SWA-LDM: Toward Stealthy Watermarks for Latent Diffusion Models

Zhonghao Yang[1], **Linye Lyu**[2], **Xuanhang Chang**[2], **Daojing He**[1,2] **& Yu Li**[2]
[1]Software Engineering Institute, East China Normal University
[2]School of Computer Science and Technology, Harbin Institute of Technology (Shen Zhen)
{ashzhonghao, yu.li.sallylee}@gmail.com

Abstract

Latent Diffusion Models (LDMs) have established themselves as powerful tools in the rapidly evolving field of image generation, capable of producing highly realistic images. However, their widespread adoption raises critical concerns about copyright infringement and the misuse of generated content. Watermarking techniques have emerged as a promising solution, enabling copyright identification and misuse tracing through imperceptible markers embedded in generated images. Among these, latent-based watermarking techniques are particularly promising, as they embed watermarks directly into the latent noise without altering the underlying LDM architecture. In this work, we demonstrate—for the first time—that such latent-based watermarks are practically vulnerable to detection and compromise through systematic analysis of output images' statistical patterns. To counter this, we propose SWA-LDM (Stealthy Watermark for LDM), a lightweight framework that enhances stealth by dynamically randomizing the embedded watermarks using the Gaussian-distributed latent noise inherent to diffusion models. By embedding unique, pattern-free signatures per image, SWA-LDM eliminates detectable artifacts while preserving image quality and extraction robustness. Experiments demonstrate an average of 20% improvement in stealth over state-of-the-art methods, enabling secure deployment of watermarked generative AI in real-world applications.

1 Introduction

The Latent Diffusion Models (LDMs) (Rombach et al., 2022) represent a significant advancement in efficient, high-quality image generation. By leveraging Variational Autoencoders (VAEs) (Kingma & Welling, 2014), LDMs transfer diffusion model operations from pixel space to latent space, allowing UNet (Ronneberger et al., 2015) architectures to perform denoising in a lower-dimensional space. This shift dramatically enhances computational efficiency, enabling companies and individuals with limited resources to train models for commercial usage. Consequently, popular models such as DALL-E 2 (Ramesh et al., 2022), and Midjourney (Midjourney) have emerged, facilitating the generation of high-quality, realistic images via user-accessible APIs.

The rapid advancements of LDMs have introduced critical challenges, particularly concerning copyright infringement and the potential misuse of generated content. Copyright violations arise when malicious actors steal and resell proprietary diffusion models, resulting in substantial financial losses for original creators. Additionally, the capability to generate hyper-realistic images has been exploited by individuals disseminating misinformation and fake news, thereby undermining public trust and social stability. Addressing these issues is paramount for safeguarding intellectual property rights and maintaining societal integrity.

To alleviate these issues, current LDMs employ watermarking techniques to embed pre-designed imperceptible watermarks within the generated images. Then, one can extract this watermark using corresponding methods to identify the image's origin. Existing watermark methods fall into two categories: post-processing watermarking (Ó Ruanaidh et al., 1996; O'Ruanaidh & Pun, 2002; Cox et al., 2007; Zhang et al., 2019) and in-generation-process watermarking (Fernandez et al., 2023; Wen et al., 2023; Yang et al., 2024b; Lei et al., 2024; Feng et al., 2024). Post-processing methods

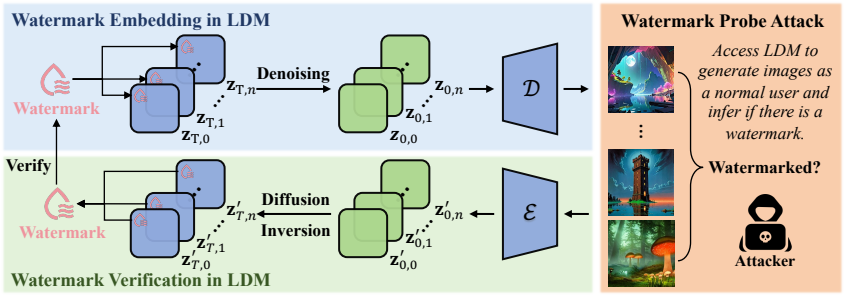


Figure 1: The general framework of the latent-based watermarking method for LDMs. They often add the same watermark signal to different generated images, which attackers can exploit to detect the presence of watermark.

