on Math. We evaluate two text-only spoofing strategies: distilled adapter spoofing and watermark-stealing spoofing. For both attacks, lower ASR@1% and higher TPR@1%/AUC indicate stronger audit robustness.

	Distilled Adapter Spoofing			Watermark Stealing Spoofing		
Setting	**TPR@1%**	**ASR@1%**	**AUC**	**TPR@1%**	**ASR@1%**	**AUC**
Q7→Q3	0.9980	0.0020	0.9999	0.7840	0.2160	0.9930
Q7→Q1.5	0.9830	0.0170	0.9939	0.9240	0.0760	0.9960
Q3→Q1.5	0.9960	0.0040	0.9983	1.0000	0.0000	1.0000
Q3→L3	0.9570	0.0430	0.9938	1.0000	0.0000	1.0000
L3→Q3	0.3710	0.6290	0.9474	1.0000	0.0000	1.0000
Q7→L3	0.7500	0.2500	0.9706	1.0000	0.0000	1.0000
L3→Q1.5	0.7890	0.2110	0.9641	1.0000	0.0000	1.0000

summaries: at 128 tokens, six of seven settings already achieve TPR@1% at least 0.925, and several settings remain perfect across all lengths. This suggests that the identity footprint is not dependent on unusually long summaries.

The main exception is Q3→L3, where TPR@1% stays around 0.65–0.70 despite high AUC. This indicates that the two distributions remain separable in aggregate, but the margin is not large enough to support near-perfect detection at the strict 1% FPR threshold. Performance is also not strictly monotonic with length, since longer prefixes may introduce additional natural lexical variation rather than only more watermark evidence. Overall, the results show that summary length affects the low-FPR margin, but short summaries are sufficient for reliable detection in most substitution directions.

Summary post-processing. Table 24 reports per-pair detection results on Math under sentence shuffling, random deletion, word truncation, and bilingual rewriting. Across the three bounded surface transformations, the average TPR@1% over all seven substitution pairs and three transformations is 0.9992, with a minimum of 0.9950. Bilingual rewriting reduces the average TPR@1% to 0.7900 and the minimum to 0.5740. These results indicate that detection remains stable under the evaluated bounded edits, while more extensive rewriting can weaken the watermark signal.

Cross-domain generalization. We train each adapter on Math prompts only, freeze it together with the Math-calibrated thresholds, and evaluate the same seven substitution pairs on Code and General without retraining or recalibration (Table 25). Macro TPR remains above 0.998 on both target domains, with observed macro FPR of 0.0061 on Code and 0.0123 on General. Per-pair FPR ranges from 0.001 to 0.014 on Code and from 0.002 to 0.025 on General, so calibration drift is pair-dependent and some General directions exceed the nominal 1% target. These results cover two task-domain shifts only, not arbitrary shifts or an adaptive provider that alters the generation process.

Training data poisoning. Trusted onboarding assumes an uncorrupted adapter-training corpus. We stress-test this assumption on Qwen-7B→Qwen-3B by flipping the carrier identity label between the two models for a fraction of the 1,000 training examples. Prompts and teacher summaries are unchanged, so the L_{wm} and L_{sep} targets are corrupted while L_{sft} is not (Table 26). Detection is nearly unaffected at 5% poisoning, degrades at 15%–25%, and TPR@1% drops to zero from 50%

Table 23: Length sensitivity of substitution detection on Code. We truncate each released summary to the first 128, 256, 384, or 512 tokens, and report TPR@FPR=1%, ACC@FPR=1%, and AUC. Setting A→B denotes serving B while claiming A.

	128 tokens			256 tokens			384 tokens			512 tokens		
Setting	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**	**TPR@1%**	**ACC@1%**	**AUC**
Q7→Q3	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Q7→Q1.5	0.9250	0.9625	0.9944	**1.0000**	**1.0000**	**1.0000**	0.9750	0.9875	0.9969	0.9750	0.9875	0.9994
Q3→Q1.5	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.7750	0.8875	0.9944	0.9750	0.9875	0.9981
Q3→L3	**0.7000**	**0.8500**	0.9825	**0.7000**	**0.8500**	0.9788	0.6750	0.8375	**0.9856**	0.6500	0.8250	0.9756
L3→Q3	0.9750	0.9875	0.9938	0.9750	0.9875	0.9925	**1.0000**	**1.0000**	**1.0000**	0.9750	0.9875	0.9994
Q7→L3	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
L3→Q1.5	**0.9750**	**0.9875**	**0.9963**	**0.9750**	**0.9875**	0.9931	0.8500	0.9250	0.9644	0.9000	0.9500	0.9656

Table 24: Robustness of watermark detection under summary post-processing and rewriting on Math. We report three bounded surface transformations (sentence shuffle, random deletion, and word truncation), and one stronger bilingual rewriting stress test.

