AuthPrint: Fingerprinting Generative Models Against Malicious Model Providers

Kai Yao, Marc Juarez

School of Informatics, University of Edinburgh
Edinburgh, UK
kai.yao@ed.ac.uk, marc.juarez@ed.ac.uk

Abstract

Generative models are increasingly adopted in high-stakes domains, yet current deployments offer no mechanisms to verify the origin of model outputs. We address this gap by extending model fingerprinting techniques beyond the traditional collaborative setting to one where the model provider may act adversarially. To our knowledge, this is the first work to evaluate fingerprinting for provenance attribution under such a threat model. The methods rely on a trusted verifier that extracts secret fingerprints from the model's output space, unknown to the provider, and trains a model to predict and verify them. Our empirical evaluation shows that our methods achieve near-zero FPR@95%TPR for instances of GAN and diffusion models, even when tested on small modifications to the original architecture and training data. Moreover, the methods remain robust against adversarial attacks that actively modify the outputs to bypass detection. Source codes are available at `https://github.com/PSMLab/authprint`.

Introduction

Recent advances in generative AI have led to the widespread deployment of generative models across various domains, with providers of generative AI services increasingly monetizing their models by offering subscription-based access. However, this rapid adoption has raised serious concerns about the risks posed by these models, particularly in high-stakes and sensitive sectors, such as healthcare and defense, where erroneous model outputs can have disastrous consequences (Bommasani et al. 2021).

In response, policymakers are introducing legal frameworks to regulate the use of AI and, in particular, the deployment of generative models. For instance, the European Union's AI Act mandates independent, periodic *audits* for "high-risk" AI systems deployed in domains such as healthcare, education, employment, and critical infrastructure (European Parliament and Council of the European Union 2023). This requirement to pass or be certified by an audit raises a critical question: *How can users verify that a given output indeed originated from the audited model?* Even if a model passes an audit, there is currently no guarantee that providers will adhere to the audited model and not substitute it for a cheaper but potentially flawed version.

Without verification mechanisms, audits may fail to enforce compliance. In particular, it becomes difficult to detect whether the model deployed after an audit matches the one audited. Similar challenges have been observed in other industries, where companies have deceived auditors by manipulating a product's performance only to pass an audit, while deploying a product with detrimental effects on climate or public health (US EPA 2015). In the context of AI, service providers have similar intentions to reduce costs, possibly disregarding negative societal outcomes (Valdarrama 2023).

This issue is exacerbated in current AI/ML systems, where access to the model is offered through opaque APIs that provide no verifiable information about which models are accessed. Users are left to blindly trust the model provider: for any single query, the model provider can plausibly deny having used a harmful or defective model, as little evidence exists to disprove their claims. The lack of accountability hinders oversight and diffuses responsibility across the ML supply chain (Veale 2023). Therefore, technical means to track models through their lifecycle—from certification to deployment—is critical, not only for law enforcement but also for maintaining trust in AI applications.

Prior work has proposed cryptographic commitment schemes based on zero-knowledge proofs to verify that an output is obtained by evaluating the committed model (Kang et al. 2023; Lee et al. 2024). Alternatively, trusted execution environments have been explored to prevent unauthorized modifications of a model and verify that computations originate from a specific model snapshot (Tramer and Boneh 2018). However, these technical approaches fall short of the requirements for deployment, as they either do not scale to the size of modern generative models, or require specialized hardware, limiting their potential adoption. Most prior work in this area targets small classifiers, overlooking the unique challenges posed by modern generative models.

In this paper, we propose **AUTHPRINT** (**Auth**entication via Finger**print**ing), a novel framework for detecting the source model of generated images. AUTHPRINT relies on a trusted third party (e.g., an auditor) to train a model that learns to reconstruct *secret* fingerprints from images generated by the original audited model. During verification, the reconstructed fingerprint is compared against the true fingerprint to determine whether the image was produced by this authentic model. AUTHPRINT satisfies two key properties: (i) it reliably detects deviations in the output distribution from the original audited model; and (ii) it is robust against

attacks attempting to forge images that pass verification. Our main contributions are:

- We introduce AUTHPRINT, a black-box fingerprinting framework for attributing image outputs to a specific generative model, even under a malicious provider.
- We demonstrate its effectiveness across architectures and datasets, including StyleGAN2 (FFHQ, LSUN-Cat) and multiple Stable Diffusion versions.
- We show that AUTHPRINT remains robust against adaptive attackers with full access to the original model, reducing forgery success rates from 100% to zero.
- We provide a theoretical analysis of fingerprint recovery attacks, showing such attacks are infeasible for attackers with realistic constraints.

Background

In this section, we define the problem and introduce the necessary background on model fingerprinting techniques.

Problem Statement

We consider a setting involving two entities: the *model provider* and the *verifier*. The model provider maintains a generative image model and monetizes it by offering remote access (e.g., via a web API). We assume the model has been trained and can be represented as a function mapping structured inputs to high-dimensional image outputs. More formally, we define an image generation model as a function $G_\psi : \Xi \times \mathcal{C} \to \mathbb{R}^d$, where ψ denotes its parameters, Ξ is the space of latent variables, $\mathcal{C}$ is the space of optional conditioning inputs, such as class labels or text prompts, and $\mathbb{R}^d$ is the output space corresponding to images of dimension $d =$ channels $\times$ height $\times$ width. The parameters ψ include all components required to fully specify the model for evaluation. For example, if G_ψ is implemented as a neural network, ψ would include its weights and architecture.

Before deployment, the model provider must grant the verifier query access. We refer to this pre-deployment stage as the *certification phase*. During this phase, the verifier may issue a number of queries via the API, and the provider responds by evaluating G_ψ on the verifier's inputs and returning the resulting output images. We refer to the period after the model has been deployed as the *verification phase*. The central question we investigate is: *Given any output $x \in \mathbb{R}^d$ at verification time, can the verifier determine whether it was produced by the same model used during certification?*

We envision real-world scenarios where this problem is relevant. Our running example is an AI audit, where the verifier is an auditor certifying the model's compliance with AI regulations, such as fairness and privacy. Depending on how the regulations are implemented, the provider may grant the auditor query access, or extend the access under legal protections for intellectual property. Next, we detail the goals and capabilities of the model provider and verifier.

Threat Model

We consider a model provider aiming to maximize profit from its image generation service. The main threat we address is the provider acting maliciously and replacing the model with one of inferior quality but cheaper to evaluate, thereby reducing costs. The bogus model may produce images with flaws that are imperceptible to humans but may have a significant impact (e.g., quality of medical images in cancer diagnosis). In this paper, we focus on image quality deterioration, as it is often a direct consequence of evaluating a cheaper model. However, there are more subtle model flaws harder to detect on a single output, such as increased privacy risk or overrepresentation of certain demographics.

Verifier's goal and capabilities. The verifier is only permitted black-box access to the model through a large number of queries at certification time. They may use this information to develop a detector D, such that $D(x)$ accurately detects whether or not x was sampled from the certified model.

