Supplementary Material for Black-Box Forgery Attacks on Semantic Watermarks for Diffusion Models

Table of Contents for the Supplementary Material

A. Semantic Watermarking

In this section, we present more details on the examined semantic watermarking methods.

Tree-Ring. Wen et al. [41] have initially presented the concept of inversion-based semantic watermarking. Fig. 11a illustrates the proposed Tree-Ring watermark approach, which can be divided into two phases.

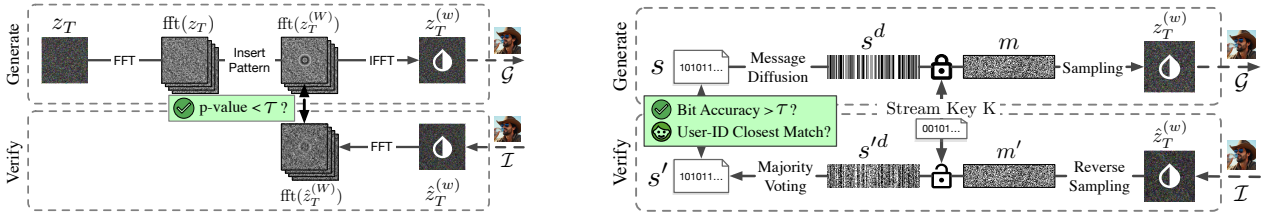
- *Generation.* During image generation, Tree-Ring modifies a clean initial latent $z_T \sim \mathcal{N}(\mathbf{O}, \mathbf{I})$ by adding a concentric circular pattern into its frequency representation. This step produces the watermarked latent noise $z_T^{(w)}$. This noise vector is then used as usual in the further generation process (denoising + decoding) to finally obtain the watermarked image: $x^{(w)} = D(\mathcal{G}_{T\to 0}(z_T^{(w)}; u))$.
- *Verification.* The inverse $\hat{z}_T^{(w)} = \mathcal{I}_{0\to T}(\mathcal{E}(x^{(w)}); u)$ is computed, and then the existence of the original circular pattern in the frequency spectrum of $\hat{z}_T^{(w)}$ is checked. The final detection is based on a statistical test. The test statistic aggregates the squared absolute difference between the observed and the expected frequency values from each respective ring. The null hypothesis H_0 is that an input image is not watermarked. In this case, its inverted initial latent noise should follow a Gaussian distribution with an unknown variance. Under H_0, the test statistic leads to a noncentral χ^2 distribution. This allows computing a p-value that reflects the probability of observing the test-statistic output under the assumption that the input image is not watermarked. If the p-value is below a pre-defined threshold τ, H_0 is rejected and the watermark's presence is assumed.

Tree-Ring is designed as a so-called zero-bit watermarking scheme, as only watermark presence or absence can be detected. Note that the modification to the sampling procedure changes which images will be generated (it is not distribution-preserving), but appears to still have sufficient variety in generated images while retaining image quality.

A subsequent work, RingID [8], improves Tree-Ring by making it more robust to perturbations, ensuring that the distribution of $z_T^{(w)}$ is closer to $\mathcal{N}(\mathbf{O}, \mathbf{I})$, and providing so-called multi-bit watermarking which can carry a multiple bit long message and thus allows distinguishing between different users.

Gaussian Shading. Yang et al. [45] expand the previous concept by relying on cryptographic primitives to achieve distribution-preserving generation. Fig. 11b illustrates the two phases of Gaussian Shading:

- *Generation.* A message s of length k is created. This message s is first repeated ("diffused") ρ times to obtain s^d, which is then encrypted using the symmetric stream chipher ChaCha20 [4]. This stream cipher takes a secret key and a nonce as input, which are completely random for each image. The encrypted message m is then used to steer the sampling of $z_T^{(w)}$. To this end, Gaussian Shading splits a Gaussian distribution into 2^ℓ bins with equal probability. In the following, we focus on the standard setting with $\ell = 1$. This means we have two bins, the negative and the positive area of the Gaussian curve. The bit in the encrypted message $m[i] \in \{0, 1\}$ specifies if $z_T^{(w)}[i]$ is sampled from the negative or the positive area of a Gaussian distribution. Due to the encryption, the bits in m are uniformly distributed, ensuring a similar number of positive and negative samples from the Gaussian distribution. Hence, $z_T^{(w)}$ still follows a Gaussian distribution. After this sampling step, the usual generation process continues with $x^{(w)} = D(\mathcal{G}_{T\to 0}(z_T^{(w)}; u))$.
- *Verification.* To verify the watermark, Gaussian Shading also relies on a full inversion using $\mathcal{I}_{0\to T}(\hat{z}_0^{(w)}; u)$ to obtain an estimated $\hat{z}_T^{(w)}$. This inverted latent noise $\hat{z}_T^{(w)}$ is quantized to retrieve encrypted message bits m', that is, we assume



(a) Tree-Ring (b) Gaussian Shading

Figure 11. Concept of the two inversion-based semantic watermarking approaches

$m'[i] = 0$ if $\hat{z}_T^{(w)}[i] < 0$ and $m'[i] = 1$ if $\hat{z}_T^{(w)}[i] \geq 0$. Next, m' is decrypted to reconstruct the message s'^d. Since each bit was repeated ρ times, its value is recovered by performing a majority voting on all its duplicates to finally obtain s'. This step corrects errors and makes the scheme more robust to noise.

Gaussian Shading is applicable for both zero-bit watermarking and multi-bit watermarking. In both cases, s' is compared to some s bit-wise and the number of matches per message length is recorded as bit-accuracy $r(s, s')$. Like Tree-Ring, a statistical test is performed, but this time the distribution is assumed to be binomial if the image is not watermarked and therefore the test is computed using a regularized incomplete beta function. In zero-bit watermarking, s' is compared to the predefined message s used by the SP. It is checked if a certain fraction of at least τ % of the bits match, such that it is unlikely to observe this by chance with a predefined FPR, which is denoted as $\mathrm{FPR}(\tau)$. In multi-bit watermarking, s' is a specific user id. It is compared with all user ids known to the SP. The match with the best accuracy $r(s, s')$ is selected and then checked if the value is above the threshold τ. As this resembles a multiple test, in order to maintain a certain FPR, the threshold τ is adapted depending on the number of users N by

$$\mathrm{FPR}(\tau, N) = 1 - (1 - \mathrm{FPR}(\tau))^N. \tag{6}$$

For further information, we refer to Yang et al. [45, Section 7.1].

Deployment Ambiguity of Gaussian Shading. Finally, we note that there is a considerable ambiguity on how Gaussian Shading [45] is deployed and evaluated. This is examined in detail by Thietke et al. [39]. In a nutshell, the correct and secure way to use Gaussian Shading is to draw a *new* nonce (and/or a new secret key) for the stream cipher for each generated image. This is how Gaussian Shading is implemented in its original Github repository. However, this key-nonce aspect and the resulting need for key management were not clearly addressed in the original publication, leading to some misconceptions. For example, prior work [17, 44] has used the *same* key and nonce for the generated images. This reduces image diversity and also makes the watermarking scheme vulnerable to attacks that leverage multiple watermarked reference images to learn a common pattern, specifically the *Averaging* [44] and the *Surrogate* Attack [17, 32]. As shown in our baseline comparison in Sec. C, these attacks are no longer effective if Gaussian Shading is implemented securely. In all our experiments, we draw new nonces and keys for every generated image.

B. Example Images and Image Quality Trade-off

We provide image examples to give more intuition on the attacks and the resulting visual quality. Note that all image examples are of size 512×512, and the attacker's proxy model is SD2.1. We examine different aspects in the following figures:

- Fig. 12 shows the progression of the Imprint-Forgery attack with respect to the number of optimization steps.
- Fig. 13 shows successful examples of the Imprint-Forgery attack on different target models and both watermarking approaches (Tree-Ring and Gaussian Shading).
- Fig. 14 depicts successful examples of the Imprint-Removal attack.
- Fig. 15 shows successful examples of the Reprompting attack.
- Finally, Fig. 16 shows the trade-off between detection rate by the target model and image quality for four combinations of settings: Imprint-Forgery & Imprint-Removal, and Tree-Ring & Gaussian Shading.