	Original		Sentence Shuffle		Random Deletion		Truncate Words		Bilingual Rewrite	
Setting	**TPR@1%**	**AUC**	**TPR@1%**	**AUC**	**TPR@1%**	**AUC**	**TPR@1%**	**AUC**	**TPR@1%**	**AUC**
Q7→Q3	**1.0000**	**1.0000**	**1.0000**	0.9995	**1.0000**	0.9995	**1.0000**	0.9995	0.9760	0.9963
Q7→Q1.5	**0.9960**	0.9974	**0.9960**	0.9969	0.9950	0.9969	0.9950	**0.9981**	0.9010	0.9931
Q3→Q1.5	**1.0000**	**0.9998**	**1.0000**	0.9994	**1.0000**	0.9993	**1.0000**	0.9994	0.8220	0.9840
Q3→L3	**1.0000**	**0.9997**	**1.0000**	0.9993	**1.0000**	0.9992	**1.0000**	0.9994	0.6570	0.9652
L3→Q3	**1.0000**	**1.0000**	**1.0000**	0.9990	**1.0000**	0.9990	**1.0000**	0.9990	0.6390	0.9646
Q7→L3	**1.0000**	**1.0000**	**1.0000**	0.9995	**1.0000**	0.9995	**1.0000**	0.9995	0.9610	0.9961
L3→Q1.5	**1.0000**	**1.0000**	**1.0000**	0.9990	0.9990	0.9989	0.9990	0.9989	0.5740	0.9388

Table 25: Cross-domain transfer from **Math** to **Code** and **General**. Adapters and detection thresholds are trained and calibrated on Math and kept fixed; observed FPR is measured on target-domain data. Metrics are macro-averaged over seven substitution pairs.

	SUMMARK			
Train→test	**TPR**	**Observed FPR**	**Balanced ACC**	**AUC**
Math→Math	0.9994	0.0100	0.9947	0.9996
Math→Code	0.9993	0.0061	0.9966	0.9991
Math→General	0.9989	0.0123	0.9933	0.9990

onward. At 50%, AUC is still 0.8767: some score ranking survives, but detection at the calibrated low-FPR threshold does not. This marks a training-integrity boundary rather than a defense against poisoning; authenticated data provenance lies outside the verifier studied here.

Table 26: Carrier-label poisoning on Qwen-7B→Qwen-3B (**Math**, 1,000 training examples per rate). Prompts and teacher summaries are unchanged; only the L_{wm} and L_{sep} carrier targets are corrupted. At 100%, labels are fully swapped and the score ordering is inverted.

	SUMMARK		
Poison rate	**TPR@1%**	**ACC@1%**	**AUC**
0%	1.0000	0.9950	1.0000
5%	0.9900	0.9925	0.9998
15%	0.7950	0.8925	0.9761
25%	0.6900	0.8400	0.9602
50%	0.0000	0.4950	0.8767
100%	0.0000	0.4950	0.0032

E.3 Ablation Studies

Ablation 1: Do we need identity-specific watermarking? Table 20 compares three variants on Math:**Plain** (no watermark), a **Greenlist** watermark that uses a *single shared* greenlist across identities, and **SUMMARK**. Under a strict auditing regime (TPR at FPR$= 1\%$), both Plain and Greenlist fail

across all substitution settings, even when AUC appears moderate. This highlights a core requirement of model *identity* auditing: a global greenlist can create a detectable bias in aggregate, but it does not provide *separable, identity-specific* evidence needed to reliably distinguish a claimed identity from closely related alternatives at low false-positive rates.