add watermarks after the images have been generated by LDMs, but they often compromise image quality (Fernandez et al., 2023). Alternatively, in-generation-process methods embed watermarks during the image generation process, which can be further divided into model-based (Fernandez et al., 2023; Feng et al., 2024) and latent-based methods (Wen et al., 2023; Yang et al., 2024b; Lei et al., 2024). The former embeds watermarks by modifying LDMs' parameters (*e.g.* VAE, UNet), resulting in training costs. In contrast, the latent-based methods, as shown in Fig. 1, embed a watermark to the latent noise before the denoising process. This approach eliminates the need for extensive retraining and incurs minimal computational overhead, making it highly efficient for practical applications.

However, a significant limitation of current latent-based watermarking techniques is their reliance on constant watermarks across all generated outputs, making them susceptible to detection by malicious users. This paper highlights this vulnerability for the first time, demonstrating that the stealthiness of existing methods can be easily compromised using only the generated images. Unlike prior works that attempt to remove watermarks without first verifying their presence (Saberi et al., 2024; Yang et al., 2024a), we propose an attack to determine whether an image generated by an LDM contains a watermark, which can further inform adversarial actions. This attack also serves as an evaluation metric for the stealthiness of latent-based watermarking techniques. Specifically, we design a feature extractor to identify constant watermark signals in images generated by the target LDM. Successful extraction of a constant signal indicates the presence of a watermark. Through this attack, we emphasize the urgent need to enhance the stealthiness of watermarking techniques to safeguard against unauthorized use.

To address this vulnerability, we introduce SWA-LDM, a plug-and-play component compatible with any latent-based watermarking method to create stealthy watermarks. Our approach randomizes the watermark by embedding image-dependent signals into generated images, effectively preventing the detection of a constant signal. The closest related work, Gaussian Shading, uses stream ciphers for randomization but incurs high management costs due to the need to remember a unique nonce for each image. In contrast, SWA-LDM leverages the inherent randomness of latent noise to generate image-dependent watermarks without additional management overhead. Specifically, we introduce a key channel sampled from the latent noise to create a random key that shuffles the watermark, ensuring uniqueness for each image. During watermark verification, SWA-LDM reconstructs the latent variable via diffusion inversion and extracts the key to retrieve the original watermark. However, inaccuracies may arise due to diffusion inversion errors and image transmission noises. To mitigate this, we propose an enhancement algorithm to store redundant keys in the key channel while preserving its distribution. The combination of randomized watermarks and key channel enhancement facilitates the generation of stealthy and robust watermarks.

Our contributions are summarized as follows: ① We are the first to expose the stealthiness vulnerabilities inherent in current latent-based LDM watermarking methods, which generate constant watermarks that can be easily exploited by malicious users for detection. Our effective watermark probe attack demonstrates this vulnerability, underscoring the critical need for enhanced watermarking strategies. ② We present SWA-LDM, a versatile plug-and-play component compatible with any latent-based watermarking method, designed to create stealthy watermarks. By leveraging the inherent randomness of latent noise, SWA-LDM generates image-dependent watermarks without incurring additional management costs. Additionally, we propose an enhancement algorithm that incorporates redundant keys within the key channel, preserving its distribution while significantly

improving watermark robustness. ③ We conduct comprehensive experiments to evaluate the proposed watermark probe attack and SWA-LDM. Results show that SWA-LDM effectively improves the stealthiness of latent-based watermarks while achieving competitive visual quality, image-text similarity, and watermarking robustness.