Adversary's goal. The goal of the attack is to craft an output $\hat{x} \in \mathbb{R}^d$ in such a way that $\hat{x}$ deviates from the certified model's output distribution, but D incorrectly detects as drawn from it. Borrowing terminology from the ML watermarking literature, we call this attack a *forgery attack*, since if successful, the attacker deceives the verifier by making the sample appear to be drawn from the legitimate model.

Adversary's capabilities. During certification, the provider must consistently respond using the same model. Note that the verifier's goal is not to detect model defects, but to uncover a model swap. Moreover, the increased scrutiny during certification would be a deterrent to such misconduct. Second, the model provider will not spend more than the cost of sampling from the certified model, as that is against their goal of saving costs. Importantly, the adversary has full access to the certified model, which gives them a significant advantage in mounting forgery attacks.

Our approach to this problem is based on model fingerprinting techniques, which we introduce next.

Model fingerprinting techniques

Model fingerprinting refers to techniques that identify ML models based on distinctive patterns in their outputs. These methods are typically used to protect the intellectual property of model owners when models are stolen or leaked. By *fingerprinting* a model, unauthorized uses can be identified.

Most fingerprinting techniques fall into two categories: *passive* and *active*. *Passive fingerprinting* relies solely on detecting patterns in model outputs, while *active fingerprinting* modifies the model to make its outputs more identifiable.

We focus on passive techniques, as we assume the verifier only has query access during certification. While white-box access could enable active methods, these are vulnerable in adversarial settings: a provider with access to both original and fingerprinted models could conduct side-by-side comparisons to uncover the fingerprint or mount forgery attacks.

Passive fingerprinting. Prior work has proposed passive fingerprinting methods for model attribution. Marra et al. apply denoising filters to extract *noise residuals* from images, showing that the average residuals of a GAN correlate with its outputs and can serve as a fingerprint (Marra et al. 2019). Improving upon this, Yu et al. explore deep architectures to extract more discriminative features (Yu, Davis, and

Fritz 2019). Corvi et al. identify patterns in the power spectra and autocorrelation of noise residuals of latent diffusion models (Corvi et al. 2023). Lastly, Song et al.'s geometric approach improves detection by approximating the projection of generated images onto the data manifold as vectors to their nearest counterparts in a real image dataset (Song, Khayatkhoei, and AbdAlmageed 2024).

Our approach differs from these works in two key ways. First, instead of a *closed world* of candidate models, we tackle attribution in an *open-world*: whether or not an image originates from the certified model. Second, we assume an adversarial model provider, as opposed to a trusted one. To our knowledge, we are the first to evaluate the susceptibility of these techniques to adaptive forgery attacks.

AUTHPRINT: Authenticity Auditing via Covert Fingerprinting

We introduce AUTHPRINT (Figure 1), a covert, passive fingerprinting framework for verifying the outputs of image generation models when providers of these models may behave adversarially. AUTHPRINT is run by the verifier to build a fingerprint detector during the certification phase and later use it to determine the authenticity of model outputs during the verification phase.

AUTHPRINT could be offered as a service to users of the model provider's API, in which case the verifier operates as a trusted third party. In many real-world scenarios, there are trusted entities who could play the role of the verifier, such as regulatory agencies or public interest auditors.

Certification Phase. As shown in Algorithm 1, the verifier is granted black-box access to a generative model G_ψ and its input distribution P_{in}. This input distribution defines a joint prior over latent variables and optional conditioning inputs, i.e., $(\xi, c) \sim P_{\text{in}}$, where $\xi \in \mathbb{R}^k$ is typically sampled from a normal distribution, and c represents conditioning information, such as prompts. The verifier selects a secret sequence of l pixel indices uniformly at random as a vector, $s \in [d]^l$, where d is the image dimension. The vector s constitutes the model's fingerprint which, importantly, is output-dependent.

Algorithm 1: Certification Phase

Input: P_{in} over $\Xi \times \mathcal{C}$, model $G_\psi : \Xi \times \mathcal{C} \to \mathbb{R}^d$
Parameter: Fingerprint length l, batch size B, learning rate η, training steps T
Output: Detector D

Sample secret index vector: $s \xleftarrow{\$} \mathcal{U}([d]^l)$
Initialize reconstructor parameters ϕ
for $t = 1$ to T **do**
  Sample batch $\{(\xi_i, c_i)\}_{i=1}^B \xleftarrow{\$} P_{\text{in}}$
  Generate images: $x_i \leftarrow G_\psi(\xi_i, c_i)$
  Reconstruct fingerprint: $r_i \leftarrow R_\phi(x_i)$
  Extract fingerprint: $f_i \leftarrow (x_i)_s$
  Compute loss: $\mathcal{L} \leftarrow \frac{1}{B}\sum_{i=1}^B \frac{1}{l}\|r_i - f_i\|_2^2$
  Update reconstructor: $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}$
end for
return detector $D = (R_\phi, s)$

The goal of the certification phase is to train a *fingerprint reconstructor* model $R_\phi : \mathbb{R}^d \to \mathbb{R}^l$—typically a deep neural network—that can predict the fingerprint values (i.e., the pixel values at the secret indices s) directly from a generated image. To train the reconstructor, the verifier repeatedly samples batches (ξ_i, c_i) from P_{in} and generates corresponding images $x_i = G_\psi(\xi_i, c_i)$. For each image, the reconstructor computes a prediction $r_i = R_\phi(x_i)$, and the true fingerprint $f_i = (x_i)_s$ is extracted according to s. The fingerprint reconstructor is trained to minimize the mean squared error (MSE) between prediction and ground truth fingerprint values over the batch, i.e., $\mathcal{L} = \frac{1}{B}\sum_{i=1}^B \frac{1}{l}\|r_i - f_i\|_2^2$, where B is the batch size. After training, the verifier outputs a fingerprint detector $D = (R_\phi, s)$, consisting of the reconstructor R_ϕ and the secret indices s used for training R_ϕ.

Verification Phase. As shown in Algorithm 2, once trained, the detector $D = (R_\phi, s)$ is deployed as a black-box verification API controlled by the verifier. At inference, a user submits a test image x'; the reconstructor computes $r' = R_\phi(x')$, and the verifier also extracts the fingerprint $f' = (x')_s$. The detection error $e = \frac{1}{l}\|r' - f'\|_2^2$ is compared to a pre-agreed threshold τ (set by the provider and verifier to yield low FNR). If $e < \tau$, the image is deemed *authentic*, meaning it originates from G_ψ; otherwise, *not authentic*.

Algorithm 2: Verification Phase

Input: Test image x', detector $D = (R_\phi, s)$
Parameter: Threshold τ
Output: Binary decision

Reconstruct fingerprint: $r' \leftarrow R_\phi(x')$
Extract fingerprint: $f' \leftarrow (x')_s$
Compute detection error: $e \leftarrow \frac{1}{l}\|r' - f'\|_2^2$
if $e < \tau$ **then**
  return `Authentic` (i.e. x' originates from G_ψ)
else
  return `Not authentic`
end if

The core intuition behind AUTHPRINT lies in the fine-grained pixel-level dependencies learned by modern generative models (van den Oord, Kalchbrenner, and Kavukcuoglu 2016; Theis and Bethge 2015). A reconstructor trained to recover a secret subset of pixels from generated images performs well when the images are drawn from the distribution of training images (i.e., the certified model's output distribution) but, if the provider substitutes the model with another generator, even slight shifts in these statistical dependencies result in prediction errors. AUTHPRINT exploits this out-of-distribution sensitivity by training the reconstructor on ample samples from the original model, enabling robust detection of distributional shifts without modifying the model or images. It can be directly applied to a wide range of models and learning algorithms, requires no retraining of the generator, is hardware-independent (unlike secure hardware-based approaches (Tramer and Boneh 2018)), and remains practical for high-capacity models where cryptographic approaches such as zero-knowledge proofs are infeasible.