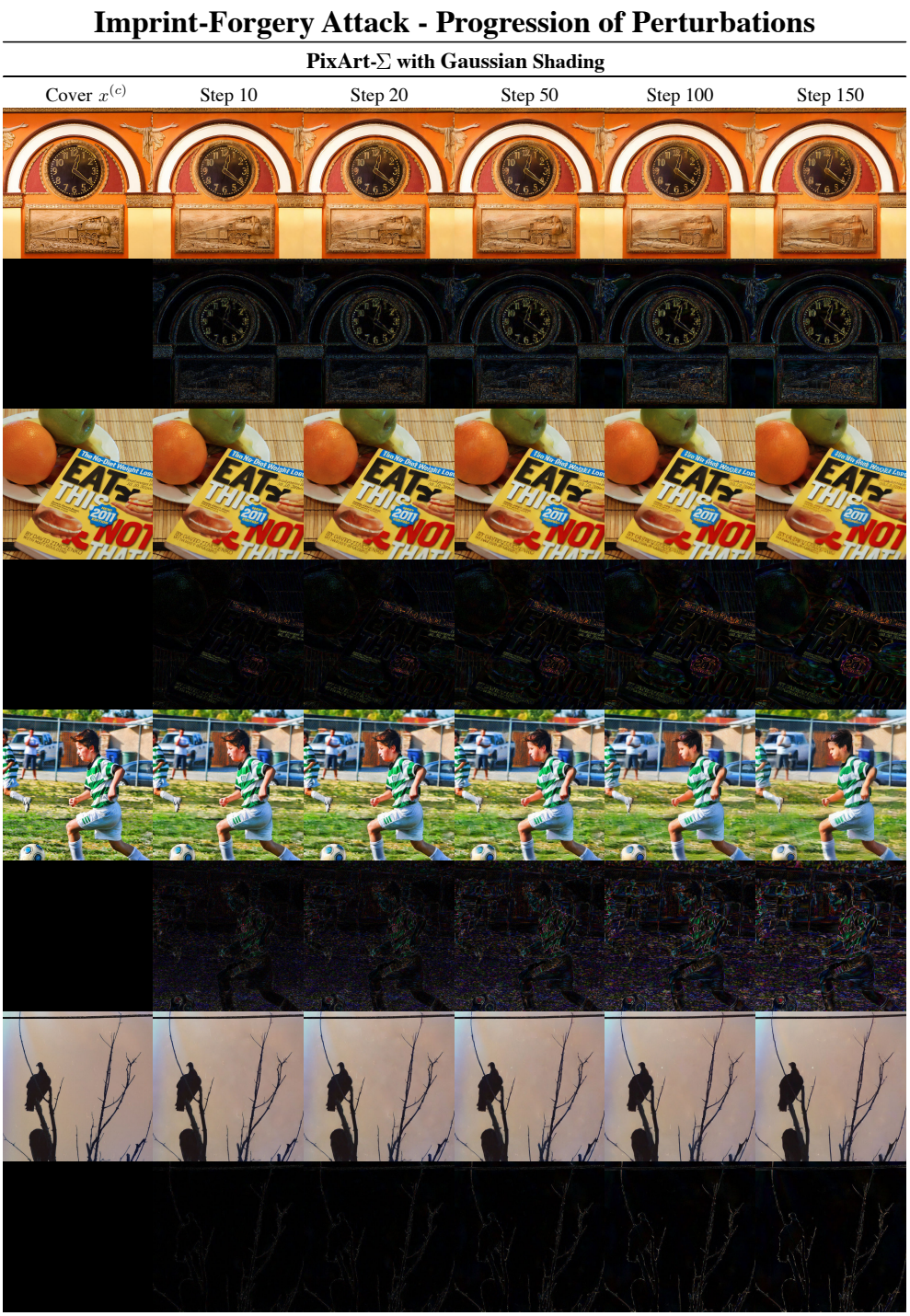


Figure 12. Progression of our Imprint-Forgery attack with respect to optimization steps. The target model is PixArt-Σ, and the forged watermark is Gaussian Shading. Results are presented for four cover images $x^{(c)}$. For each cover image, the top row shows the initial cover and then the attack versions after different numbers of optimization steps. The bottom row illustrates the absolute pixel difference between the attack image and its initial cover version. *Observation:* The imprinting procedure is targeting edges that carry semantic meaning and reflect unique features in the initial watermarked latent $z_T^{(w)}$ which we are optimizing towards. Some images with small objects defined by fine characteristics are noticeably changed (small text, small faces), while other features such as the outline of dark objects on bright backgrounds remain almost indistinguishable. The removal attack introduces very similar changes to the images. Note that such critical changes can be avoided simply by using masks during optimization, as described in Sec. 3.

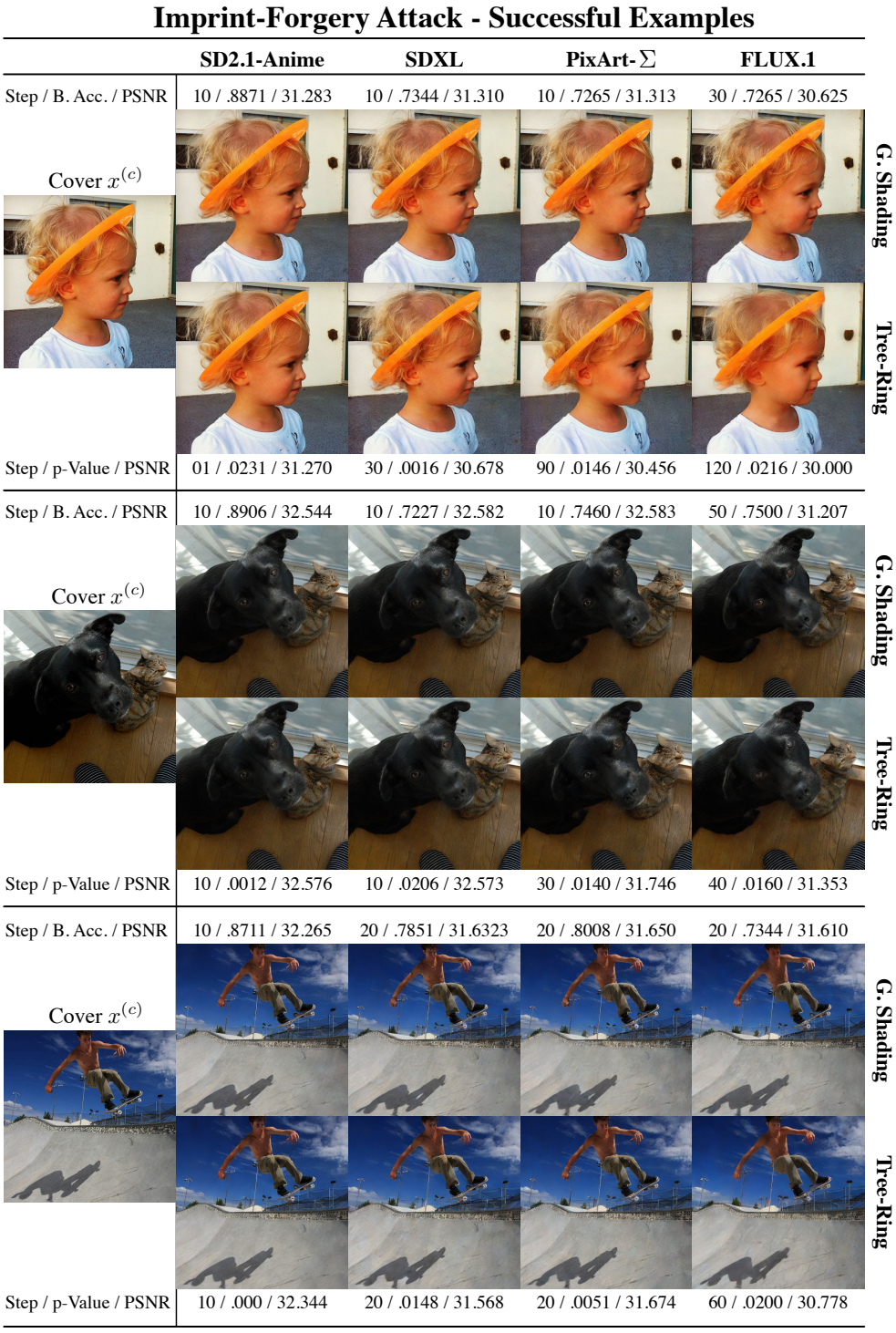


Figure 13. Examples of our Imprint-Forgery attack on different target models and both watermark approaches. For each cover image $x^{(c)}$, during the different optimization steps, we show the first successful attack image that passes the detection threshold τ of Gaussian Shading with an FPR of 10^{-6} (top row) or Tree-Ring with an FPR of 1% (bottom row). For each image, we report the step, the bit accuracy (for Gaussian Shading) or the p-value (for Tree-Ring), and the PSNR between cover images and attack example. *Observation:* Attacks against PixArt-Σ and FLUX.1 usually have stronger visual perturbations, reflecting a higher imprinting difficulty.

Figure 14. Examples of our watermark removal attack on different target models and both watermark approaches. For each target model (in the different columns), the 1st row shows the generated, watermarked image $x^{(w)}$, the 2nd row the attack example, and the 3rd row the absolute pixel difference between the two. Each attack example is the first one during generation to pass the detection threshold τ of Gaussian Shading (FPR=10^{-6}) or Tree-Ring (FPR=1%). We report the step, the bit accuracy (for Gaussian Shading) or the p-value (for Tree-Ring), and the PSNR between the original and the attack examples. *Observation:* Attacks against FLUX.1 generally display weaker visual perturbations, reflecting a lower removal difficulty.

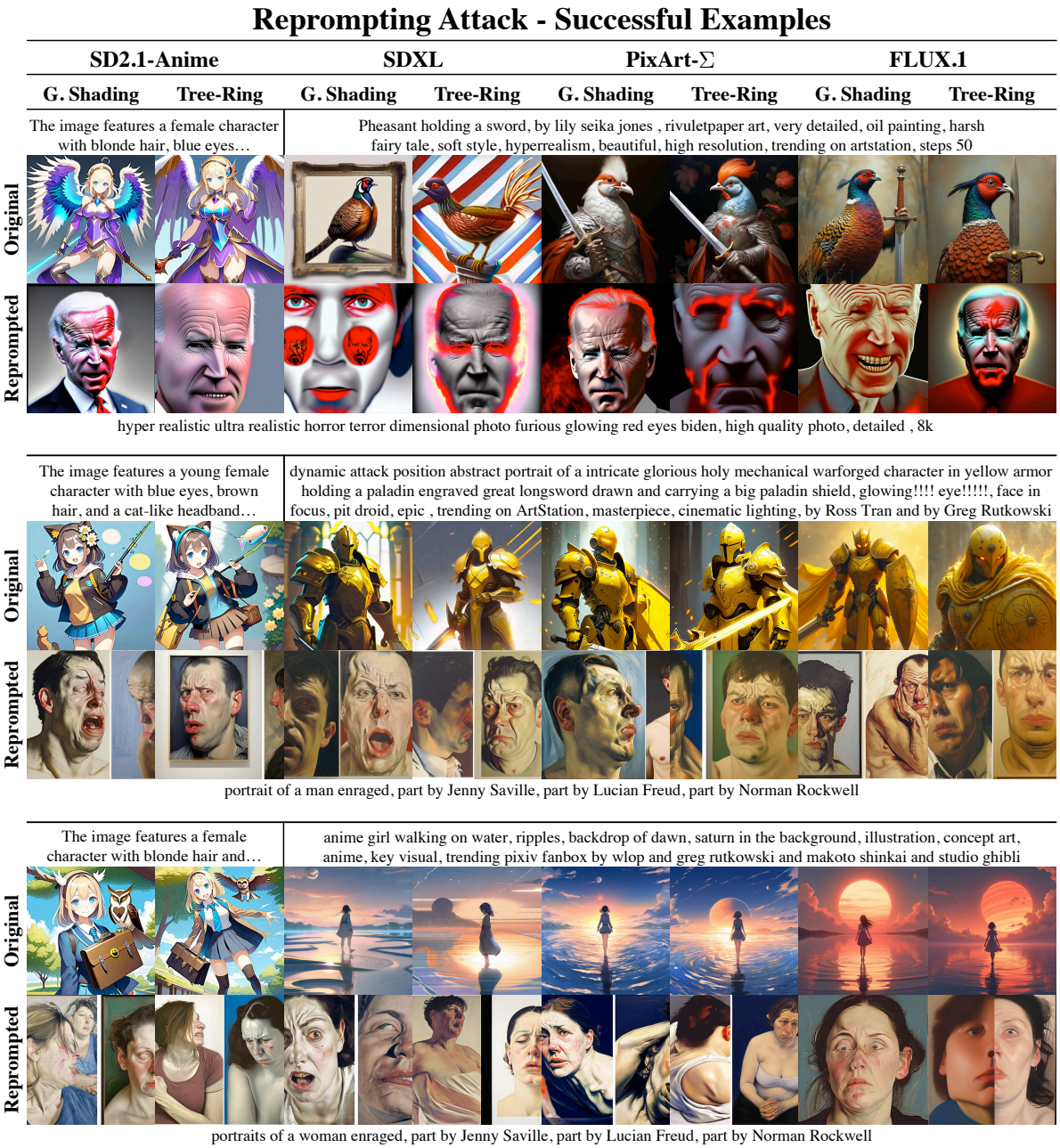
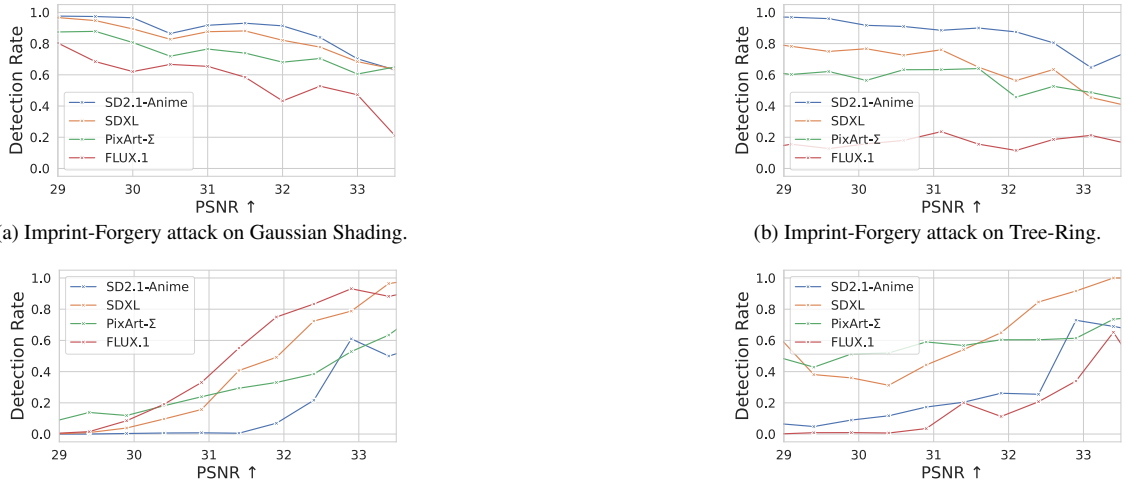