2 Background and Related Works

Latent Diffusion Models. Latent diffusion models are a computationally efficient version of diffusion models (Rombach et al., 2022). LDMs leverage a pretrained autoencoder to compress image $x \in \mathbb{R}^{3\times H\times W}$ in RGB space into a lower dimensional latent representation $z \in \mathbb{R}^{c\times h\times w}$. Training and sampling LDMs in the latent space significantly reduces the computational complexity. More specifically, during training, the encoder $\mathcal{E}$ encodes the image x into a latent representation by $z = \mathcal{E}(x)$. Next, LDMs conduct diffusion and denoising process in the latent space, which converts z to a latent noise z_T and recovers the image latent $\tilde{z}$ from z_T respectively over T timesteps. Then, the decoder $\mathcal{D}$ reconstructs the image $\tilde{x}$ from the recovered latent by $\tilde{x} = \mathcal{D}(\tilde{z})$. During sampling, the LDMs sample a noise latent vector z_T from Gaussian distribution $\mathcal{N}(0, \mathbf{I})$. Subsequently, the trained LDM can utilize sampling methods like Denoising Diffusion Implicit Models (DDIM) (Song et al., 2020; Nichol & Dhariwal, 2021) or DPM-Solver (Lu et al., 2022) to obtain the latent representation of the sampled image z_s from z_T over T timesteps. Then, the decoder reconstructs the image from the latent by $x_s = \mathcal{D}(z_s)$. Besides, one can use methods like DDIM Inversion (Mokady et al., 2023) to invert the denoising process and recover the initial noise z_T from the generated image x_s.

Watermarks for Latent Diffusion Models. LDMs enable individuals to customize their own models for specific styles of image generation via training and fine-tuning, which they can publish and exchange in the online market space such as Civitai (Inc.) and Tensor.art (Tensor.art). However, these advancements have also raised concerns about the potential abuse of these models and the generated images. For instance, unauthorized commercial exploitation of LDM-generated images lacking inherent copyright protection is a significant risk. Besides, malicious users can generate realistic images to spread rumors and fake news on social media, potentially manipulating important social and economic events such as political elections and the stock market. Therefore, enhancing LDMs with copyright protection and traceability techniques is crucial. Watermarking has a long history to alleviate these issues via labeling image content (Ó Ruanaidh et al., 1996), which involving incorporating watermark information into the generated images. Then, one can identify the origin of the images by verifying the watermark.

Existing watermarking methods for LDMs can be categorized into post-processing and in-generation-process watermarks. Post-processing methods add watermarks to images after they have been generated by LDMs. For instance, the Stable Diffusion repository provides methods like DWT-DCT (Rahman, 2013) and RivaGAN (Zhang et al., 2019). Despite their widespread usage, direct modification to the images can degrade image quality (Fernandez et al., 2023). Alternatively, recent research proposes in-generation-process watermarks, which integrate the watermark embedding with the image generation process. Stable Signature (Fernandez et al., 2023) and AquaLora (Feng et al., 2024) embed watermarks by fine-tuning the VAE decoder and UNet of the LDMs, respectively. These model-based methods improve the watermarked image quality but introduce substantial computational costs for training the model parameters. Conversely, recent works propose latent-based watermarks, which embed the watermarks into the latent space of the diffusion models. Tree-Ring (Wen et al., 2023) encodes the watermark in the frequency domain of the latent noise, while Gaussian Shading (Yang et al., 2024b) maps the watermark to the latent variable following Gaussian distribution. DiffuseTrace (Lei et al., 2024) uses an encoder model to modify the initial latent noise variable. Latent-based methods are free of model parameter modifications, making them much less computational and more user-friendly.

While latent-based methods hold great promise for practical usage, our research reveals a critical issue: even though invisible, these techniques produce a constant signal across generated images. This uniformity undermines the stealthiness of the watermarks, increasing the risk of copyright infringement. To address this, we propose a plug-and-play component that integrates with existing latent-based watermarking methods and enhances their stealthiness.

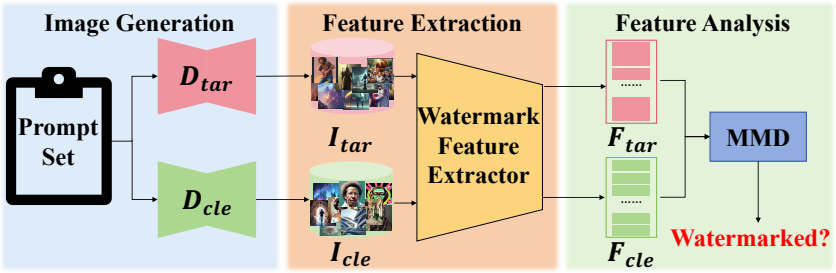


Figure 2: The overview of watermark probe attack.