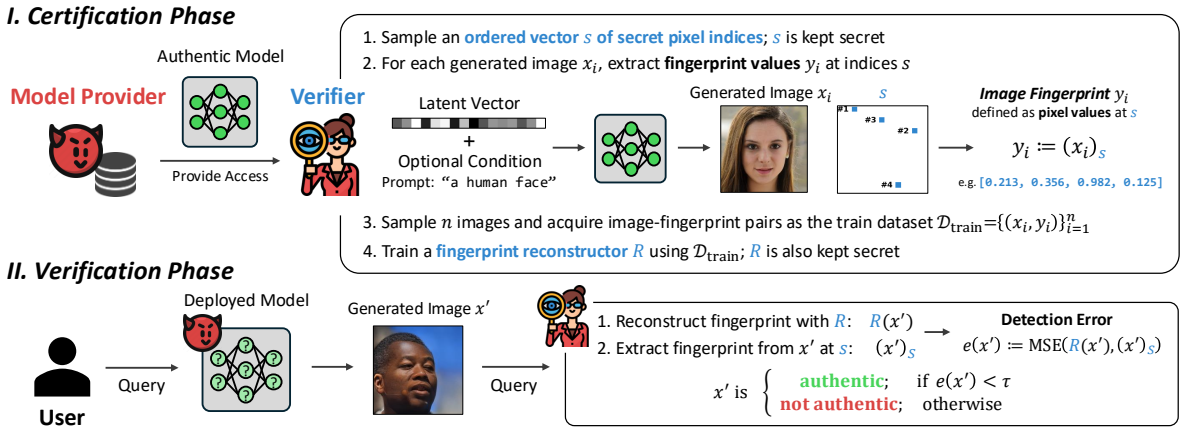

Figure 1: Overview of the AUTHPRINT pipeline. In the certification phase, a verifier with black-box access to the claimed model selects secret pixel locations (channel-aware; simplified in figure for illustration) as a fingerprint and trains a private reconstructor to predict them from generated images. After certification, the provider deploys the model via its black-box API, while the verifier hosts a separate verification API with the reconstructor and secret indices. In the verification phase, a user submits an image to this API, which applies the reconstructor, compares predicted and actual values at the secret locations, computes a detection error, and returns the authenticity verdict.

The security of AUTHPRINT rests on the secrecy of the detector, namely the reconstructor parameters ϕ and the index vector s. The model provider is aware of AUTHPRINT's algorithms and may query the detector during the verification phase, e.g., impersonating a user of the verification API, but the API only provides binary responses. Importantly, the model provider has no access to gradients, logits, or auxiliary information about the reconstructor. Our evaluation shows that this information asymmetry thwarts forgery and reverse-engineering attacks, ensuring the robustness of AUTHPRINT even when deployed in an adversarial setting.

Results

Detection Performance on Model Substitution

We first evaluate AUTHPRINT's detection performance on model substitution for both GAN-based (StyleGAN2) and diffusion-based (SD) models (see Appendix A for experiment details). To test rejection of negative models, we report *detection error* as the FPR at 95% TPR, treating target model samples as positives. As an ablation, we vary the fingerprint length (i.e., number of secret pixels) to assess its effect on detection error, holding all other factors constant[1].

In the StyleGAN2 experiments, the target and negative models share the same architecture but differ in training configurations such as dataset size and data augmentation. This setup probes AUTHPRINT's sensitivity to subtle changes in the training procedure that induces negligible image quality shifts. We evaluate on FFHQ and LSUN-Cat, with results shown in Figure 2. For SD, the target is SD 2.1, and negatives are earlier versions (SD 1.5 to 1.1), introducing progressively larger distributional shifts due to architectural and dataset differences. Results are shown in Figure 3.

[1]Variance across random seeds is $<1\%$; we report a single run.

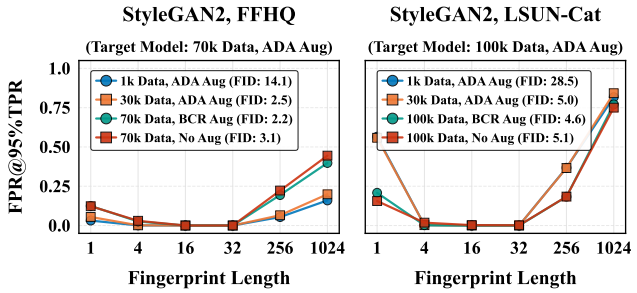

Figure 2: AUTHPRINT detection performance on model substitution for pre-trained StyleGAN2 models (left: FFHQ, right: LSUN-Cat) across varying fingerprint lengths. Target and negative models share the architecture but differ in training data size or augmentation, as detailed in the figure.

AUTHPRINT consistently achieves low detection error across both model families, validating its robustness in distinguishing target from non-target outputs. Detection performance peaks at fingerprint lengths of 16 and 32 for StyleGAN2 and 64 for SD. Short fingerprints yield insufficient signal, while overly long ones introduce redundancy that limits reconstructor accuracy under fixed capacity.

Detection on SD models appears weaker than on GANs, largely due to practical limitations in computational resources. Generating a 1024×1024 image from SD 2.1 (25 denoising steps) is roughly $50\times$ slower than a 256×256 StyleGAN2 image, which constrains the volume of training data. As shown in later sections, reconstructor capacity and training set size play a crucial role in performance.

Another contributor to the performance gap is the generative process itself. GANs leave subtle artifacts or statistical

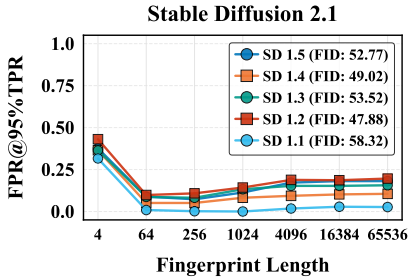

Figure 3: AUTHPRINT detection performance on model substitution for SD models across varying fingerprint lengths. The target is SD 2.1; negative models are earlier versions (SD 1.5, 1.4, 1.3, 1.2, 1.1) with lower image qualities. The *Japanese café* prompt is used throughout.

footprints that AUTHPRINT may implicitly exploit (Marra et al. 2019). Diffusion models, by contrast, suppress such artifacts through iterative denoising. As shown in Figure 9 in Appendix B, reducing denoising steps substantially lowers detection error in SD—suggesting that shallower sampling may preserve more model-specific traces.

Scaling Up Training Enhances Detection

Motivated by the performance gap between StyleGAN2 and SD models, we examined whether increased training resources improve detection. We focused on two factors: (1) reconstructor model size and (2) number of samples for training the reconstructor, both of which significantly affected detection performance.