Figure 15. Examples of the Reprompting attack on different target models and both watermarking approaches. In each section, the top row shows original images which carry a semantic watermark generated with prompts from the Stable-Diffusion-Prompts dataset using the respective target model indicated at the top of the figure. The bottom row shows successful attack instances which our Reprompting attack generated from each original reference image (using SD2.1 as proxy model) using prompts from the Inappropriate Image Prompts (I2P) dataset. Note that we tried to choose less disturbing prompts (without blood, violence, or nudity) for this figure. *Observation:* These results illustrate how an attacker can potentially generate arbitrary harmful images (within the limits of the proxy model) with semantic watermarks spoofed from retrieved watermarked images to appear as generated by a service or user using the watermarking method.



(a) Imprint-Forgery attack on Gaussian Shading.

(b) Imprint-Forgery attack on Tree-Ring.

(c) Imprint-Removal attack on Gaussian Shading.

(d) Imprint-Removal attack on Tree-Ring.

Figure 16. Trade-off between attack success and image quality. To this end, we plot the detection rate vs. PSNR for watermark forgery (top) and removal (bottom) for both watermark approaches (left and right) on different target models. An attacker aims at a higher detection rate in the forgery case, and at a lower detection rate in the removal case. For Gaussian Shading (left), the single-bit detection accuracy is reported with a threshold τ corresponding to an FPR of 10^{-6}. For Tree-Ring (right), the detection accuracy is reported with a threshold corresponding to an FPR of 1%. PSNR is computed between the attack image and its initial starting image: the cover image for forgery, and watermarked image for removal. *Observation:* In the case of forgery, we observe a consistent trend where the detection rate increases as image quality decreases. For removal, the detection rate tends to decline rapidly as PSNR decreases.

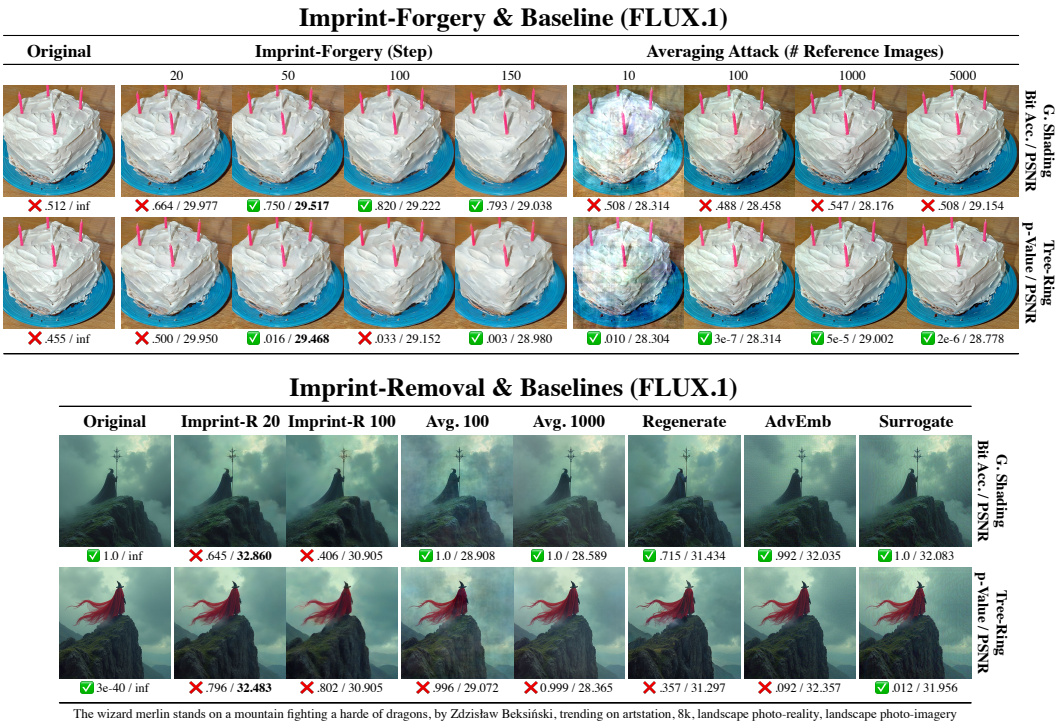


Figure 17. Visual comparison of our Imprint attacks against baselines for watermark forgery (top) and removal (bottom) against a service provider using FLUX.1. This model is the one against which most of the baseline attacks are relatively effective, while our attacks have higher difficulty because we use SD2.1 as proxy model, which is very dissimilar to FLUX.1. Green check marks indicate successful detection of a watermark while red crossed indicate no detection (against threshold τ of Gaussian Shading (FPR=10^{-6}) or Tree-Ring (FPR=1%)). Imprint-Forgery and Imprint-Removal quickly achieve forgery/removal success while generally preserving high PSNR.

C. Full Experimental Results and Baselines

Here we compare our attacks with related work.

Baselines. We use the *Averaging Attack* [44] as baseline for watermark forgery, and the *Averaging* [44], *Regeneration* [49], *Adversarial Embedding (AdvEmb)* [2], and *Surrogate* [32] Attacks as baselines for watermark removal. As implementation, we use the provided Github repositories[1]. Similar to the main attack experiments via imprinting, the forgery baselines are evaluated using 100 *MS-COCO-2017* [21] cover images and the removal baselines are evaluated on 100 watermarked images generated using the *Stable-Diffusion-Prompts* dataset. We use the strongest attack settings available. Our setup is as follows:

- The *Averaging Attack* [44] is applicable for forgery and removal. To obtain sets of averaged images, we randomly draw 30 different sets of size 10, 100, 1k, and 5k from a set of 10k watermarked images which are then added to cover images (forgery) or subtracted from generated images (removal).
- The *Regeneration Attack* [49] is deployed with maximum settings (i.e. 200 noising steps) using SD2.1.
- The *Adversarial Embedding (AdvEmb) Attack* [2] is deployed with maximum settings as well, i.e. a L_∞-perturbation budget of $\frac{8}{255}$, using the auto-encoder of the SD2.1 model for 200 optimization steps.
- For the *Surrogate Attack* [32], we follow the original training procedure. In particular, we train a binary classifier (ResNet-18) with 7500 real images from the ImageNet validation dataset [31] as negative class and 7500 watermarked images generated with each respective model using randomly drawn ImageNet class names as positive class. Optimization is performed with the maximum setting for the L_∞-perturbation budget of $\frac{8}{255}$.

Results and Comparison. Tab. 4 shows detailed results for watermark forgery while Tab. 5 shows the outcomes for watermark removal. In addition, Fig. 17 compares our attacks with all baselines *visually*.

1) Watermark Forgery. For Tree-Ring, the Averaging Attack performs similarly to our attack on three out of four target models. With FLUX.1 as target model, the baseline outperforms our attack in terms of detection rate (of the forged watermarks). To avoid visible distortions, however, this attack requires a large number of reference images with the same watermark pattern. The cake example in Fig. 17 shows that 10 reference images lead to visible distortions. We further note that this large number of references images is a rather strong assumption. In contrast, our attack has no strong distortions and requires only one reference image. For FLUX.1, we can still achieve up to 35% forgery rate in our realistic setup.

Regarding Gaussian Shading, the Averaging Attack fails in all cases. This is in contrast to reported results in [44]. We attribute this discrepancy to an incorrect implementation of Gaussian Shading in the initial evaluation of the Averaging Attack, where the key and nonce of Gaussian Shading were reused. If Gaussian Shading is used correctly, the attack fails as expected. We explain this issue in more detail in Sec. A.

2) Watermark Removal. For Tree-Ring, the Averaging Attack is the only removal attack that consistently works on all four target models—again with the assumption to have a large number of images with the same watermark key. For Gaussian Shading, our proposed attacks are the only attacks that are able to remove the watermarks.

Note that Gunn et al. [17] demonstrated that the Surrogate Attack is effective against Gaussian Shading. Like the Averaging Attack, this success relied on an incorrect instantiation of Gaussian Shading where key and nonce were re-used. Moreover, the Surrogate Attack also fails for Tree-Ring in the realistic setting where the negative class consists of real images (in contrast to unwatermarked images obtained by the same service). This is in line with the observation by An et al. [2].