Why Greenlist fails at low FPR. Math summaries often have constrained lexical choices driven by the solution structure, leaving limited room for a weak global token bias to accumulate. As a result, non-watermarked summaries can match a shared greenlist by chance, and the separation becomes insufficient when the threshold is set to control false alarms. By contrast, SUMMARK explicitly allocates *identity-specific* carrier sets and trains adapters to concentrate additional probability mass on the correct bucket, which yields a predictable shift in summary token statistics that remains separable across identities.

Ablation 2: Training objective components. We next isolate the role of the detectability and separability components in our training objective by ablating L_{wm} (green-mass regularization) and L_{sep} (margin-based separation across identities). As shown in Table 19 and Figure 4 when varying weights, removing either term can reduce TPR at FPR$= 1\%$, even when AUC remains high. This reflects the difference between *overall separability* and *auditing at strict operating points*. Concretely, L_{wm} supplies the primary watermark signal by increasing usage of the correct identity bucket, while L_{sep} suppresses the most confusable competing identity and stabilizes decisions across substitution pairs. Combining both terms achieves consistently strong low-FPR detection, matching the design goal of robust identity attribution under closely related substitutions.

Ablation 3: Effect of precision-consistency regularizer. Table 28 shows that the precision-consistency regularizer L_{qs} is essential for detecting precision variants. Without L_{qs}, the detector is almost random: TPR@1% is near zero, ACC@1% stays around 0.50, and AUC is close to chance across all models. After adding L_{qs}, detection improves sharply for most identities. LLaMA-3B, Qwen-3B, and Qwen-7B all achieve near-perfect TPR@1% and AUC for both FP16→INT8 and FP16→INT4.

The main exception is Qwen-1.5B, where INT8 remains difficult and INT4 is only partially detectable. Still, L_{qs} substantially improves its AUC, especially for FP16→INT4. Overall, these results show that standard identity watermarking alone does not naturally separate FP16 and quantized variants; the quantization-separation loss is what creates a precision-sensitive audit signal.

Ablation 4: Adapter-matched text and watermarking baselines. The shared-Greenlist and generic watermark baselines above do not control for the use of an identity-specific adapter. We therefore add adapter-matched baselines on Math, using the same seven substitution pairs and 1,000 training examples per adapter. *T5 + SFT adapter* trains an SFT-only summary adapter on clean, unwatermarked reference summaries; T5 is then trained and evaluated on its outputs alone, without a key or carrier information. *Keyed Unigram/Greenlist adapters* assign each identity its own secret key and train an adapter to reproduce teacher summaries watermarked with that key. Both controls use an adapter per identity, and the keyed variants a separate key per identity, but neither uses SUMMARK's controlled-overlap carriers or hardest-negative separability objective.

As shown in Table 27, the strongest control, T5 + SFT adapter, reaches a mean TPR@1% of 0.8937 but drops to 0.6550 on its worst pair. The keyed watermark adapters perform far worse, while SUMMARK reaches 0.9980. Under these implementations and budget, the gain is therefore not explained by adapter-based generation or per-identity keys alone, although this does not rule out multi-key watermark attribution in general. This control differs from the SFT-only loss ablation in Table 19, which removes SUMMARK's own training terms.

E.4 Case Study

To complement the quantitative ablations and robustness tests above, we provide a qualitative visualization of what the identity watermark looks like in a single summary. Figure 3 highlights tokens from a fixed bucket to show how the correct adapter produces a consistently denser, more uniformly distributed footprint, while a substituted identity does not. This case study visualizes identity-specific watermarking at the token level under three settings: no watermark, Qwen-3B watermark, and Qwen-7B watermark. For visualization, all texts are colored using the same vocabulary bucket $G_{\text{Qwen-3B}}$, where tokens in this set are marked green. Without watermarking, green tokens appear only occasionally, consistent with natural generation. With the Qwen-3B watermark enabled, green

Table 27: Adapter-matched baselines on **Math** (1,000 training examples per identity). Mean and minimum are taken over seven substitution pairs. T5 + SFT adapter uses unwatermarked summaries; keyed Unigram/Greenlist adapters use a separate key and adapter per identity. Best is bold.