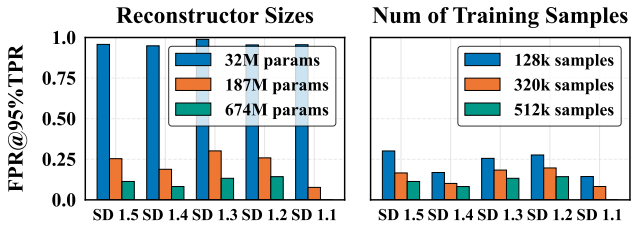

Figure 4: Effect of reconstructor size (left) and reconstructor training set size (right) on AUTHPRINT performance for SD models. Both experiments use the *Japanese café* prompt and a fingerprint length of 1024. Left: reconstructor size comparison (32M, 187M, 674M) using 512k training samples. Right: training set size comparison (128k, 320k, 512k samples) using a 674M-parameter reconstructor.

As shown in Figure 4, we evaluated reconstructors with 32M, 187M, and 674M parameters on SD models, holding all other variables constant. The smallest (32M) failed to reject negative samples, indicating insufficient capacity for the fingerprinting task. In contrast, performance improved substantially with larger reconstructors, and the 674M model achieved near-perfect detection error on SD 1.1.

We also varied reconstructor's training set size (128k, 320k, 512k samples) and observed vanishing detection error with more data. These results show that scaling both reconstructor capacity and training volume significantly improves performance. In our most demanding setting (674M reconstructor, 512k samples), training took ∼240 GPU hours on an NVIDIA H200—costly for academic clusters but feasible in industrial or governmental settings. Importantly, training is only needed once per target model.

For comparison, our StyleGAN2 experiments used 96 million samples—∼200× more than the largest SD training set—indicating substantial headroom for improving SD fingerprinting via data scaling. With sufficient resources, the detection performance of AUTHPRINT for SD may approach that for GANs.

Boosting Performance with Ensemble Detection

To reduce FPR, AUTHPRINT can be naturally extended with an *ensemble detection* scheme, where multiple reconstructors are independently trained using different fingerprint index vectors. At verification, each image is evaluated by all reconstructor-index pairs, and the final detection error is defined as the **worst-case** MSE across the ensemble, i.e., $e(x') = \max_{i \in [N]} \text{MSE}(R_{\phi_i}(x'), (x')_{s_i})$, where each R_{ϕ_i} is trained on s_i, and N is the ensemble size. This conservative strategy boosts sensitivity: while a negative sample may yield low error on one reconstructor-index pair, it is unlikely to do so across all, thereby reducing false acceptances.

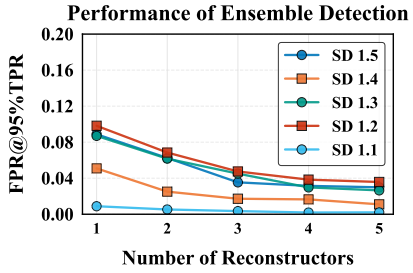

Figure 5: Effect of ensemble detection on AUTHPRINT performance for SD models using the *Japanese café* prompt. Each reconstructor has 674M parameters, trained on 512k samples with fingerprint length 64.

We evaluate this scheme on SD models (Figure 5) using identical hyperparameters but distinct index sets per reconstructor. Detection error decreases with ensemble size, with diminishing returns suggesting convergence. The ensemble design is flexible and can be extended to heterogeneous configurations with varying fingerprint lengths, architectures, or training setups. We leave exploration of such to future work.

Detection of Target Model Modifications

We evaluate AUTHPRINT's sensitivity to model compression (weight quantization and magnitude-based pruning) and output image downsampling. Quantization and pruning reflect model modifications that a malicious model provider might conduct to save costs at the expense of performance, while advertising the original, unmodified model. Although downsampling is not strictly a model modification and falls

outside our threat model—since it requires a postprocessing step after image generation and thus incurs additional costs to the provider—we include it to study the impact of image quality degradation on detection error. We experiment with varying levels of pruning, quantization, and downsampling.

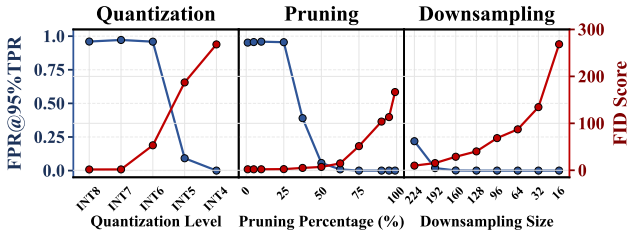

Figure 6: Effect of model modifications on detection performance for StyleGAN2 FFHQ at fingerprint length 32. We evaluate model compression (quantization and pruning) and output perturbations (downsampling). Blue dots (left y-axis) indicate FPR@95%TPR; red dots (right y-axis) indicate FID. Each plot varies the quantization level, pruning ratio, or downsampling resolution to show trends.

Figure 6 shows the relationship between detection error (FPR) and image quality (FID) under all perturbations. AUTHPRINT achieves effective trade-offs under pruning and downsampling. For example, moderate pruning (62.5%) increases FID modestly to 14.69 but drives FPR to near-zero. Similarly, downsampling from 256×256 to 224×224 sharply reduces FPR despite only slight increases in FID. Quantization, by contrast, appears more robust to detection: high precision loss (INT4) renders high FID (> 200) and leads to 0 FPR as expected, but moderate quantization (INT6) results in high FPR despite quality degradation (FID ≈ 53). This indicates that detection is less effective for quantization than for pruning or downsampling under the same fingerprint configuration. These results indicate that while AUTHPRINT is effective at detecting distribution shifts, its sensitivity varies by perturbation type.

Prompt Specificity Enhances Detection for SD

Unconditioned generators like StyleGAN2 sample from a fixed latent distribution, yielding a well-defined and bounded output distribution $p(x)$. In contrast, conditional models such as SD are typically trained to solve more general generation tasks, conditioned to a prompt c, and thus have a much broader range of output distributions $p(x \mid c)$.

This raises a key question for AUTHPRINT: *How does prompt specificity affect detection performance*? We evaluate detection error under five prompt regimes, ordered by increasing specificity (see Appendix A for prompt details):

I. Diverse prompts from DiffusionDB,

II. "People"-category prompts from PartiPrompts,

III. A focused subset of simple "People" prompts (e.g., "a man", "a woman", "a child"),

IV. A single fixed generic prompt ("a high quality photo"),

V. A single fixed specific prompt (*Japanese café*)

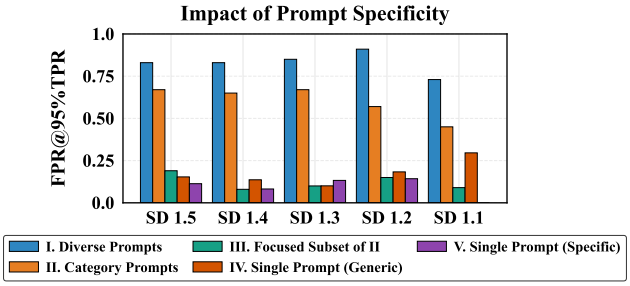

Figure 7: Effect of prompt specificity on detection performance for SD. We compare five regimes: I. Diverse prompts from DiffusionDB; II. Category-specific "People" prompts from PartiPrompts; III. A focused subset of II (see Appendix A); IV. A single generic prompt ("a high quality photo"); V. A single specific prompt (*Japanese café* prompt).