Summary. In terms of watermark forgery and removal, our attacks outperform or perform similarly to previously proposed methods while only requiring one watermarked target image. Importantly, we observe that previous attacks fail on Gaussian Shading, whereas our proposed attacks are effective.

[1]The Github Repositories for the Averaging (https://github.com/showlab/watermark-steganalysis), Regeneration (https://github.com/XuandongZhao/WatermarkAttacker), AdvEmb (https://github.com/umd-huang-lab/WAVES), and Surrogate (https://github.com/umd-huang-lab/WAVES) attacks.

Model	Attk	Ref.Img.	Step	Gaussian Shading (FPR=10^{-6}) Bit Acc.	Det.	Attr.	PSNR	Tree-Ring (FPR=1%) p-Value	Det.	PSNR
SD2.1-Anime	Cover	-	-	$0.50_{\pm 0.03}$	0.00	0.00	inf	$4.97 \times 10^{-1}{}_{\pm 2.54 \times 10^{-1}}$	0.00	inf
	Imprint-F	1	20	$0.93_{\pm 0.04}$	1.00	1.00	$30.84_{1.59}$	$4.74 \times 10^{-3}{}_{\pm 1.81 \times 10^{-2}}$	0.94	$30.85_{1.56}$
		1	50	$0.97_{\pm 0.02}$	1.00	1.00	$30.35_{1.40}$	$1.83 \times 10^{-4}{}_{\pm 8.89 \times 10^{-4}}$	1.00	$30.35_{1.36}$
		1	100	$0.98_{\pm 0.02}$	1.00	1.00	$30.00_{1.27}$	$6.59 \times 10^{-6}{}_{\pm 4.87 \times 10^{-5}}$	1.00	$29.99_{1.22}$
		1	150	$0.99_{\pm 0.01}$	1.00	1.00	$29.78_{1.19}$	$1.45 \times 10^{-6}{}_{\pm 1.25 \times 10^{-5}}$	1.00	$29.77_{1.13}$
	Reprompt	1	-	$0.92_{\pm 0.07}$	0.98	0.98	-	$1.40 \times 10^{-2}{}_{\pm 5.45 \times 10^{-2}}$	0.90	-
	Reprompt+	1	-	$1.00_{\pm 0.01}$	1.00	1.00	-	$3.83 \times 10^{-5}{}_{\pm 1.69 \times 10^{-4}}$	1.00	-
	Averaging	10	0	$0.50_{\pm 0.03}$	0.00	0.00	$28.34_{0.25}$	$2.65 \times 10^{-4}{}_{\pm 1.13 \times 10^{-3}}$	1.00	$28.32_{0.27}$
		100	0	$0.50_{\pm 0.03}$	0.00	0.00	$30.06_{0.41}$	$4.46 \times 10^{-9}{}_{\pm 3.70 \times 10^{-8}}$	1.00	$29.57_{0.39}$
		1000	0	$0.50_{\pm 0.03}$	0.00	0.00	$32.18_{0.81}$	$3.13 \times 10^{-10}{}_{\pm 3.09 \times 10^{-9}}$	1.00	$30.56_{0.45}$
		5000	0	$0.50_{\pm 0.03}$	0.00	0.00	$32.84_{0.46}$	$4.67 \times 10^{-10}{}_{\pm 4.66 \times 10^{-9}}$	1.00	$30.72_{0.32}$
SDXL	Cover	-	-	$0.50_{\pm 0.03}$	0.00	0.00	inf	$4.72 \times 10^{-1}{}_{\pm 2.47 \times 10^{-1}}$	0.01	inf
	Imprint-F	1	20	$0.79_{\pm 0.04}$	0.99	0.99	$30.83_{1.60}$	$9.63 \times 10^{-2}{}_{\pm 1.30 \times 10^{-1}}$	0.43	$30.82_{1.55}$
		1	50	$0.87_{\pm 0.03}$	1.00	1.00	$30.33_{1.41}$	$2.70 \times 10^{-2}{}_{\pm 5.44 \times 10^{-2}}$	0.72	$30.32_{1.35}$
		1	100	$0.91_{\pm 0.02}$	1.00	1.00	$29.99_{1.29}$	$6.89 \times 10^{-3}{}_{\pm 2.82 \times 10^{-2}}$	0.95	$29.97_{1.24}$
		1	150	$0.93_{\pm 0.03}$	1.00	1.00	$29.78_{1.22}$	$5.43 \times 10^{-3}{}_{\pm 3.67 \times 10^{-2}}$	0.99	$29.77_{1.18}$
	Reprompt	1	-	$0.91_{\pm 0.06}$	0.99	0.99	-	$5.20 \times 10^{-3}{}_{\pm 3.46 \times 10^{-2}}$	0.97	-
	Reprompt+	1	-	$0.95_{\pm 0.02}$	1.00	1.00	-	$2.57 \times 10^{-4}{}_{\pm 1.73 \times 10^{-3}}$	1.00	-
	Averaging	10	0	$0.50_{\pm 0.03}$	0.00	0.00	$28.25_{0.36}$	$3.25 \times 10^{-4}{}_{\pm 1.94 \times 10^{-3}}$	1.00	$28.22_{0.30}$
		100	0	$0.50_{\pm 0.03}$	0.00	0.00	$28.55_{0.56}$	$3.49 \times 10^{-6}{}_{\pm 3.49 \times 10^{-5}}$	1.00	$28.46_{0.40}$
		1000	0	$0.50_{\pm 0.03}$	0.00	0.00	$27.85_{0.40}$	$3.99 \times 10^{-8}{}_{\pm 2.72 \times 10^{-7}}$	1.00	$28.21_{0.31}$
		5000	0	$0.50_{\pm 0.03}$	0.00	0.00	$27.64_{0.37}$	$3.28 \times 10^{-8}{}_{\pm 2.16 \times 10^{-7}}$	1.00	$28.09_{0.30}$
PixArt-Σ	Cover	-	-	$0.50_{\pm 0.03}$	0.00	0.00	inf	$4.74 \times 10^{-1}{}_{\pm 2.63 \times 10^{-1}}$	0.01	inf
	Imprint-F	1	20	$0.74_{\pm 0.07}$	0.72	0.72	$30.84_{1.59}$	$1.19 \times 10^{-1}{}_{\pm 1.67 \times 10^{-1}}$	0.29	$30.84_{1.57}$
		1	50	$0.82_{\pm 0.07}$	0.91	0.91	$30.34_{1.40}$	$5.02 \times 10^{-2}{}_{\pm 9.19 \times 10^{-2}}$	0.55	$30.34_{1.38}$
		1	100	$0.86_{\pm 0.08}$	0.94	0.94	$30.00_{1.28}$	$1.98 \times 10^{-2}{}_{\pm 5.38 \times 10^{-2}}$	0.78	$29.99_{1.26}$
		1	150	$0.88_{\pm 0.08}$	0.96	0.96	$29.78_{1.21}$	$1.08 \times 10^{-2}{}_{\pm 3.65 \times 10^{-2}}$	0.84	$29.78_{1.19}$
	Reprompt	1	-	$0.88_{\pm 0.09}$	0.93	0.93	-	$1.64 \times 10^{-2}{}_{\pm 6.91 \times 10^{-2}}$	0.88	-
	Reprompt+	1	-	$0.92_{\pm 0.06}$	1.00	1.00	-	$1.54 \times 10^{-3}{}_{\pm 1.03 \times 10^{-2}}$	0.99	-
	Averaging	10	0	$0.50_{\pm 0.03}$	0.00	0.00	$28.08_{0.47}$	$3.73 \times 10^{-2}{}_{\pm 8.14 \times 10^{-2}}$	0.69	$28.06_{0.46}$
		100	0	$0.50_{\pm 0.03}$	0.00	0.00	$27.95_{0.55}$	$1.35 \times 10^{-2}{}_{\pm 4.87 \times 10^{-2}}$	0.85	$27.96_{0.53}$
		1000	0	$0.50_{\pm 0.03}$	0.00	0.00	$27.95_{0.53}$	$1.14 \times 10^{-2}{}_{\pm 5.22 \times 10^{-2}}$	0.91	$27.94_{0.52}$
		5000	0	$0.49_{\pm 0.03}$	0.00	0.00	$27.94_{0.52}$	$1.18 \times 10^{-2}{}_{\pm 6.07 \times 10^{-2}}$	0.93	$27.94_{0.52}$
FLUX.1	Cover	-	-	$0.49_{\pm 0.04}$	0.00	0.00	inf	$4.99 \times 10^{-1}{}_{\pm 2.65 \times 10^{-1}}$	0.00	inf
	Imprint-F	1	20	$0.67_{\pm 0.05}$	0.26	0.26	$30.94_{1.60}$	$2.69 \times 10^{-1}{}_{\pm 2.19 \times 10^{-1}}$	0.07	$30.84_{1.61}$
		1	50	$0.74_{\pm 0.05}$	0.66	0.66	$30.46_{1.41}$	$2.62 \times 10^{-1}{}_{\pm 2.42 \times 10^{-1}}$	0.12	$30.35_{1.43}$
		1	100	$0.78_{\pm 0.06}$	0.86	0.86	$30.11_{1.28}$	$1.92 \times 10^{-1}{}_{\pm 2.05 \times 10^{-1}}$	0.20	$30.02_{1.31}$
		1	150	$0.81_{\pm 0.05}$	0.95	0.95	$29.88_{1.19}$	$1.80 \times 10^{-1}{}_{\pm 2.22 \times 10^{-1}}$	0.23	$29.81_{1.23}$
	Reprompt	1	-	$0.74_{\pm 0.08}$	0.66	0.66	-	$2.37 \times 10^{-1}{}_{\pm 2.30 \times 10^{-1}}$	0.13	-
	Reprompt+	1	-	$0.79_{\pm 0.06}$	0.88	0.88	-	$8.99 \times 10^{-2}{}_{\pm 1.09 \times 10^{-1}}$	0.35	-
	Averaging	10	0	$0.50_{\pm 0.03}$	0.00	0.00	$28.30_{0.50}$	$6.88 \times 10^{-2}{}_{\pm 1.47 \times 10^{-1}}$	0.63	$28.39_{0.46}$
		100	0	$0.50_{\pm 0.03}$	0.00	0.00	$29.24_{0.78}$	$1.84 \times 10^{-2}{}_{\pm 5.62 \times 10^{-2}}$	0.83	$29.18_{0.76}$
		1000	0	$0.50_{\pm 0.03}$	0.00	0.00	$29.31_{0.66}$	$9.05 \times 10^{-3}{}_{\pm 2.90 \times 10^{-2}}$	0.90	$29.18_{0.63}$
		5000	0	$0.50_{\pm 0.03}$	0.00	0.00	$29.22_{0.50}$	$9.31 \times 10^{-3}{}_{\pm 3.15 \times 10^{-2}}$	0.90	$29.01_{0.52}$

Table 4. Full experimental results on forgery attacks. *Ref.Img.* and *Step* refer to the number of reference images and the attack step used, respectively. *Bit Acc.* refers to Gaussian Shading bit accuracy. *Det.* and *Att.* refer to mean detection and attribution success. *Imprint-F* and *Reprompt/Reprompt+* refer to our Imprint-Forgery and Reprompting/Reprompt+ attacks, respectively. The Averaging attack [44] is tested with varying amounts of reference images.