	TPR@1%		AUC	
Method	**Mean**	**Min**	**Mean**	**Min**
T5 + SFT adapter	0.8937	0.6550	0.9899	0.9619
Keyed Unigram adapter	0.3328	0.0020	0.7890	0.1340
Keyed Greenlist adapter	0.0163	0.0000	0.5319	0.4823
SUMMARK	**0.9980**	**0.9920**	**0.9981**	**0.9970**

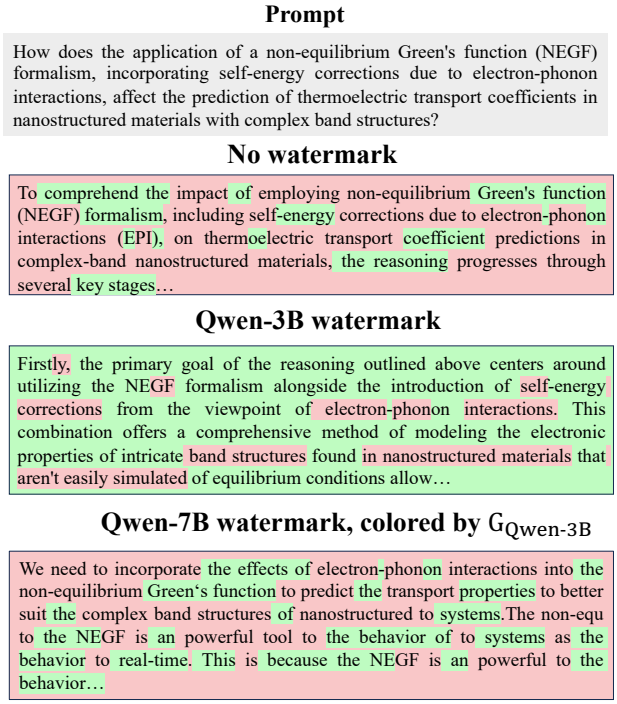

Figure 3: Visualizing identity footprints: tokens in $G_{\text{Qwen-3B}}$ are highlighted. The Qwen-3B adapter yields a denser, more uniform green-token pattern than no-watermark, while a substituted identity (Qwen-7B) does not reproduce this footprint when colored by $G_{\text{Qwen-3B}}$.

tokens become noticeably more frequent and more evenly spread throughout the summary, indicating a stable, detectable bias. We color Qwen-7B summaries by $G_{\text{Qwen-3B}}$ to simulate a substitution case (serving Qwen-7B while claiming Qwen-3B); the resulting summary does not exhibit the same green-token pattern, illustrating that watermark footprints are separable across model identities.

E.5 Hyperparameter Analysis

Substitution ratio. To ensure fair comparison, we fix the total number of test samples to 1,000 for each substitution ratio across all three datasets. As shown in Tables 29, and 30, detection performance consistently improves as the substitution ratio increases on Math and General. Even at low substitution ratios, TPR@FPR=1% is close to 1.0 for most substitution settings, and AUC remains around 0.99. At a substitution ratio of 0.5, performance is effectively saturated, with near-perfect TPR and consistently high AUC across all model pairs. We therefore adopt a ratio of 0.5 as the default setting in our experiments.

Loss design. We study the sensitivity of our method to the weights of the watermark loss λ_{wm} and the separability loss λ_{sep} across three types of substitution. We report TPR@FPR = 1%, with the ratio defined as $\lambda_{\text{sep}}/\lambda_{\text{wm}}$. Figure 4 shows TPR@1% for one Code example from each substitution

Table 28: Effect of the quantization-separation loss L_{qs} on within-identity precision-consistency detection. We compare watermark adapters trained with and without L_{qs} when the claimed configuration is FP16 and the served configuration is INT8 or INT4.

Model	Setting	w/o L_{qs}			w/ L_{qs}		
		TPR@1%	ACC@1%	AUC	TPR@1%	ACC@1%	AUC
LLaMA-3B	FP16→INT8	0.0150	0.5040	0.5034	**0.9990**	**0.9960**	**0.9990**
	FP16→INT4	0.0000	0.5000	0.4256	**0.9990**	**0.9995**	**0.9990**
Qwen-1.5B	FP16→INT8	0.0090	0.5000	0.4952	**0.1132**	**0.5521**	**0.7575**
	FP16→INT4	0.0181	0.5050	0.4875	**0.3417**	**0.6663**	**0.9209**
Qwen-3B	FP16→INT8	0.0020	0.5000	0.5298	**0.9860**	**0.9890**	**0.9973**
	FP16→INT4	0.0000	0.5000	0.3719	**0.9960**	**0.9945**	**0.9986**
Qwen-7B	FP16→INT8	0.0140	0.5030	0.4868	**0.9910**	**0.9920**	**0.9982**
	FP16→INT4	0.0140	0.5045	0.5611	**0.9990**	**0.9975**	**0.9998**