As shown in Figure 7, detection improves consistently with greater prompt specificity. For diverse prompts, FPR exceeds 80%, indicating poor separation between models. Restricting prompts to the "People" category reduces FPR to below 65%, and the focused subset lowers it to around 10%. The best performance (FPR $< 5\%$) occurs under a fixed, specific prompt. A fixed generic prompt performs similarly in some cases, but with slightly higher error overall.

These results reflect an intuitive pattern: diverse prompts lead to a broad, overlapping output space, making it harder for the detector to distinguish models. Specific prompts constrain the output distribution, making model differences more detectable. Importantly, strong performance only for specific prompts does not limit practical applicability. Many real-world applications—e.g., profile portraits, product photos, architectural renderings, or avatars—rely on consistent, repeatable prompts, where our approach can be effective.

Evaluation of Adaptive Adversarial Attacks

Evasion Attacks. A common strategy for bypassing detectors is evasion, such as attacks based on *Projected Gradient Descent (PGD)* (Madry et al. 2017). In our setting, evasion translates to forgery, where an adversary perturbs an image sampled from a negative model to evade detection. We evaluate this threat in the model substitution scenario, using a reconstructor trained with a fingerprint length of 32 on a target StyleGAN2 model (FFHQ, 70k, ADA), against the four negative models shown in Figure 2 (left) that yield exactly 0% FPR@95%TPR. The adversary has full access to both target and negative models. It trains a classifier to distinguish their outputs and uses it to guide a PGD attack on the detector. PGD parameters are in Appendix A.

We compare the robustness of three detectors:

1. **Baseline:** A ResNet-18 classifier trained to distinguish target vs. negative outputs without using any fingerprint.
2. **Yu-2019:** A state-of-the-art model fingerprinting approach where fingerprints are public and not output-dependent (Yu, Davis, and Fritz 2019).
3. **AUTHPRINT:** Our proposed method using a reconstructor trained on secret, output-dependent fingerprints.

With AUTHPRINT, the attacker operates in a regime where the detector details are kept secret. To mount the attack, they assume the reconstructor has the same architecture as the baseline classifier, train a surrogate detector, and use its gradients to guide PGD. For each detector, we report the attack success rate, average number of steps to success, and LPIPS perceptual distance of the forged images.

Results show that AUTHPRINT is immune to this attack: the attacker fails to generate even a single successful forgery. As shown in Table 1, Appendix C, both baseline detectors are defeated by PGD, with a 100% attack success rate—i.e., all non-authentic images are misclassified as authentic. Moreover, the attacks preserve perceptual quality, with LPIPS consistently $<$0.01. While the Yu-2019 detector requiress slightly more steps to evade, both are ultimately ineffective. In contrast, AUTHPRINT achieves a 0% attack success rate. The attacker's inability to access the secret pixel indices or the reconstructor prevents them from training a viable surrogate. This secrecy is essential: if the attacker had white-box access to the reconstructor, or knowledge of the fingerprint index vector, a PGD attack would likely succeed.

Fingerprint Recovery Attacks. Beyond evasion-based forgery, an adversary may attempt to recover the fingerprint indices s. We decompose this attack into two steps: first, recovering the fingerprint *set* $S := \mathrm{set}(s)$; second, searching the ordered *vector* s from S. The order is crucial—without it, the adversary cannot replicate the behavior of R.

Step 1: Recover the set S. To recover the set S, one can probe the detector through pixel manipulation: starting from an authentic image x, the attacker manipulates a subset $M \subset [d]$ of m pixels (e.g.setting them to 0) to produce a modified image x', and observes whether it passes or is rejected.

The detector computes $D(x) = \mathrm{MSE}[R(x), x_s]$, which stays low if two conditions hold: (1) $R(x)$ is stable, which happens when m is small (see Figure 11, Appendix D), and (2) the manipulated pixels do not significantly overlap with the fingerprint set S. In other words, the overlap $|M \cap S|$ must be large enough to corrupt x_s for the MSE to increase and lead to rejection. We refer to this required overlap as rejection threshold k (see Appendix D for how k is estimated).

This rejection behavior reduces Step 1 to the *non-adaptive threshold group testing* (NATGT) problem, where the fingerprint S plays the role of the hidden "defective" set, and each attacker query corresponds to a test set M_i. The detector returns positive (rejection) iff $|M_i \cap S| \geq k$. The attacker's goal is to recover S using as few queries as possible.

Equation 1 in Appendix D reflects recent theoretical estimation for the query complexity T from NATGT applied to our problem. For instance, with $d = 3 \cdot 256^2$, $\ell = 32$, and $k = 2$, it yields $T = \mathcal{O}(10^5)$ queries, which is impractical for detectors under oversight, e.g., enforcing rate limits or access controls to prevent abuse. T exhibits approximately exponential growth with respect to the length ratio k/l (see Figure 10 in Appendix D). Importantly, this estimation assumes the attacker can query with arbitrary m, whereas our threat model restricts to small m. As a result, this serves as a *lower bound* on recovery complexity. Tight estimates under limited queries are out of the scope of this work.

An ensemble of detectors can further boost robustness: accepting if any detector accepts reduces false negatives and makes recovery harder. This differentiates from the earlier max-MSE ensemble strategy, which prioritized minimizing false positives. The choice reflects verifier goals and tradeoffs between robustness and performance.

Step 2: Recover the vector s. If the attacker learns S but not its order in s, they cannot train an effective surrogate reconstructor $\hat{R}$ because a different order would result in fingerprint mismatches on out-of-distribution samples. To recover s from S requires searching over all permutations, $O(l!)$. For example, when $l = 32$, this corresponds to roughly $O(10^{35})$. Hence, Step 2 dominates for longer fingerprints, while Step 1 dominates for shorter fingerprints.

Discussion and Conclusion

With AI regulations coming into force, model providers in sensitive domains may be subject to compliance audits, thereby creating a demand for accountability and transparency. To address this, the academic community has proposed mechanisms to attribute generated outputs to the specific model that produced them. However, these approaches face challenges for a scalable and practical deployment.

We have tackled this problem for image generation models by adapting model fingerprinting techniques and investigating their robustness in adversarial settings. Our proposed approach allows a verifier to detect whether a given image was sampled from a model of quality lower than that of a reference model. This scheme does not require specialized hardware or modifications to the reference model, and is compatible with modern generative algorithms, offering a more practical alternative to existing approaches.

We evaluate the scheme on two popular generative models—StyleGAN2 and Stable Diffusion—and observe low detection error for images generated by models trained with sub-optimal configurations, older versions from the same model family, and compressed model variants. Furthermore, the error steadily decreases with more training data and larger reconstructor models, suggesting that verifiers can trade computational resources for improved performance. We also show that the scheme is robust to two types of adversarial forgery attacks: gradient-based evasion and fingerprint recovery.