Model	Attack	Ref.Img.	Str.	Step	Gaussian Shading (FPR=10^{-6}) Bit Acc.	Det.	Attr.	PSNR	Tree-Ring (FPR=1%) p-Value	Det.	PSNR
SD2.1-Anime	Original	-	-	-	$1.00_{\pm 0.00}$	1.00	1.00	inf	$4.29 \times 10^{-17}{}_{\pm 3.41 \times 10^{-16}}$	1.00	inf
	Imprint-R	1	-	20	$0.45_{\pm 0.13}$	0.04	0.04	$32.05_{0.76}$	$2.39 \times 10^{-1}{}_{\pm 3.16 \times 10^{-1}}$	0.50	$32.15_{0.73}$
		1	-	50	$0.16_{\pm 0.07}$	0.00	0.00	$31.08_{0.71}$	$6.15 \times 10^{-1}{}_{\pm 4.00 \times 10^{-1}}$	0.14	$31.12_{0.67}$
		1	-	100	$0.05_{\pm 0.03}$	0.00	0.00	$30.49_{0.67}$	$8.06 \times 10^{-1}{}_{\pm 3.29 \times 10^{-1}}$	0.04	$30.45_{0.63}$
		1	-	150	$0.02_{\pm 0.01}$	0.00	0.00	$30.11_{0.65}$	$8.79 \times 10^{-1}{}_{\pm 2.73 \times 10^{-1}}$	0.03	$30.02_{0.58}$
	Averaging	10	-	0	$0.91_{\pm 0.05}$	1.00	1.00	$28.47_{0.24}$	$8.15 \times 10^{-1}{}_{\pm 3.01 \times 10^{-1}}$	0.05	$28.41_{0.22}$
		100	-	0	$0.97_{\pm 0.03}$	1.00	1.00	$30.06_{0.46}$	$9.75 \times 10^{-1}{}_{\pm 1.19 \times 10^{-1}}$	0.01	$29.54_{0.36}$
		1000	-	0	$0.99_{\pm 0.01}$	1.00	1.00	$31.56_{0.82}$	$9.96 \times 10^{-1}{}_{\pm 3.31 \times 10^{-2}}$	0.00	$30.35_{0.41}$
		5000	-	0	$1.00_{\pm 0.00}$	1.00	1.00	$31.99_{0.40}$	$9.97 \times 10^{-1}{}_{\pm 2.32 \times 10^{-2}}$	0.00	$30.46_{0.19}$
	Regen.	1	-	200	$0.92_{\pm 0.04}$	1.00	1.00	$31.00_{0.85}$	$4.43 \times 10^{-3}{}_{\pm 1.14 \times 10^{-2}}$	0.95	$31.08_{0.84}$
	AdvEmb	1	8	200	$0.86_{\pm 0.06}$	0.99	0.99	$31.30_{0.19}$	$6.41 \times 10^{-3}{}_{\pm 2.96 \times 10^{-2}}$	0.94	$31.27_{0.16}$
	Surrogate	7500	8	200	$0.96_{\pm 0.03}$	1.00	1.00	$32.04_{0.17}$	$1.54 \times 10^{-3}{}_{\pm 1.18 \times 10^{-2}}$	0.99	$32.02_{0.16}$
SDXL	Original	-	-	-	$1.00_{\pm 0.00}$	1.00	1.00	inf	$1.10 \times 10^{-21}{}_{\pm 1.10 \times 10^{-20}}$	1.00	inf
	Imprint-R	1	-	20	$0.75_{\pm 0.10}$	0.66	0.66	$31.50_{0.82}$	$2.51 \times 10^{-4}{}_{\pm 1.02 \times 10^{-3}}$	1.00	$32.04_{1.08}$
		1	-	50	$0.46_{\pm 0.09}$	0.00	0.00	$30.50_{0.66}$	$7.75 \times 10^{-2}{}_{\pm 1.56 \times 10^{-1}}$	0.64	$30.89_{0.87}$
		1	-	100	$0.26_{\pm 0.06}$	0.00	0.00	$29.97_{0.60}$	$3.13 \times 10^{-1}{}_{\pm 3.28 \times 10^{-1}}$	0.33	$30.22_{0.76}$
		1	-	150	$0.19_{\pm 0.05}$	0.00	0.00	$29.70_{0.58}$	$4.54 \times 10^{-1}{}_{\pm 3.44 \times 10^{-1}}$	0.12	$29.90_{0.72}$
	Averaging	10	-	0	$0.90_{\pm 0.06}$	1.00	1.00	$28.25_{0.33}$	$5.28 \times 10^{-1}{}_{\pm 3.61 \times 10^{-1}}$	0.17	$28.08_{0.17}$
		100	-	0	$0.97_{\pm 0.03}$	1.00	1.00	$28.54_{0.51}$	$8.90 \times 10^{-1}{}_{\pm 2.55 \times 10^{-1}}$	0.04	$28.29_{0.33}$
		1000	-	0	$1.00_{\pm 0.01}$	1.00	1.00	$27.78_{0.37}$	$9.58 \times 10^{-1}{}_{\pm 1.68 \times 10^{-1}}$	0.01	$28.05_{0.15}$
		5000	-	0	$1.00_{\pm 0.00}$	1.00	1.00	$27.49_{0.34}$	$9.71 \times 10^{-1}{}_{\pm 1.32 \times 10^{-1}}$	0.01	$27.98_{0.13}$
	Regen.	1	-	200	$0.85_{\pm 0.05}$	0.99	0.99	$30.50_{0.77}$	$3.41 \times 10^{-4}{}_{\pm 1.59 \times 10^{-3}}$	1.00	$30.45_{0.61}$
	AdvEmb	1	8	200	$0.96_{\pm 0.03}$	1.00	1.00	$31.36_{0.25}$	$1.10 \times 10^{-3}{}_{\pm 6.21 \times 10^{-3}}$	0.99	$31.41_{0.28}$
	Surrogate	7500	8	200	$0.97_{\pm 0.03}$	1.00	1.00	$32.17_{0.30}$	$5.89 \times 10^{-5}{}_{\pm 4.45 \times 10^{-4}}$	1.00	$32.38_{0.25}$
PixArt-Σ	Original	-	-	-	$1.00_{\pm 0.02}$	1.00	1.00	inf	$1.22 \times 10^{-6}{}_{\pm 1.21 \times 10^{-5}}$	1.00	inf
	Imprint-R	1	-	20	$0.86_{\pm 0.08}$	0.94	0.94	$32.20_{1.40}$	$5.59 \times 10^{-3}{}_{\pm 3.20 \times 10^{-2}}$	0.96	$32.39_{1.46}$
		1	-	50	$0.64_{\pm 0.08}$	0.18	0.18	$31.14_{1.17}$	$4.37 \times 10^{-2}{}_{\pm 1.36 \times 10^{-1}}$	0.75	$31.22_{1.22}$
		1	-	100	$0.40_{\pm 0.07}$	0.00	0.00	$30.49_{1.04}$	$1.94 \times 10^{-1}{}_{\pm 2.69 \times 10^{-1}}$	0.41	$30.49_{1.07}$
		1	-	150	$0.27_{\pm 0.07}$	0.00	0.00	$30.14_{0.99}$	$3.55 \times 10^{-1}{}_{\pm 3.15 \times 10^{-1}}$	0.14	$30.10_{1.00}$
	Averaging	10	-	0	$0.92_{\pm 0.07}$	0.99	0.99	$28.07_{0.40}$	$1.11 \times 10^{-1}{}_{\pm 2.23 \times 10^{-1}}$	0.52	$28.05_{0.46}$
		100	-	0	$0.97_{\pm 0.06}$	0.99	0.99	$27.90_{0.41}$	$2.82 \times 10^{-1}{}_{\pm 3.57 \times 10^{-1}}$	0.42	$27.91_{0.44}$
		1000	-	0	$0.98_{\pm 0.05}$	1.00	1.00	$27.89_{0.42}$	$5.94 \times 10^{-1}{}_{\pm 4.20 \times 10^{-1}}$	0.20	$27.89_{0.45}$
		5000	-	0	$0.99_{\pm 0.03}$	1.00	1.00	$27.89_{0.41}$	$6.76 \times 10^{-1}{}_{\pm 4.17 \times 10^{-1}}$	0.19	$27.88_{0.45}$
	Regen.	1	-	200	$0.87_{\pm 0.06}$	0.97	0.97	$30.89_{1.08}$	$6.40 \times 10^{-3}{}_{\pm 3.50 \times 10^{-2}}$	0.94	$30.93_{1.13}$
	AdvEmb	1	8	200	$0.94_{\pm 0.06}$	0.99	0.99	$31.53_{0.29}$	$1.09 \times 10^{-2}{}_{\pm 6.85 \times 10^{-2}}$	0.92	$31.50_{0.29}$
	Surrogate	7500	8	200	$0.95_{\pm 0.06}$	1.00	1.00	$31.82_{0.26}$	$2.12 \times 10^{-3}{}_{\pm 1.32 \times 10^{-2}}$	0.97	$31.79_{0.21}$
FLUX.1	Original	-	-	-	$1.00_{\pm 0.00}$	1.00	1.00	inf	$7.92 \times 10^{-6}{}_{\pm 7.92 \times 10^{-5}}$	1.00	inf
	Imprint-R	1	-	20	$0.74_{\pm 0.06}$	0.75	0.75	$31.19_{1.02}$	$4.00 \times 10^{-1}{}_{\pm 3.16 \times 10^{-1}}$	0.07	$31.47_{0.84}$
		1	-	50	$0.54_{\pm 0.05}$	0.00	0.00	$30.31_{0.89}$	$4.99 \times 10^{-1}{}_{\pm 2.97 \times 10^{-1}}$	0.00	$30.70_{0.73}$
		1	-	100	$0.39_{\pm 0.04}$	0.00	0.00	$29.73_{0.81}$	$4.79 \times 10^{-1}{}_{\pm 2.62 \times 10^{-1}}$	0.00	$30.11_{0.68}$
		1	-	150	$0.34_{\pm 0.05}$	0.00	0.00	$29.41_{0.76}$	$6.18 \times 10^{-1}{}_{\pm 2.43 \times 10^{-1}}$	0.00	$29.79_{0.65}$
	Averaging	10	-	0	$0.99_{\pm 0.02}$	1.00	1.00	$28.19_{0.42}$	$3.00 \times 10^{-1}{}_{\pm 3.55 \times 10^{-1}}$	0.36	$28.29_{0.41}$
		100	-	0	$1.00_{\pm 0.01}$	1.00	1.00	$29.22_{0.81}$	$5.94 \times 10^{-1}{}_{\pm 4.31 \times 10^{-1}}$	0.22	$29.14_{0.67}$
		1000	-	0	$1.00_{\pm 0.00}$	1.00	1.00	$29.42_{0.65}$	$7.48 \times 10^{-1}{}_{\pm 3.80 \times 10^{-1}}$	0.12	$29.31_{0.65}$
		5000	-	0	$1.00_{\pm 0.00}$	1.00	1.00	$29.35_{0.42}$	$7.71 \times 10^{-1}{}_{\pm 3.73 \times 10^{-1}}$	0.11	$29.15_{0.53}$
	Regen.	1	-	200	$0.72_{\pm 0.05}$	0.64	0.64	$30.89_{1.08}$	$2.58 \times 10^{-1}{}_{\pm 2.39 \times 10^{-1}}$	0.13	$31.04_{1.15}$
	AdvEmb	1	8	200	$1.00_{\pm 0.02}$	1.00	1.00	$31.38_{0.47}$	$5.13 \times 10^{-2}{}_{\pm 9.55 \times 10^{-2}}$	0.66	$31.44_{0.26}$
	Surrogate	7500	8	200	$0.99_{\pm 0.02}$	1.00	1.00	$31.79_{0.59}$	$4.79 \times 10^{-2}{}_{\pm 1.26 \times 10^{-1}}$	0.75	$31.79_{0.57}$