Table 29: Sensitivity to **Substitution Ratio** on **Math**. We report TPR@FPR=1%, ACC@FPR=1%, and AUC. Q7/Q3/Q1.5 = Qwen-7B/3B/1.5B, L3 = LLaMA-3B. Best is bold.

	5%			10%			15%			25%			50%			65%		
Setting	TPR	ACC	AUC	TPR	ACC	AUC	TPR	ACC	AUC	TPR	ACC	AUC	TPR	ACC	AUC	TPR	ACC	AUC
Q7→Q3	**1.0000**	0.9910	**0.9995**	**1.0000**	0.9910	0.9994	**1.0000**	0.9920	0.9994	**1.0000**	0.9930	0.9993	**1.0000**	0.9950	0.9990	**1.0000**	**0.9970**	0.9986
Q7→Q1.5	**1.0000**	0.9910	**0.9995**	**1.0000**	0.9910	0.9994	0.9933	0.9910	0.9931	0.9920	0.9910	0.9955	0.9940	0.9920	0.9953	0.9969	**0.9950**	0.9957
Q3→Q1.5	**1.0000**	0.9910	**0.9994**	**1.0000**	0.9910	0.9989	**1.0000**	0.9920	0.9990	**1.0000**	0.9930	0.9985	**1.0000**	0.9950	0.9989	**1.0000**	**0.9970**	0.9985
Q3→L3	**1.0000**	0.9910	**0.9994**	**1.0000**	0.9910	0.9992	**1.0000**	0.9920	0.9993	**1.0000**	0.9930	0.9992	**1.0000**	0.9950	0.9988	**1.0000**	**0.9970**	0.9940
L3→Q3	**1.0000**	0.9910	**0.9990**	**1.0000**	0.9910	0.9989	**1.0000**	0.9920	0.9988	**1.0000**	0.9930	0.9987	**1.0000**	0.9950	**0.9990**	**1.0000**	**0.9970**	0.9986
Q7→L3	**1.0000**	0.9910	**0.9995**	**1.0000**	0.9910	**0.9995**	**1.0000**	0.9920	0.9994	**1.0000**	0.9930	0.9993	**1.0000**	0.9950	0.9990	**1.0000**	**0.9970**	0.9986
L3→Q1.5	**1.0000**	0.9910	0.9989	**1.0000**	0.9910	0.9989	**1.0000**	0.9920	0.9988	**1.0000**	0.9930	**0.9993**	**1.0000**	0.9950	0.9990	**1.0000**	**0.9970**	0.9986

type, and Figure 6 reports AUC for the same pairs. Figures 7 and 8 report paired AUC and TPR@1% results for four representative General pairs, covering changes in size, family, and both, together with an additional parameter-sensitive case.

First, comparing watermark strengths of 0.2 and 0.5, we observe that a larger λ_{wm} can improve detection, although the benefit depends on the model pair and loss ratio. Second, many substitution settings achieve high performance at larger loss ratios. These trends support the usefulness of balancing the watermark and separability terms, but the response is not uniformly monotonic. In particular, General Qwen-7B→Qwen-1.5B remains sensitive, with TPR@1% of 0.57 at $\lambda_{\mathrm{wm}} = \lambda_{\mathrm{sep}} = 0.5$.

Watermark design. We study the sensitivity of substitution detection performance to the green ratio ρ and the overlap τ. In Figure 5, the first row fixes $\rho = 0.3$ and varies τ, while the second row fixes $\tau = 0.3$ and varies ρ. Most tested configurations maintain high detection, although particular model pairs are sensitive: LLaMA-3B→Qwen-1.5B reaches TPR@1% of 0.11 at $\rho = 0.35$, and LLaMA-3B→Qwen-3B reaches 0.816 at $\tau = 0.7$. More importantly, as discussed in Equation 6, the feasible combinations of ρ and τ are constrained by model identity capacity. Choosing overly small overlap limits the number of supported identities, while a small green ratio may restrict the effective vocabulary during summary generation, creating a potential trade-off with summary quality. We therefore use $\rho = \tau = 0.3$ as a practical default motivated by these design considerations.