Although the scheme does not offer the formal guarantees of cryptographic methods, it explores a complementary direction by leveraging statistical differences in output distributions. In real-world scenarios, such evidence may suffice to raise concerns or prompt further investigation. To mitigate attacks, we recommend the verifier follows standard operational security practices to protect the fingerprint indices and reconstructor, enforces rate limits, and deploys API authentication and reputation mechanisms.

We view this work as a step toward practical, scalable verification tools for generative models, especially while cryptographic and hardware-based solutions are still maturing. Promising future directions include ensemble methods, techniques for amplifying detection confidence, and approaches to improve performance on conditioned models.

References

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; et al. 2021. On the Opportunities and Risks of Foundation Models. *arXiv preprint arXiv:2108.07258*.

Boneh, D.; Nguyen, W.; and Ozdemir, A. 2021. Efficient functional commitments: How to commit to a private function. *Cryptology ePrint Archive*.

Bui, T. V.; Cheraghchi, M.; and Echizen, I. 2021. Improved non-adaptive algorithms for threshold group testing with a gap. *IEEE Transactions on Information Theory*, 67(11): 7180–7196.

Chen, H.; Rohani, B. D.; and Koushanfar, F. 2018. Deepmarks: A digital fingerprinting framework for deep neural networks. *arXiv preprint arXiv:1804.03648*.

Corvi, R.; Cozzolino, D.; Zingarini, G.; Poggi, G.; Nagano, K.; and Verdoliva, L. 2023. Intriguing Properties of Synthetic Images: From Generative Adversarial Networks to Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 973–982.

European Parliament and Council of the European Union. 2023. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain Union legislative acts. Official Journal of the European Union, pp. 1–144.

Ghodsi, Z.; Gu, T.; and Garg, S. 2017. Safetynets: Verifiable execution of deep neural networks on an untrusted cloud. *Advances in Neural Information Processing Systems*, 30.

Kang, D.; Hashimoto, T.; Stoica, I.; and Sun, Y. 2023. Scaling up Trustless DNN Inference with Zero-Knowledge Proofs. In *Advances in Neural Information Processing Systems*, volume 36. Curran Associates, Inc.

Lee, S.; Ko, H.; Kim, J.; and Oh, H. 2024. vcnn: Verifiable convolutional neural network based on zk-snarks. *IEEE Transactions on Dependable and Secure Computing*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Marra, F.; Gragnaniello, D.; Verdoliva, L.; and Poggi, G. 2019. Do GANs Leave Artificial Fingerprints? In *Proceedings of the 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 506–511.

Nataraj, L.; Mohammed, T. M.; Manjunath, B. S.; Chandrasekaran, S.; Flenner, A.; Bappy, J. H.; and Roy-Chowdhury, A. K. 2019. Detecting GAN Generated Fake Images using Co-Occurrence Matrices. In *IS&T International Symposium on Electronic Imaging (EI) – Media Watermarking, Security, and Forensics 2019*, 532–1–532–7.

Song, H. J.; Khayatkhoei, M.; and AbdAlmageed, W. 2024. ManiFPT: Defining and Analyzing Fingerprints of Generative Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10971–10981.

Theis, L.; and Bethge, M. 2015. Generative Image Modeling Using Spatial LSTMs. In *Advances in Neural Information Processing Systems*, volume 28.

Tramer, F.; and Boneh, D. 2018. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. *arXiv preprint arXiv:1806.03287*.

US EPA. 2015. Notice of Violation to Volkswagen AG, Audi AG, and Volkswagen Group of America, Inc. Notice of violation, U.S. Environmental Protection Agency (EPA).

Valdarrama, S. L. 2023. GPT-4 is getting worse over time, not better. Under Twitter handle @svpino: https://x.com/svpino/status/1681614284613099520. Accessed: Jul 27, 2025.

van den Oord, A.; Kalchbrenner, N.; and Kavukcuoglu, K. 2016. Pixel Recurrent Neural Networks. In *Proceedings of the 33rd International Conference on Machine Learning*, 1747–1756. PMLR.

Veale, M. 2023. Governing the AI Business Model: Platforms All the Way Down? https://efi.ed.ac.uk/event/governing-the-ai-business-model-platforms-all-the-way-down/. Accessed: Dec 15, 2023.

Wahby, R. S.; Ji, Y.; Blumberg, A. J.; Shelat, A.; Thaler, J.; Walfish, M.; and Wies, T. 2017. Full accounting for verifiable outsourcing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2071–2086.

Wang, Z. J.; Montoya, E.; Munechika, D.; Yang, H.; Hoover, B.; and Chau, D. H. 2023. DiffusionDB: A Large-Scale Prompt Gallery Dataset for Text-to-Image Generative Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023)*. Association for Computational Linguistics. Long paper, Best Paper Honorable Mention.

Yu, J.; Xu, Y.; Koh, J. Y.; Luong, T.; Baid, G.; Wang, Z.; Vasudevan, V.; Ku, A.; Yang, Y.; Karagol Ayan, B.; Hutchinson, B.; Han, W.; Parekh, Z.; Li, X.; Zhang, H.; Baldridge, J.; and Wu, Y. 2022. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. *arXiv preprint arXiv:2206.10789*.

Yu, N.; Davis, L. S.; and Fritz, M. 2019. Attributing Fake Images to GANs: Learning and Analyzing GAN Fingerprints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 7555–7565.

A Experimental Methodology

We evaluate AUTHPRINT on two classes of pre-trained image generators: StyleGAN2 and Stable Diffusion (SD).

Pre-trained Generative Models. StyleGAN2 models are sourced from the official StyleGAN2-ADA repository by NVLabs[2] and pre-trained on two datasets: FFHQ and LSUN-Cat. All models generate fixed-size 256×256 outputs. We experiment with several variants trained on different datasets and/or augmentation strategies.

For model substitution evaluations:

- **FFHQ models:** The target model is trained on 70k images with ADA (Adaptive Discriminator Augmentation). The four negative models are: (1) 1k data with ADA, (2) 30k data with ADA, (3) 70k data with BCR (Balanced Consistency Regularization), and (4) 70k data with no augmentation.
- **LSUN-Cat models:** The target model is trained on 100k images with ADA. The four negative models are: (1) 1k data with ADA, (2) 30k data with ADA, (3) 100k data with BCR, and (4) 100k data with no augmentation.

In each case, negative models differ from the target model either in data volume or augmentation method.

SD 2.1 and SD 1.5–1.1 models are obtained from HuggingFace[3], and by default generate 1024×1024 images across all experiments.

Fingerprint Reconstructor. The AUTHPRINT reconstructor is a convolutional neural network that takes a full generated image as input and predicts the pixel values at a secret subset of spatial pixel locations (channel-aware) of this image. It is trained to recover these values solely from the full image itself.

We evaluate three reconstructor sizes: 32M, 187M, and 674M parameters. Unless otherwise noted, the 32M model is used for StyleGAN2 experiments and the 674M model for SD experiments, with all sizes included in the ablation study on reconstructor capacity.

Across all sizes, the architecture follows a common design: a deep convolutional encoder with progressively increasing channel widths and stride-2 downsampling, implemented via stacked convolutional blocks using 4×4 kernels, batch normalization, and LeakyReLU activations. The encoder is followed by a global average pooling layer and a multi-layer MLP head. The model takes an image as input and outputs a vector representation of the fingerprint.