3 Watermark Probe Attack

We introduce a promising watermark probe attack to detect the presence of latent-based watermarks by analyzing a set of images generated by the target LDM.

3.1 Threat Model

The watermark probe attack targets a scenario with two parties: the model owner providing the image generation service and the watermark probe attacker.

Model Owner. The owner of the LDM deploys it on a platform (*e.g.* Face; Inc.; Tensor.art) and provides image generation services through API access. To protect image copyrights and ensure traceability in cases of misuse, the owner embeds imperceptible watermarks in each generated image without degrading image quality. For any given image, the owner can verify whether it contains their watermark and identify the associated user, a process known as watermark verification, which must remain highly accurate even after image perturbations. The model owner controls the entire LDM, image generation, and verification process.

Watermark Probe Attacker. The attacker aims to detect the presence of watermarks in images generated by target LDM D_{tar}. The attacker generates images using the API and controls only the prompts, without access to the model's internals, the initial noise, or any knowledge of the model, watermark method, or watermark detector. Also, the attacker can utilize open-source models D_{cle} to generate watermark-free images with the same prompts.

3.2 Overview

Figure 2 illustrates our watermark probe attack, consisting of three modules: Image Generation, Feature Extraction, and Feature Analysis.

Image Generation. In this module, we generate two image sets: the target image set I_{tar} from the target LDM D_{tar} and the clean image set I_{cle} from the clean LDM D_{cle}. Both sets share the same prompt set P to ensure any differences are primarily due to the watermark. These image sets are then used to train the watermark feature extractor in the next module.

Feature Extraction. The goal of this module is to train a Watermark Feature Extractor WFE, which tries to extract the constant watermark from the generated images. To achieve this, we design three loss functions for training WFE based on the extracted features: the first loss $\mathcal{L}_{at}$ encourages WFE to aggregate target image features to find the constant watermark signal; the second loss $\mathcal{L}_{dtc}$ motivates the extractor to distinguish between target image and clean image features; and the third loss $\mathcal{L}_{gc}$ let the clean image features follow a random distribution to prevent extractor from only detecting the signals caused by model difference instead of watermark. These loss components are explained in the Sec. 3.4. Thus, the total loss function is:

$$\mathcal{L}_{total} = \mathcal{L}_{at} + \alpha\mathcal{L}_{dtc} + \beta\mathcal{L}_{gc}, \tag{1}$$

where α and β are hyperparameters to control the contribution of each loss.

Feature Analysis. This module determines whether the target diffusion model D_{tar} contains a watermark. We use the trained WFE from previous method to extract F_{cle} and F_{tar} from I_{cle} and I_{tar}. Then, we measure the distribution difference between these two features using Maximum Mean

Discrepancy (MMD) (Gretton et al., 2012) metric. If the feature distributions differ significantly, our method predicts that the target model is watermarked and vice versa.

3.3 Image Generation

We begin by collecting a prompt set P to create the training dataset, comprising I_{cle} and I_{tar}. Ideally, a watermark-free version of the target model would be used as the clean model to generate a corresponding watermark-free image. The only distinction between the two sets is the presence of the watermark. By analyzing distributional differences, we can infer the watermark's presence—if no difference is observed, the image is watermark-free; if differences exist, a watermark is likely present. In practice, attackers often have access to only an approximate watermark-free model. Since most LDMs are fine-tuned from open-source models (Zhang et al., 2023; 2024), there is a similarity between the output distributions of the target and clean LDMs. This similarity amplifies the differences caused by the watermark, facilitating effective detection of its presence.