Table 30: Sensitivity to **Substitution Ratio** on **General**. We report TPR@FPR=1%, ACC@FPR=1%, and AUC. Q7/Q3/Q1.5 = Qwen-7B/3B/1.5B, L3 = LLaMA-3B. Best is bold.

	5%			10%			15%			25%			50%			65%		
Setting	**TPR**	**ACC**	**AUC**	**TPR**	**ACC**	**AUC**	**TPR**	**ACC**	**AUC**	**TPR**	**ACC**	**AUC**	**TPR**	**ACC**	**AUC**	**TPR**	**ACC**	**AUC**
Q7→Q3	**1.0000**	0.9910	0.9999	**1.0000**	0.9910	0.9999	**1.0000**	0.9920	0.9999	**1.0000**	0.9930	0.9999	**1.0000**	0.9950	**1.0000**	**1.0000**	**0.9970**	0.9999
Q7→Q1.5	0.8200	0.9820	0.9958	0.8400	0.9750	0.9931	0.8800	0.9740	0.9951	0.9840	**0.9890**	0.9965	0.9860	0.9880	**0.9971**	**0.9877**	**0.9890**	0.9969
Q3→Q1.5	**1.0000**	0.9910	**1.0000**	**1.0000**	0.9910	**1.0000**	**1.0000**	0.9920	**1.0000**	**1.0000**	0.9930	**1.0000**	0.9980	0.9940	**1.0000**	0.9985	**0.9960**	**1.0000**
Q3→L3	**1.0000**	0.9910	**1.0000**	**1.0000**	0.9910	0.9999	**1.0000**	0.9920	0.9999	**1.0000**	0.9930	0.9999	**1.0000**	0.9950	0.9999	**1.0000**	**0.9970**	0.9999
L3→Q3	**1.0000**	0.9910	**1.0000**	**1.0000**	0.9910	**1.0000**	**1.0000**	0.9920	**1.0000**	**1.0000**	0.9930	**1.0000**	**1.0000**	0.9950	**1.0000**	**1.0000**	**0.9970**	**1.0000**
Q7→L3	**1.0000**	0.9910	**0.9996**	**1.0000**	0.9910	**0.9996**	**1.0000**	**0.9920**	**0.9996**	0.9880	0.9900	0.9995	0.9840	0.9870	0.9990	0.9862	0.9880	0.9986
L3→Q1.5	**1.0000**	**0.9910**	**1.0000**	0.9700	0.9880	0.9990	0.9733	0.9880	0.9990	0.9640	0.9840	0.9987	0.9580	0.9740	0.9990	0.9600	0.9710	0.9990

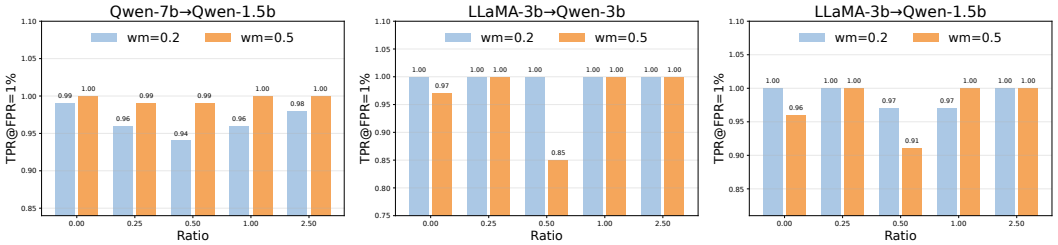

Figure 4: Effect of training objective components λ_{sep} and λ_{wm} on substitution performance on **Code** dataset. Ratio is defined as $\lambda_{\text{sep}}/\lambda_{\text{wm}}$. We report TPR@FPR = 1% as the score.