Training Data for Reconstructors. For StyleGAN2, training images are generated by sampling latent vectors $z \sim \mathcal{N}(0, I)$. For SD, unless otherwise noted, training uses a single fixed prompt—referred to as the *Japanese café* prompt—described below and illustrated in Figure 8:

> *"A photorealistic photo for a Japanese café named NOVA CAFE, with the name written clearly on a street sign, a storefront banner, and a coffee cup. The scene is set at night with neon lighting, rain-slick streets reflecting the glow, and people walking by in motion blur. Cinematic tone, Leica photo quality, ultra-detailed textures."*

Figure 8: An example image output from SD 2.1 using the *Japanese café* prompt.

Prompt Specificity Experiments. To evaluate the effect of prompt specificity on fingerprinting for SD models, we use the following prompt regimes:

- **Diverse Prompts:** Randomly sampled from DiffusionDB (Wang et al. 2023) on HuggingFace (`poloclub/diffusiondb`).
- **Category-Specific Prompts:** "People" category prompts from PartiPrompts (Yu et al. 2022) on HuggingFace (`nateraw/parti-prompts`).
- **Focused Subset:** A small set of generic human-related prompts: *"a child," "a man," "a woman," "a girl," "a person," "a boy"*.
- **Generic Single Prompt:** *"a high quality photo"*.
- **Specific Single Prompt:** The *Japanese café* prompt described above.

Generation Configuration. By default, SD images are generated using 25 inference (denoising) steps, classifier-free guidance scale of 7.5, and `fp16` precision.

Training Details. Reconstructors are trained using the Adam optimizer to minimize MSE between predicted (reconstructed) and ground-truth pixel values at secret indices. Learning rates are set to 10^{-3} for StyleGAN2 experiments and 10^{-4} for SD experiments for best convergence.

PGD Setups in Forgery Attacks. We evaluate the forgery threat on a reconstructor trained for a target StyleGAN2 model (FFHQ, 70k, ADA) at fingerprint length 32, against four negative models: (i) 1k with ADA, (ii) 30k with ADA, (iii) 70k with BCR, and (iv) 70k without augmentation. As shown in Figure 2 (left), all negative variants yield 0% FPR@95%TPR at fingerprint length 32.

PGD attacks are configured with an ℓ_∞ perturbation budget of $\epsilon = 0.5$, a step size of 0.001, and momentum of 0.9. Attacks are run for up to 500 steps with early stopping upon successful evasion. These settings are used across all forgery experiments unless otherwise specified. We evaluated multiple PGD configurations by sweeping over different levels of

[2]https://github.com/NVlabs/stylegan2-ada-pytorch

[3]https://huggingface.co

step sizes and momentum values, and this setup yielded the strongest attack performance against baseline detectors. All configurations consistently result in a 0% attack success rate against the AUTHPRINT detector.

Hardware. All experiments are conducted on NVIDIA H200 GPUs.

B Impact of Denoising Steps on Detection Performance in SD

As shown in Figure 9, reducing the number of denoising steps significantly reduces detection error for SD models.

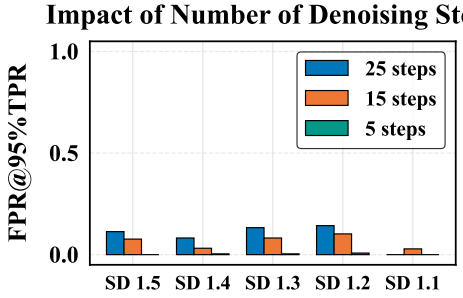

Figure 9: Impact of the number of denoising steps on the performance of AUTHPRINT for SD models. The evaluation uses the *Japanese café* prompt, with a fingerprint length of 1024, a reconstructor with 674M parameters, and 512k training samples. We compare three inference settings: 25, 15, and 5 steps.

C Evasion Attacks

The experiment results for the evasion-based forgery attack using PGD are shown in Table 1.

D Fingerprint Recovery Attacks

Let $x \in \mathbb{R}^d$ denote an image sampled from the certified generator, and $x' \in \mathbb{R}^d$ a manipulated version of x. Let $S \subseteq [d]$ be the secret fingerprint index set of size $l := |S|$, and $M \subseteq [d]$ be the set of manipulated pixel indices. Let $R : \mathbb{R}^d \to \mathbb{R}^l$ denote the fingerprint reconstructor, with prediction $R(x) = r \in \mathbb{R}^l$. Let τ be the needed increase in MSE for a rejection to take place, and define $\delta^2 := \mathbb{E}[(x_i - x'_i)^2]$ as the expected pixel perturbation variance over manipulated indices.

Derivation of Rejection Threshold. Here, we derive the rejection threshold k—defined as the minimum number of pixel indices in the manipulated set M that must overlap with the true fingerprint set S (i.e. the minimum $|M \cap S|$) for the detector to reject a manipulated authentic image.

Under reconstructor stability, $R(x') \approx R(x) = r$, thus the change in the MSE before and after manipulation is:

$$\Delta\text{MSE}(x', x) = \frac{1}{l} \sum_{i \in S} \left[(r_i - x'_i)^2 - (r_i - x_i)^2 \right].$$

Only manipulated fingerprint pixels contribute to the delta in MSE, therefore we obtain:

$$\Delta\text{MSE}(x', x) = \frac{1}{l} \sum_{i \in M \cap S} \left[(r_i - x'_i)^2 - (r_i - x_i)^2 \right].$$

Assuming that R is accurate, i.e., $r_i \approx x_i$, we obtain:

$$\Delta\text{MSE}(x', x) = \frac{1}{l} \sum_{i \in M \cap S} \left[(x_i - x'_i)^2 \right],$$

thus, we can approximate the delta in MSE as:

$$\Delta\text{MSE}(x', x) \approx \frac{|M \cap S| \cdot \delta^2}{l}.$$

Thus, the detector rejects if

$$|M \cap S| \geq k := \left\lceil \frac{\tau l}{\delta^2} \right\rceil,$$

where k is the minimum number of manipulated fingerprint pixels required to trigger detection.

As an example, suppose $l = 32$, $\tau = 0.01$, and $\delta = 0.5$. Then $\delta^2 = 0.25$, and the rejection threshold becomes:

$$k = \left\lceil \frac{0.01 \times 32}{0.25} \right\rceil = \lceil 1.28 \rceil = 2.$$

This means the detector will reject any image where at least $k = 2$ fingerprint pixels in S have been manipulated.

Theoretical Query Complexity Estimation by NATGT. We reduce Step 1 of the problem of fingerprint recovery to a group testing task. Under the *non-adaptive threshold group testing* (NATGT) framework, the query complexity required to recover S depends on the fingerprint length l, the rejection threshold k, and the image dimensionality d.