Table 5. Full experimental results on removal attacks. *Ref.Img.*, *Str.* and *Step* refer to the number of reference images, the strength setting, and the attack step used, respectively. *Bit Acc.* refers to Gaussian Shading bit accuracy. *Det.* and *Att.* refer to mean detection and attribution success. *Imprint-R* refers to our Imprint-Removal attack. The Averaging [44] attack is tested with varying amounts of reference images. The Regeneration [49] (*Regen.*), AdvEmb [2], and Surrogate [32] attacks are set to the strongest settings available in their corresponding Github repositories.

D. Experimental Details

Here, we provide more details on our experimental setup, including the datasets used, details on our finetune of Stable Diffusion 2.1 (SD2.1-Anime), the number of samples used in each experiment, the watermark parameters, and the runtime of our attacks.

D.1. Attacker and Target Models

Our default setup is to use Stable Diffusion 2.1 as the attacker model, and SDXL, PixArt-Σ, FLUX.1, and our SD2.1-Anime model as the target models. Our transferability analysis (Sec. 4.5) is the exception where we test multiple combinations of attacker and target models. Tab. 6 provides an overview of all the models used in our experiments, including the used scheduler and other relevant settings. The choice of schedulers for each model is based on the default setting in the corresponding pipelines from the Huggingface diffusers library.

							G. Shad. Params		
Model	Huggingface ID	Type	L. Ch.	Scheduler	Steps	G. Scale	ℓ	ρ	k
SD1.5	runwayml/stable-diffusion-v1-5	UNet	4	DDIM	50	7.5	1	64	256
SD2.1	stabilityai/stable-diffusion-2-1-base	UNet	4	DDIM	50	7.5	1	64	256
SD2.1-Anime	stabilityai/stable-diffusion-2-1-base	UNet	4	DDIM	50	7.5	1	64	256
SDXL	stabilityai/stable-diffusion-xl-base-1.0	UNet	4	DDIM	50	7.5	1	64	256
PixArt-Σ	PixArt-alpha/PixArt-Sigma-XL-2-512-MS	DiT	4	DPM	50	7.5	1	64	256
FLUX.1	black-forest-labs/FLUX.1-dev	DiT	16	FlowMatchEuler	20	3.5	1	256	256
Waifu Diffusion	hakurei/waifu-diffusion	UNet	4	DDIM	50	7.5	1	64	256
Mitsua Diffusion One	Mitsua/mitsua-diffusion-one	UNet	4	DDIM	50	7.5	1	64	256
Common Canvas	common-canvas/CommonCanvas-S-C	UNet	4	DDIM	50	7.5	1	64	256

Table 6. Overview of model settings. The settings outlined here are used across all experiments without deviation for both the case that the model at hand is a target model, or the case that it is the proxy model used by the attacker. *L. Ch.* refers to the number of latent channels. *Scheduler* refers to the scheduler during image generation and inversion. *Steps* refers to the number of denoising & inversion steps during generation and inversion. *G. Scale* refers to the guidance scale during generation. *G. Shad. Params* refers to the Gaussian Shading parameters when the model is deployed as a target model.

D.2. Prompting and Cover Image Datasets

Except for our SD2.1-Anime model, we use the *Stable-Diffusion-Prompts*[2] dataset to prompt benign watermarked images from the target models for all experiments (Secs. 4.2 to 4.6). For the cover images used in our Imprint-Forgery attack (Sec. 4.2), we only use the *MS-COCO-2017 Dataset* [21]. In experiments involving our Reprompting attack (Secs. 4.4 to 4.6), we use the *Inappropriate Image Prompts (I2P)*[3] dataset (sorting the *"inappropriate_percentage"* column in descending order) to generate harmful images with the attacker model. Across all experiments, images are of size 512×512.

D.3. Finetuned Model SD2.1-Anime

Our own finetune of SD2.1 is trained on 10,000 pairs of anime images and captions from the Anime-with-caption-CC0[4] dataset. We use the training scripts for Stable Diffusion models from Huggingface with default parameters for full finetuning of the UNet parameters[5] and for LoRA finetuning[6]. We add a keyword to each caption in the training set and reuse this keyword when generating images during our experiments. For every experiment involving SD2.1-Anime as the target model, we prompt the model with prompts similar to those in the training phase, by taking prompts from a separate split of Anime-with-caption-CC0 dataset and adding the keywords as prefix. By finetuning our own model, we achieve two goals: First, we obtain a baseline for a target model which is equal to the attacker model (SD2.1) in all aspects except for slight changes in the UNet parameters. This represents a scenario in which a service provider finetunes a publicly available model, and considers the resulting model unique enough to securely deploy semantic watermarks. Second, we are able to quantify the success of forgery attacks at different numbers of training steps for two different finetuning methods (full finetuning and LoRA) (see Sec. H). The model ultimately used in experiments is the fully finetuned variant.

[2] Stable-Diffusion-Prompts
[3] Inappropriate Image Prompts (I2P)
[4] Anime-with-caption-CC0
[5] Huggingface SD training script
[6] Huggingface SD LoRA training script

D.4. Number of Samples in Experiments

We provide the exact numbers of prompts, generated images, and cover images for our main experiments.

- *Imprint-Forgery Attack (Sec. 4.2).* We generate 100 images with each target model (using 100 prompts) and apply our Imprint-Forgery attack on 100 cover images. This adds up to 400 imprinting forgery attacks (100 for 4 target models each) across a fixed set of 100 pairs of prompts and cover images in total. This procedure is performed for both Tree-Ring and Gaussian Shading, doubling the final number of attacks.
- *Imprint-Removal Attack, Sec. 4.3.* We generate 100 images with each target model (using 100 prompts) and apply our Imprint-Removal on each image. This adds up to 400 removal attacks (100 for 4 target models each) across a fixed set of 100 prompts in total. This procedure is done for both Tree-Ring and Gaussian Shading, doubling the final number of attacks.
- *Reprompting Attack, Sec. 4.4.* We evaluate two different variations of our Reprompting attack.
  - *"Reprompt"*: Here, we generate 1,000 images with each target model using 1,000 benign prompts and reprompt each one with the attacker model using 1,000 harmful prompts. This adds up to 4,000 Reprompting attacks (1,000 for 4 target models each) across a fixed set of 1,000 pairs of benign and harmful prompts. This procedure is performed for both Tree-Ring and Gaussian Shading, doubling the final number of attacks.
  - *"Reprompt+"*: Here, we generate 100 images with each target model using 100 benign prompts. We reprompt each one 3 times with the attacker model using 300 harmful prompt, for both Tree-Ring and Gaussian Shading. For Gaussian Shading, we further resample the recovered latent $\tilde{z}_T^{(w)}$ for each target image 3 times before reprompting. This adds up to 1,200 Reprompting attacks for Tree-Ring (300 for each target model, across a fixed set of 100 benign and 300 harmful prompts), and 3,600 Reprompting attacks for Gaussian Shading (900 for each target model, across a set of 100 benign and 300 harmful prompts) in total.
- *Transferability Analysis, Sec. 4.5.* For each pair of attacker and target model, we generate 100 images using 100 benign prompts and reprompt each one with the attacker model using a set of 100 harmful prompts. This adds up to 4,900 Reprompting attacks (100 for each of the 49 combination of target/attacker model) across a fixed set of 100 pairs of benign and harmful prompts in total. We apply this procedure for Tree-Ring and Gaussian Shading, doubling the number of attacks.
- *No Defense by Adjusting Thresholds, Sec. 4.6.* We reuse the results from Secs. 4.2 and 4.4 as attack instances. Furthermore, we generate 1,000 images with each target model using 1,000 prompts. These are then transformed using common transformations and again verified by the target model. This adds up to an additional 1,000 image generations (1,000 for each of the two target models). We apply this procedure for Tree-Ring and Gaussian Shading, doubling the number.

D.5. Runtime of Attack Algorithms

All experiments were performed on 8 NVIDIA A40 GPUs. When executed on a single GPU for a single-batch attack example, the approximate runtimes for each attack algorithm are as follows:

- The *Imprint-Forgery* (Sec. 4.2) and *Imprint-Removal* (Sec. 4.3) attacks require between 25 and 40 minutes to perform 150 steps. The most time-consuming part of these algorithms is the gradient-based optimization done by the attacker model (SD2.1). Verification by the target model is set to take place every 10 optimization steps and is comparably fast.
- The *Reprompting attack* (Sec. 4.4) requires between 30 seconds (smaller models including SD2.1, Mitsua Diffusion One) and 2 minutes (FLUX.1). This time includes all steps: generating a single image with a semantic watermark on the target model, inverting and regenerating on the attacker model, and verifying the presence of the watermark with the target model.