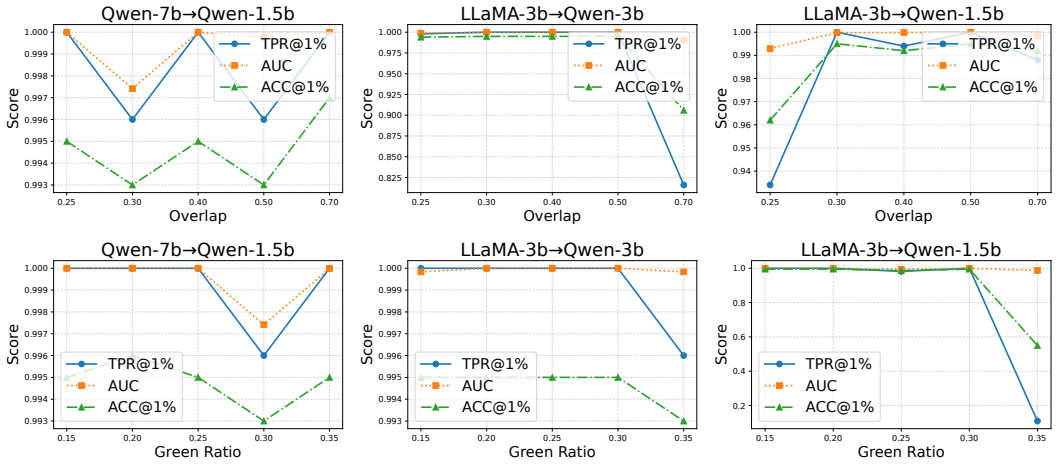

Figure 5: Effect of green ratio ρ and overlap τ on substitution performance on **Math** dataset.

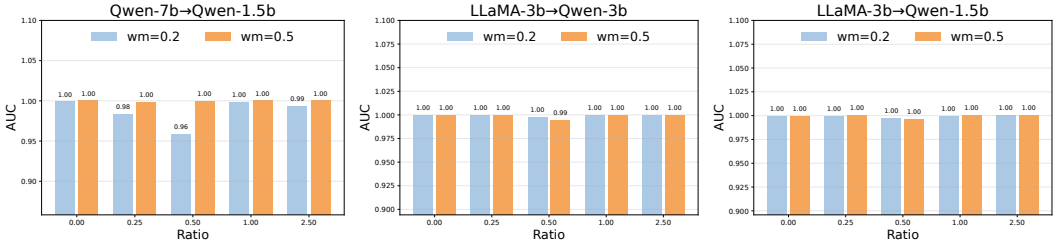

Figure 6: Effect of training objective components λ_{sep} and λ_{wm} on substitution performance on **Code** dataset. Ratio is defined as $\lambda_{\text{sep}}/\lambda_{\text{wm}}$. We report AUC as the score.

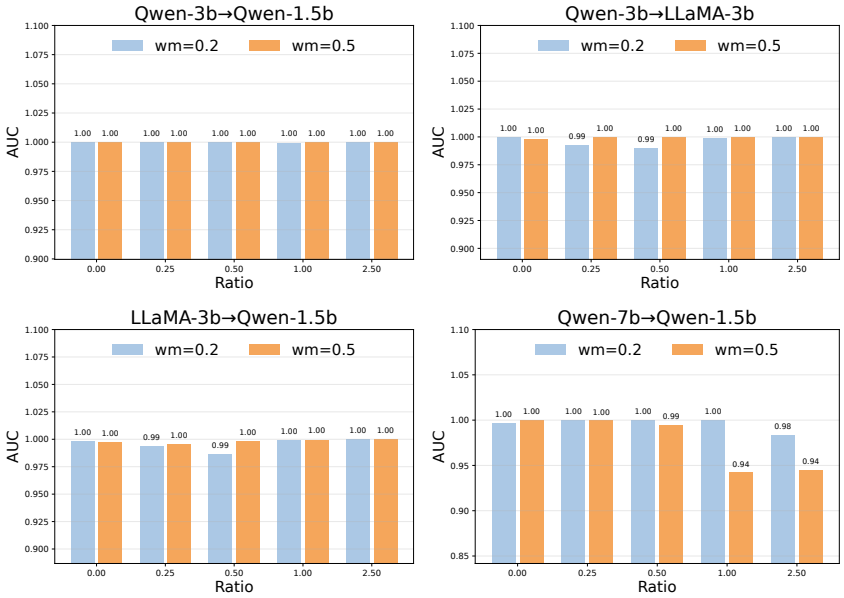

Figure 7: Effect of training objective components λ_{sep} and λ_{wm} on substitution performance on **General** dataset. Ratio is defined as $\lambda_{\text{sep}}/\lambda_{\text{wm}}$. We report AUC as the score.