3.4 Feature Extraction

In this module we attempt to detect watermark signal in the generated images by training a Watermark Feature Extractor WFE. The WFE should have the following behaviors for successful watermark probe attack: when the watermark exists in the target images, the extractor should identify the watermark signal, causing F_{tar} to converge; besides, the extractor should also identify the distribution difference between the F_{cle} and F_{tar} caused by the watermark: furthermore, the extractor should only detect the constant signal contributed by the watermark instead of the inherent difference between D_{cle} and D_{tar}. To achieve these behaviors, We design three types of losses to achieve these properties. To encourage F_{tar} to converge during training, we introduce aggregating loss for targe feature $\mathcal{L}_{at}$, which calculates the variance of the target features as shown in Equation 2:

$$\mathcal{L}_{at} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \left\| f_{tar}^{i} - f_{tar}^{j} \right\|^2 \tag{2}$$

Besides, we introduce difference $\mathcal{L}_{dtc}$ loss to distinguish the difference between the F_{cle} and F_{tar}. $\mathcal{L}_{dtc}$ calculates the reciprocal of the difference between the matched f_{tar} and f_{cle} as shown in Equation 3.

$$\mathcal{L}_{dtc} = \frac{1}{\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left\| f_{tar}^{i} - f_{cle}^{i} \right\|^2} \tag{3}$$

Even if the target images are watermark-free, $\mathcal{L}_{at}$ and $\mathcal{L}_{dtc}$ may converge due to the model difference between D_{cle} and D_{tar}. WFE can falsely treat the model difference as the watermark difference, which leads to false positive detection result. To alleviate this , we propose the third loss $\mathcal{L}_{gc}$ to prevent the WFE model from learning the model-difference features. $\mathcal{L}_{gc}$ is motivated by one property of watermarking: when the input is a watermark-free image the watermark extractor should produce a random output. Therefore, $\mathcal{L}_{gc}$ encourages the extracted features from the clean images to follow a random distribution. Hence, $\mathcal{L}_{gc}$ calculates the KL divergence (Csiszar, 1975) between the F_{cle}'s distribution and a Gaussian distribution, as shown in Equation 4 .

$$\mathcal{L}_{gc} = \frac{1}{N} \sum_{i=1}^{N} \text{KL} \left(f_{cle}^{i} \parallel \frac{1}{M} \right), \tag{4}$$

where N is the batch size, M is the feature dimension size.

4 SWA-LDM

We introduce SWA-LDM, a plug-and-play component for existing latent watermarking methods that generates image-dependent watermarks to counter watermark probe attack.

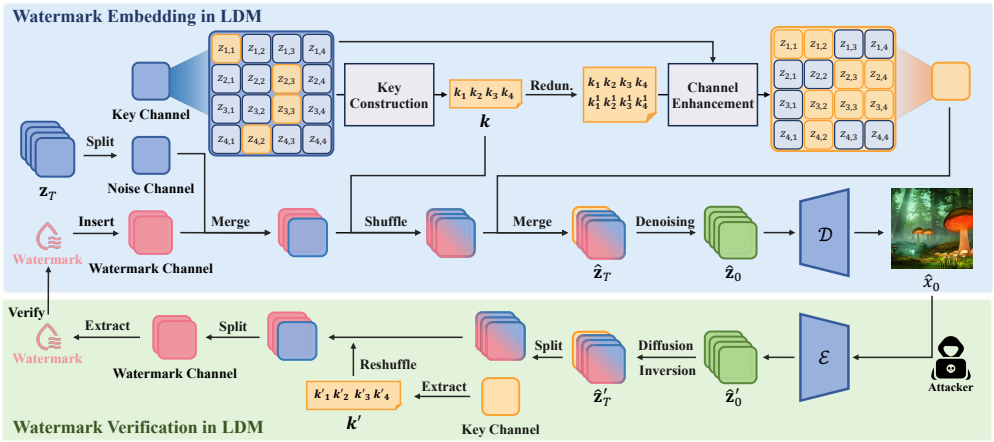


Figure 3: The framework of SWA-LDM. We extract the key $\mathbf{k}$ from randomly initialized latent variables and use it to shuffle the remaining latent variables where the watermark is inserted. This ensures the watermark information is randomized in each generated image.