D.6. Image Transformations

Fig. 18 shows examples for the standard image transformations that we apply on watermarked images to examine the achievable thresholds of Tree-Ring and Gaussian Shading under these transformations (cf. Sec. 4.6 and Sec. G).

D.7. Parameters for Tree-Ring and Gaussian Shading

In order to run Gaussian Shading and Tree-Ring, several parameters need to be selected.

Tree-Ring. In the original work of Wen et al. [41], the threshold for the p-value to assume the watermark's presence is computed by calculating receiver operating characteristics (ROC) curves. For their main results, they report AUROC and TPR@FPR=1%. As precise p-values are not reported, we conduct our own experiment similar to the original work. For each model evaluated, we generate 5,000 watermarked images as well as 5,000 clean images to determine the thresholds for desired FPRs. The thresholds for 10%, 1%, and 0.1% FPR are reported in Tab. 7. We kept all other parameters of the scheme as

Figure 18. Examples of common image transformations

provided in the implementation of the authors[7], i.e. we follow their instruction in the readme by inserting the *RING* pattern with default parameters into the 3rd latent channel. For the FLUX.1 model, which uses 16 latent in contrast to 4 latent channels as with the other models, this means that the embedded signal is weaker.

Gaussian Shading [45] requires the user to set the message length k, the repetition factor ρ, and the number of bins 2^ℓ to sample from a Gaussian distribution. The system was originally evaluated for SD2.1-like 4-channel latents with a scaling factor of 8 (i.e. $4 \times 64 \times 64$ for images of size 512×512). Ideal parameters for this size were experimentally determined by Yang et al. [45] as $k = 256$, $\rho = 64$, and $\ell = 1$. In Tab. 8, we provide detection thresholds τ for both zero-bit and multi-bit scenario for a desired FPR according to Eq. (6) (with $N = 1$ in case of zero-bit). Given $k = 256$, the detection threshold for a FPR of 10^{-6} is 0.64844 for a zero-bit scenario. Assuming $k = 256$ and $N = 100,000$ in line with Yang et al. [45], the detection threshold τ is set at 0.70703 for a multi-bit scenario. Throughout our work, we always assume this multi-bit scenario, meaning we use a definitive Gaussian Shading detection threshold of $\tau = 0.70703$ above which an image is recognized as watermarked. User attribution was done as described in Sec. 4: For each image to be attributed, first, the user id with the highest number of matching bits from a pool of $N = 100,000$ randomly generated users is retrieved. Then, if the number of matching bits exceeds τ, the sample is counted as attributed to this user. Our results regarding successful user attribution in Tabs. 1 to 3 always match the detection success because of small sample sizes, making it improbable for another user to match the bit string at hand and also pass the detection threshold.

In order to adapt the scheme to models like FLUX.1 with 16-channel latents (and the same scaling factor of 8), we have to adjust the parameters. We choose to use the same message length with a higher repetition factor ρ. A higher repetition factor makes the scheme more robust to bit-errors and therefore to perturbations. Using the same message length lets us keep thresholds τ unchanged regardless of the dimensions of the latent. We consider this to be at least as hard to attack as alternative adjustments, as increasing the message length will also yield lower detection thresholds in return as the probability of randomly drawing correct bit-strings of larger size decreases significantly. For example, using $k = 512$ would lower the thresholds for an FPR of 10^{-6} to 0.60547 (zero-bit) and 0.64648 (multi-bit), which marking removal harder and forgery easier.

In Tab. 6, we report our parameters for Gaussian Shading for multiple models with different latent sizes.

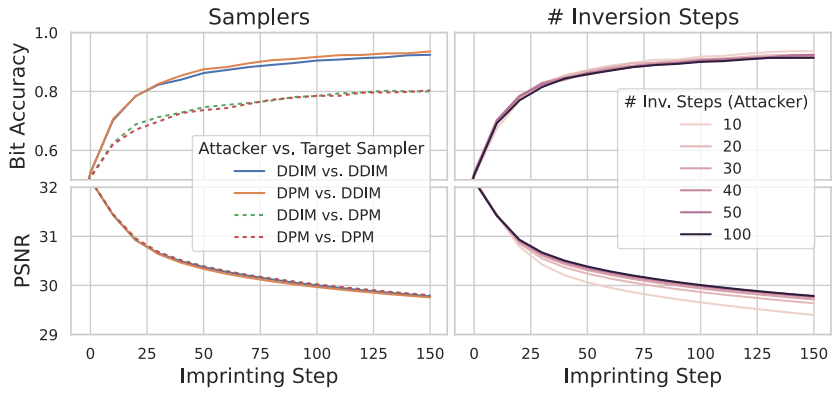
FPR	SD2.1-Anime	SDXL	PixArt-Σ	FLUX.1
10%	0.14275	0.13493	0.15809	0.14287
1%	**0.02466**	**0.02610**	**0.01591**	**0.02185**
0.1%	0.00555	0.00553	0.00048	0.00074

Table 7. Thresholds τ in terms of p-value for Tree-Ring for multiple FPRs and models. The thresholds used in our work are marked in bold.

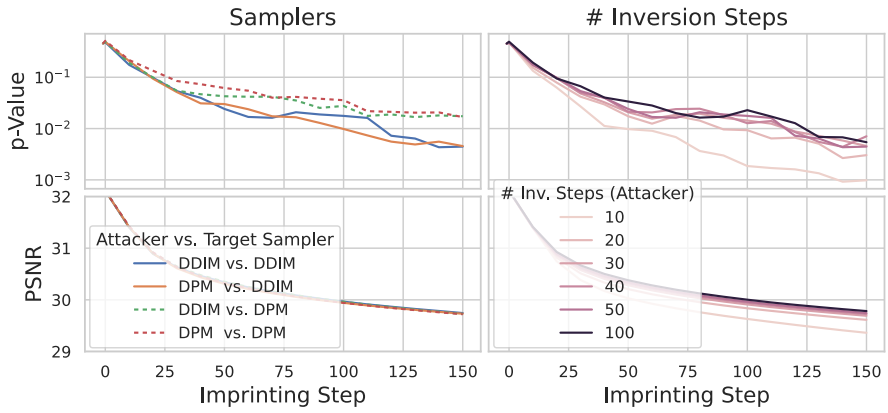
FPR	Zero-bit Threshold	Multi-bit Threshold
10^{-6}	0.64844	**0.70703**
10^{-16}	0.75000	0.78906
10^{-32}	0.85156	0.87500
10^{-64}	0.97266	0.98438

Table 8. Thresholds τ in terms of bit accuracy for Gaussian Shading, zero-bit and multi-bit configurations ($N = 100,000$ users) with a watermark capacity $k = 256$ at various FPR levels for all models. The threshold used in our work is marked in bold.

[7] Github Repository of Tree-Ring



(a) Gaussian Shading



(b) Tree-Ring

Figure 19. Ablation study over samplers used by the attacker and the target model used by the SP (left), as well as inversion steps used by the attacker (right) during the Imprint-Forgery attack.

E. Ablation Study of Sampler and Inversion Steps

We include ablation experiments to examine the impact of the samplers chosen by the service provider (SP) and the attacker. For this, we apply the Imprint-Forgery attack using SD2.1 against SDXL and report averaged results over 30 attack samples. In all cases, the watermarked reference images were generated by the SP with 50 denoising steps. Fig. 19 shows the results for both Gaussian Shading and Tree-Ring. For both schemes, we observe similar effects.

First, **the choice of the sampler** used for both generating watermarked images and retrieving initial latents on the side of the SP affects the attack's effectiveness in terms of bit accuracy and p-values measured on attack examples. If the SP chooses DDIM, the Imprint-Forgery attack is more effective, regardless of the sampler choice by the attacker. However, this is due to the fact that the SP's choice of sampler affects their general ability to recovery initial latents. We confirm this by measuring the average p-value measured for 100 generated watermarked images. While SDXL with DDIM achieves a mean p-value of 1.10×10^{-21} (see Table 5), it is much higher for DPM with 1.81×10^{-7}. This means that DPM is generally worse at retrieving initial latents and detecting the contained watermark. Hence, the SP would have to adjust detection thresholds in order to account for their lowered robustness against common transformation, again increasing their vulnerability against our attacks. We conclude that the choice of sampler on the SP side does not lead to viable defenses. Furthermore, the attacker does not need to know what sampler was used to create the watermarked reference image, as both DDIM and DPM show very similar attack effectiveness.

Second, choosing fewer **inversion steps** on the attacker's side slightly increases attack success and drastically lowers image quality. Again, we conclude that the attacker again does not require additional knowledge. They can simply choose a reasonably high number of inversions steps that yields satisfactory image quality without much impact on attack effectiveness.

F. Transferability Analysis

In this section, we present additional data for our transferability analysis from Sec. 4.5 by looking at the similarity of auto-encoders.

We evaluate the similarities of different VAEs, since all our models tested in Sec. 4.5 deploy VAEs with 4 latent channels and the same order of layers. We first examine the weights of the different autoencoders across the different models. SD2.1 and Common Canvas appear to share the exact same VAE, as all weights of all layers match with a precision of 10^{-3}. Next, we examine if some of the VAEs are functionally similar by comparing their latents for a set of images. We use a random sample of 100 images from the MS-COCO-2017 dataset [21], for which we compute latent embeddings using a VAE's encoder, and compare the representations obtained by different autoencoders using absolute cosine similarity.

The functional similarities are reported in Fig. 20. We observe that SD2.1, SD1.5, Waifu Diffusion, and Common Canvas have very similar latents accross all test images. In addition, we observe that SDXL and PixArt-Σ share a similar VAE as well. Mitsua's autoencoder appears to behave differently from the other VAEs, which supports the claim that Mitsua's VAE was trained from scratch, as stated on the corresponding Huggingface model card[8]. Comparing these results to Fig. 10 (which are also displayed in the figure below (left and middle) for convenience) reveals that latent encoder similarity correlates with high transferability, but does not explain it entirely. For example, Mitsua and Common Canvas have quite dissimilar autoencoders but still transfer easily in our experiments on the Reprompting attack. Also note that the heatmaps are almost symmetrical.