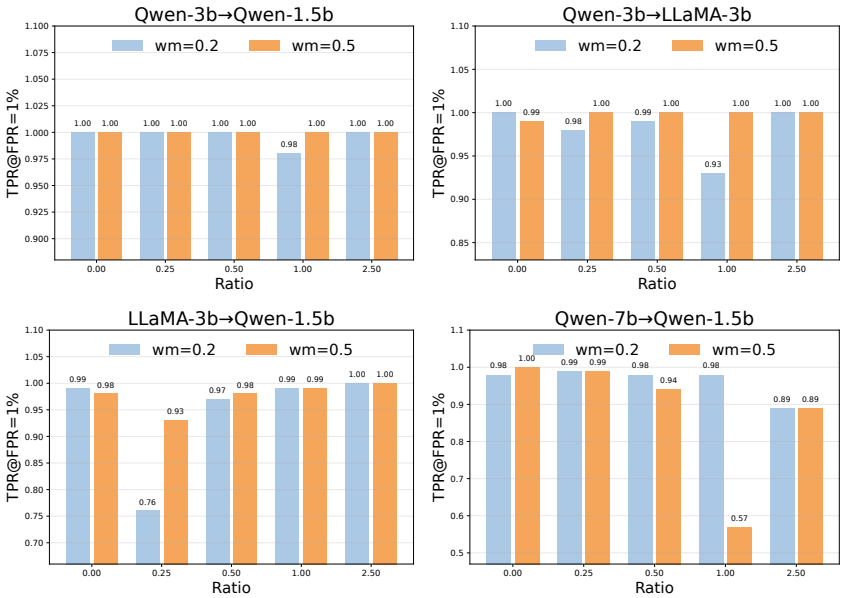

Figure 8: Effect of training objective components on substitution performance on **General** dataset. We report the TPR@FPR = 1% as the score.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately describe the paper's scope, contributions, and assumptions. The main claims are supported by the method, experiments, and stated threat model.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations and scope throughout the threat model, robustness analysis, and conclusion. In particular, it depends on trusted auditor setup and authenticated provisioning and degrades under strong rewriting or adaptive spoofing.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not make formal theoretical claims as its main contribution. The main claims are evaluated empirically.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification:The paper uses public model checkpoints and datasets and discloses the main method, datasets, models, metrics, training objective, and many hyperparameters in Section 4.1.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The public repository linked in the abstract provides the primary substitution detection pipeline for Math, General, and Code, including BERT, GPT-2, and T5 baselines, with setup and execution instructions.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the main experimental setup: datasets, metrics, carrier parameters, training sample size and split, optimizer, learning rate, training epoch count, temperature, and loss weights in Section 4.1. Additional dataset and model details and objective definitions in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper reports low-FPR operating-point metrics such as TPR@FPR=1%, ACC@FPR=1%, and AUC, but does not report confidence intervals, error bars, or statistical significance tests.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Section 4 reports the hardware used for the experiments, but we do not provide per-experiment runtime, peak memory usage, and total compute across runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work studies auditing and transparency for hosted LLM services, we are not aware of any aspect that conflicts with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses the benefits of model-identity auditing for transparency and accountability, as well as risks such as overreliance on summary-channel evidence and adaptive evasion.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We release research code but do not release a high-risk pretrained model, newly scraped dataset, or deployed system.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No]

Justification: The paper uses existing public models, datasets, and baseline methods, but does not provide a complete asset-by-asset listing of licenses, terms of use, versions, and URLs.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the SUMMARK implementation with a README describing installation, model and dataset access, configuration, and execution, together with an Apache-2.0 license.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, paid data annotation, user studies, or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification:The paper does not involve crowdsourcing or human-subject research, so IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core methodology and scientific claims do not use LLMs as an important, original, or non-standard component; any editorial assistance would not affect the research method or results.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.