4.1 Overview

The framework of SWA-LDM is shown in Fig. 3. During watermark embedding, SWA-LDM initializes latent noise $\mathbf{z}_T$ sampled from a standard Gaussian distribution, which it then splits into key, noise, and watermark channels. The key and noise channels retain random noise, while the watermark channel is reinitialized with watermark-embedded noise based on the chosen latent-based watermarking method. To randomize the watermark, SWA-LDM leverages the inherent randomness of latent noise by extracting a random seed (key) from the noise in the key channel. To ensure reliable key recovery to counter diffusion inversion errors and image transmission noises, we design a robust key construction mechanism and enhance the key channel for stronger key information. The key then seeds the random number generator to shuffle the watermark and noise channels, and the key channel is merged to produce the watermarked latent noise $\hat{\mathbf{z}}_T$. The subsequent denoising and image generation process follows the standard procedure of LDMs.

During watermark verification, SWA-LDM restores the image to latent space, obtaining $\hat{\mathbf{z}}'_0$, and uses diffusion inversion method to approximate the original latent noise $\hat{\mathbf{z}}'_T$. SWA-LDM partitions $\hat{\mathbf{z}}'_T$ to extract the key from the key channel, which is used to reshuffle the remaining channels. This process recovers the latent noise from the watermark channel, from which the watermark is extracted and verified. The closest work, Gaussian Shading (Yang et al., 2024b), using stream ciphers to encrypt latent noise, introducing randomness to the watermarked latent distribution. However, stream ciphers require a unique nonce per latent noise to achieve randomness, meaning each generated image must be paired with a specific nonce. This nonce must be managed and matched with the corresponding image during watermark verification, as it is essential for decrypting the latent noise to verify the watermark. This nonce management complicates practical implementation in LDM applications that generate high volumes of images. In contrast, SWA-LDM operates without any additional information management.

4.2 Watermark Embedding

Each step of watermarking embedding in SWA-LDM is illustrated below.

Channel Splitting. SWA-LDM initializes the latent noise $\mathbf{z}_T \in \mathbb{R}^{c\times h\times w}$ and divide it into key channels $\mathbf{z}_T^k \in \mathbb{R}^{c_k\times h\times w}$, noise channels $\mathbf{z}_T^n \in \mathbb{R}^{c_n\times h\times w}$, and watermark channels $\mathbf{z}_T^w \in \mathbb{R}^{c_w\times h\times w}$. The key and noise channels are filled with randomly sampled noise from a standard Gaussian distribution $\mathcal{N}(0,\mathbf{I})$. Meanwhile, the watermark channel is initialized with a chosen latent-based watermarking method (*e.g.*, Yang et al. (2024b))

Key Construction. SWA-LDM uses a pseudorandom number generator (PRNG) and shuffle algorithm to randomize the latent noise in the watermark and noise channel. The PRNG seed must meet three criteria: (1) each seed is randomly generated and unique per image, (2) it can be reliably reconstructed during watermark verification, and (3) it does not require additional management.

Algorithm 1: Key Channel Enhancement

Input: $\mathbf{z}_T^k$: Latent noise in key channel, $\mathbf{k}$: Extracted key bits, R: Number of redundancies, $\mathcal{M}$: Mapping function
Output: $\overline{\mathbf{z}}_T^k$: Modified latent noise with robust key information

for $r \leftarrow 1$ **to** R **do**
  for $m \leftarrow 1$ **to** *len*($\mathbf{k}$) **do**
    /* Find the latent noise corresponding to k_m^r */
    $(i, j, q) \leftarrow \mathcal{M}(r \times M + m)$
    $k_m^r \leftarrow 1$ if $z_{T,i,j,q}^k > 0$ else 0
    if $k_m^r \neq k_m$ **then**
      /* Search for latent noise to swap */
      $p \leftarrow m + 1$
      while *True* **do**
        $(i', j', q') \leftarrow \mathcal{M}(r \times M + p)$
        new_bit $\leftarrow 1$ **if** $z_{T,i',j',q'}^k > 0$ **else** 0
        if *new_bit* $= k_m$ **then**
          swap($z_{T,i,j,q}^k, z_{T,i',j',q'}^k$)
          Break
        $p \leftarrow p + 1$
$\overline{\mathbf{z}}_T^k \leftarrow \mathbf{z}_T^k$
return $\overline{\mathbf{z}}_T^k$