Theoretical analysis of NATGT shows that the query complexity in our problem satisfies:

$$T = \mathcal{O}\left(\left(\tfrac{l+1}{k}\right)^k \left(\tfrac{l+1}{l-k+1}\right)^{l-k+1} (l+1) \ln\left(\tfrac{d}{l+1}\right) \right), \quad (1)$$

as derived in (Bui, Cheraghchi, and Echizen 2021). Their analysis considers NATGT in the general case with a *gap*, where the test outcome may be uncertain if the number of intersecting items lies between two thresholds. In our fingerprint recovery setting, however, we consider the gap-free regime, where tests yield strictly binary outcomes (accept or reject) with only one threshold k. By setting the gap to zero in their results and substitute with our parameters in the problem, we obtain the above complexity estimation in Equation 1.

Impact of k/l on Query Complexity. To understand the relationship between AUTHPRINT parameters and query complexity estimation under NATGT, we examine how the query complexity T varies with respect to the length ratio k/l according to Equation 1. We systematically evaluate this relationship by fixing different fingerprint lengths $l \in \{32, 64, 128, 256, 512\}$ and computing the corresponding query complexity as k/l ranges from $1/32$ to $1/2$. We maintain $d = 3 \times 256^2$ for all data points. The results are shown in Figure 10.

Table 1: PGD-based evasion attack results under model substitution. Each group reports: success rate (SR, %), average PGD steps, and LPIPS distance for successfully attacked images. "–" indicates no successful attack within 500 steps. The positive model is FFHQ-70k (ADA aug.).

Negative Case	Baseline			Yu-2019			AUTHPRINT		
	SR	Steps	LPIPS	SR	Steps	LPIPS	SR	Steps	LPIPS
FFHQ-1k (ADA aug.)	100.0	2.17	0.0016	100.0	12.98	0.0076	**0.0**	–	–
FFHQ-30k (ADA aug.)	100.0	1.06	0.0002	100.0	4.52	0.0008	**0.0**	–	–
FFHQ-70k (BCR aug.)	100.0	1.23	0.0001	100.0	3.54	0.0003	**0.0**	–	–
FFHQ-70k (no aug.)	100.0	1.45	0.0001	100.0	3.02	0.0003	**0.0**	–	–

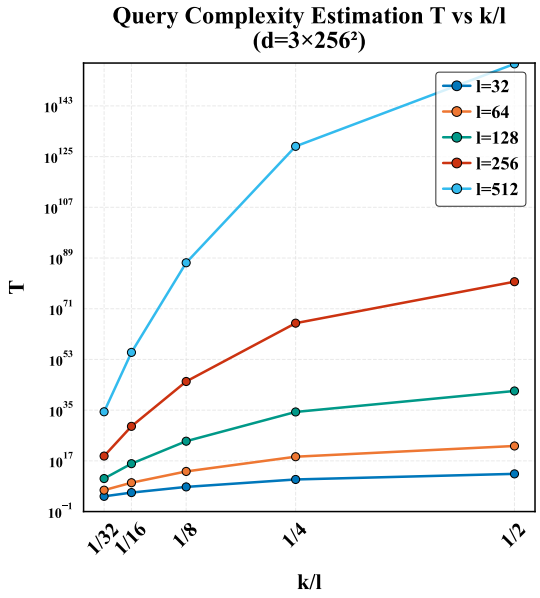

Figure 10: Query complexity T plotted against length ratio k/l under different fixed fingerprint lengths l. All experiments use image dimensionality $d = 3 \times 256^2$.

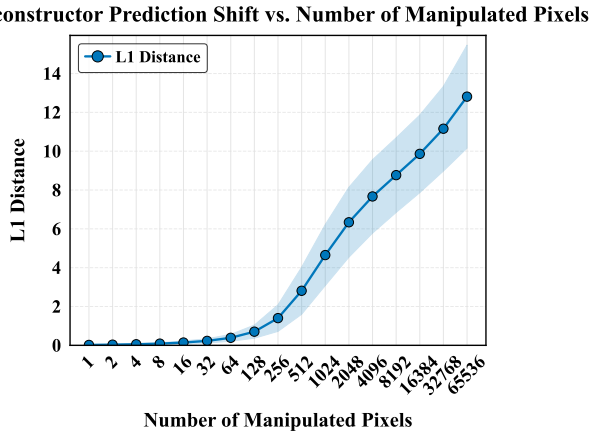

Figure 11: Reconstructor stability under pixel manipulation. Mean ℓ_1 distance between reconstructor outputs remains low for small m, supporting the stability assumption.

Reconstructor Stability Experiment. We validate the assumption that the reconstructor output $R(x)$ remains stable under small manipulations to image pixels for images from the authentic model used for training R. Using 100 generated images from a pre-trained StyleGAN2 model (FFHQ, 70k train data, ADA augmentation), we apply 1,000 random manipulation trials per image, varying the number of manipulated pixels $m \in \{1, 2, 4, \ldots, 65536\}$. The reconstructor R is trained with fingerprint length $l = 32$. In each trial, we measure the mean ℓ_1 distance between the reconstructor outputs on the clean and manipulated images. The results are shown in Figure 11.

E Related Work

Cryptographic Approaches. Ensuring that an output originates from a specific ML model is an emerging challenge. Most cryptographic approaches focus on proofs for the correctness of the model's computation. Early probabilistic proof systems for verifiable computation were highly interactive (Wahby et al. 2017; Ghodsi, Gu, and Garg 2017), which renders them unsuitable to our setting. Functional commitments offer *zero-knowledge* (ZK) proofs of the evaluation of a committed function for functions that can be represented as arithmetic circuits (Boneh, Nguyen, and Ozdemir 2021). Subsequent works in the ZK-ML literature have implemented small neural networks as arithmetic circuits, enabling zk-SNARKs (Kang et al. 2023; Lee et al. 2024). However, the conversion to an arithmetic circuit has a cost in model accuracy that would discourage adoption.

A different approach is to create an immutable snapshot of the model by loading it into a trusted execution environment (TEE) (Tramer and Boneh 2018). The auditor can then publish a certificate with the TEE's public key that clients may use to verify through *remote attestation*. TEEs on GPUs are still under development, and thus most model providers following this approach would not be able to benefit from the speedups of GPU computing for popular large generative models. To address this issue, Tramèr and Boneh propose delegating computations to an untrusted GPU (Tramer and Boneh 2018) and verify them with a probabilistic integrity check. To provide confidentiality toward the GPU, they use a symmetric cipher, which requires quantizing the model weights, thus this approach not only impacts latency but also model accuracy.

The practical limitations of cryptographic methods highlight the need for lightweight approaches that trade verifica-

tion guarantees for practical feasibility.

Other Related Work on Fingerprinting Techniques. Fingerprinting techniques like the ones used for model fingerprinting have also been explored for deepfake vs. real detection. For example, Nataraj et al. observed that GANs tend to exhibit higher co-occurrence of pixels across generated images than real images do, and train a classifier to distinguish between GAN and real based on co-occurrence matrices (Nataraj et al. 2019). Although these techniques share similarities, they address a binary classification problem, grouping all models into the same "artificial" class.

In addition, the term model fingerprinting has been used to describe techniques that embed an identifier in the model's parameters with the goal of detecting unauthorized uses of the model (Chen, Rohani, and Koushanfar 2018). Although these methods share the objective of protecting intellectual property with the model fingerprinting methods we build upon, the underlying techniques are differ fundamentally and operate under different threat models: the verifier requires white-box access to the model in order to verify information encoded in the weights.