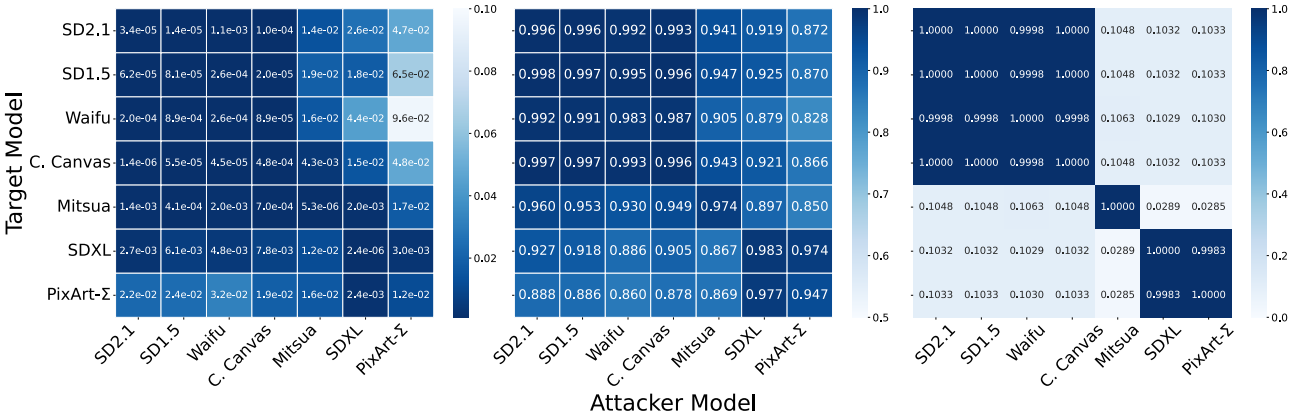
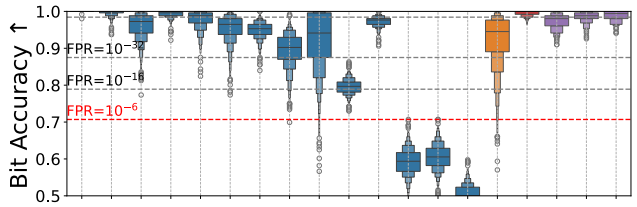


Figure 20. Transferability across different pairs of target and attacker model and functional similarity of autoencoders. For convenience, the **left** plot shows the Reprompting attack transferability for Tree-Ring in terms of p-value (↓), and the **middle** plot for Gaussian Shading in terms of bit accuracy (↑). The **right** plot shows the functional analysis of autoencoders. Functional similarity is measured as averaged cosine similarity between latents of different autoencoders for 100 different images.

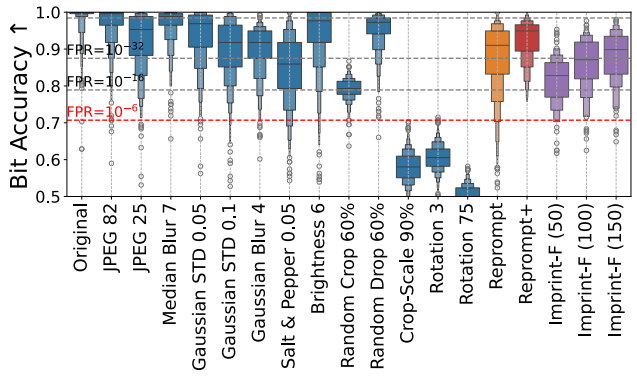
[8] Mistua Diffusion One Huggingface model card

G. A Side-note on Robustness

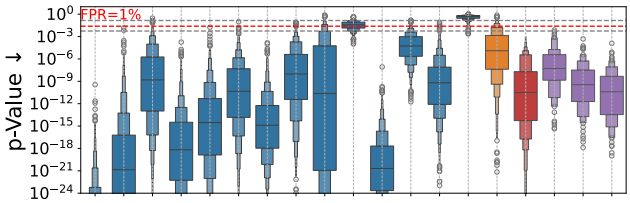
Fig. 21 provides further intuition on possible thresholds (Fig. 9) by showing results for a more complete set of transformations and settings. We can identify a notable vulnerability of the watermarking methods to some common perturbations. Gaussian Shading is especially vulnerable to cropping, scaling, and rotation. The p-values in Tree-Ring are significantly affected by crop-and-scale operations. It appears that transformations that change the location of the pixels, such as crop-and-scale and rotation, can significantly disturb both watermarks. In contrast, deleting parts of the image (e.g. random drop) preserves the watermark, as enough pixels are still in the correct position. It appears that in order to decode the watermark, Gaussian Shading requires the image to remain in the same position with which it was generated, which might not be a viable assumption in practical deployment. This illustrates that these watermarks can be easily removed and that further research into more robust semantic watermarking methods is necessary.



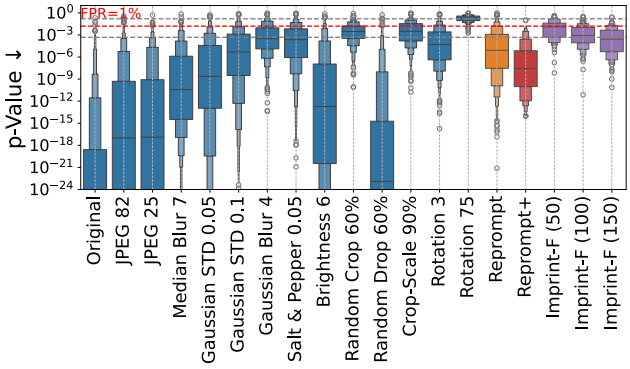
(a) Bit Accuracy (↑) for Gaussian Shading (SD2.1-Anime)



(b) Bit Accuracy (↑) for Gaussian Shading (PixArt-Σ)



(c) P-values (↓) for Tree-Ring (SD2.1-Anime)



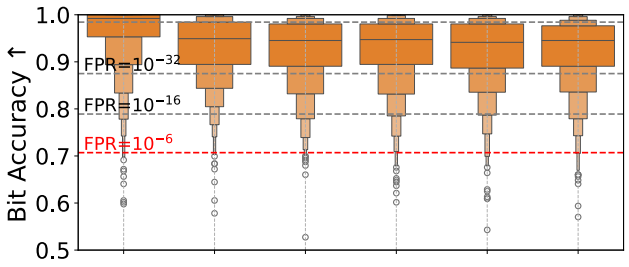
(d) P-values (↓) for Tree-Ring (PixArt-Σ)

Figure 21. Comparison between image perturbations (blue bars) and our attacks (orange, red and purple bars), for Gaussian Shading (left) and Tree-Ring (right). Orange bars are the Reprompting attack, red bars are enhanced Reprompt+ attack, purple bars are Imprint-Forgery for different numbers of optimization steps. Results are shown for two target models: SD2.1-Anime (top) and PixArt-Σ (bottom). We can observe that no threshold can effectively separate images from common transformations and attacks.

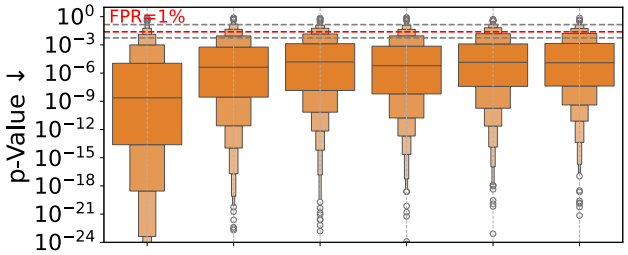
H. Attack Success vs Training Steps

For different amounts of finetuning steps of our SD2.1-Anime finetune, Fig. 22 shows bit accuracies and p-values achieved with our Reprompting attack on Gaussian Shading and Tree-Ring, respectively. The attacker model is SD2.1.

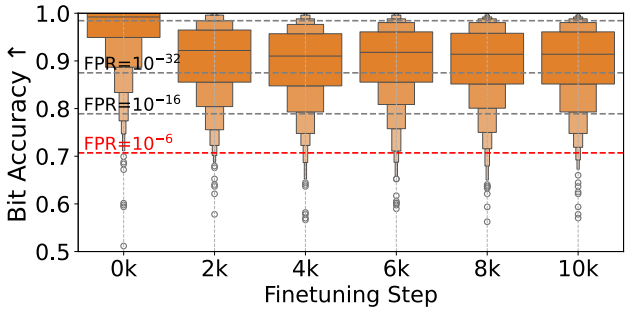
The attack effectiveness decreases with longer finetuning, as expected. However, the longer the target model is trained, the slower the rate of decrease. In our experiment, the attack metrics seems to stabilize after around 4k training steps, still crossing most detection thresholds. Importantly, the bit accuracy for the attack images exceeds the detection threshold set for FPR=10^{-6} (the threshold from the original paper, marked in red in Fig. 22). Similarly, for Tree-Ring, the p-values fall below the detection threshold for the FPR=1% setting from the original paper (marked in red).



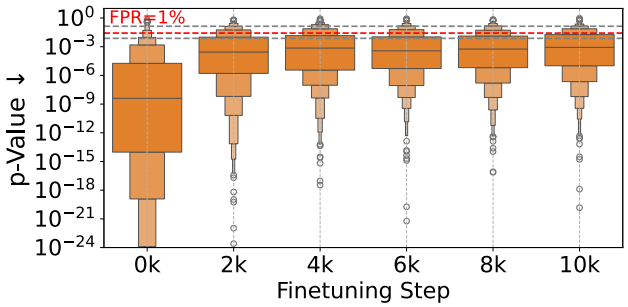
(a) Bit Accuracy (↑) for Gaussian Shading (Full Finetuning)



(c) P-Values (↓) for Tree-Ring (Full Finetuning)



(b) Bit Accuracy (↑) for Gaussian Shading (LoRA Finetuning)



(d) P-Values (↓) for Tree-Ring (LoRA Finetuning)

Figure 22. Performance of the Reprompting attack for Gaussian Shading (left, measured using bit accuracy) and Tree-Ring (right, measured using p-value), after different numbers of finetuning of our SD2.1-Anime model. Numbers are reported both for regular full finetuning (top), as well as finetuning using LoRA (bottom). Marked dashed red lines are the thresholds used in the original papers, corresponding to an FPR of 10^{-6} for Gaussian Shading and an FPR of 1% for Tree-Ring watermarks. In the case of Gaussian Shading, the theoretical thresholds hold for all the finetuning steps. For Tree-Ring, the thresholds displayed are the ones empirically determined on the 10k step for full finetuning and LoRA finetuning, respectively.