To achieve this, SWA-LDM derives the key $\mathbf{k}$ directly from the latent noise. Given that LDMs transform latent noise $\mathbf{z}_T$, sampled from a Gaussian distribution $\mathcal{N}(0, \mathbf{I})$, into an image $\mathbf{x}_0$, this approach retains the necessary randomness and ensures compatibility with diffusion inversion, fulfilling the requirements for $\mathbf{k}$. However, during diffusion inversion, the reconstructed $\mathbf{z}'_T$ may not perfectly match the original $\mathbf{z}_T$, especially when $\mathbf{x}_0$ experiences perturbations. Therefore, SWA-LDM must reliably construct $\mathbf{k}$ even in the presence of these variances. For robustness, SWA-LDM abstracts specific elements from the latent noise to construct each bit of $\mathbf{k}$. First, we define a mapping function to consistently sample fixed locations within the latent noise for each bit in $\mathbf{k}$. Specifically, a mapping function $\mathcal{M} : \{1, 2, \ldots, N\} \rightarrow \{(i, j, q) \mid i \in [1, c_k], j \in [1, h], q \in [1, w]\}$, with $N = c_k \times h \times w$, allows SWA-LDM to consistently access the same positions in $\mathbf{z}_T^k$ for $\mathbf{k}$-bit construction. For simplicity, $\mathcal{M}$ is implemented as a sequential mapping, unfolding $\mathbf{z}_T^k$ linearly to assign each bit of $\mathbf{k}$.

Next, each bit of $\mathbf{k}$ is sampled based on the sign of specific latent variables $z_{T,i,j,q}^k$ within $\mathbf{z}_T^k$. Letting M denote the bit-length of $\mathbf{k}$, each bit is determined as follows:

$$\mathbf{k} = [k_1, \ldots, k_M] \quad k_m = \begin{cases} 1, \text{if } z_{T,i_m,j_m,q_m}^k > 0 \\ 0, \text{if } z_{T,i_m,j_m,q_m}^k \leq 0 \end{cases} \tag{5}$$

where $(i_m, j_m, q_m) = \mathcal{M}(m)$ indicates the index of k_m in the latent noise $\mathbf{z}_T^k$.

Key Channel Enhancement. While the key construction accounts for noise variations, it may still fail to reliably recover $\mathbf{k}$ under perturbations. To address this, we propose a method to construct redundant key information within $\mathbf{z}_T^k$, ensuring robust key extraction with minimal modification to $\mathbf{z}_T^k$. Let R represent the number of redundant key, and define the r-th redundant key as $\mathbf{k}^r$, where $r \in [1, R]$ and $k_m^r = k_m$ for $m \in [1, M]$. For each redundant key $\mathbf{k}^r$, we map it to a set of latent noise using the mapping function $\mathcal{M}$. The latent variable z_{T,i_n,j_n,q_n}^k corresponds to k_m^r with $(i_n, j_n, q_n) = \mathcal{M}(r \times M + m)$. If the relationship between k_m^r (either 0 or 1) and z_{T,i_n,j_n,q_n}^k (either ≤ 0 or > 0) does not match, we search for a latent noise element that satisfies the condition and swap the corresponding values. The key channel enhancement process is detailed in Algorithm 1. This algorithm takes the latent noise $\mathbf{z}_T^k$, the key $\mathbf{k}$, and the number of redundant key R as input, and outputs the enhanced latent noise $\overline{\mathbf{z}}_T^k$, which includes the redundant key.

Latent Noise Shuffling. As previously discussed, SWA-LDM embeds the watermark into the latent noise of the watermark channel, resulting in $\hat{\mathbf{z}}_T^w$. To randomize this embedded watermark, SWA-LDM uses $\mathbf{k}$ as a seed for the pseudorandom number generator (PCG64) (O'Neill, 2014). The Fisher-Yates shuffle algorithm (Eberl, 2016) is then applied to permute $\text{concat}(\hat{\mathbf{z}}_T^w, \mathbf{z}_T^n)$, dispersing watermark information across the latent space. Finally, we concatenate the enhanced latent noise $\overline{\mathbf{z}}_T^k$ with the shuffled watermark channel to form the final watermarked latent noise $\hat{\mathbf{z}}_T$.

Image Generation. After constructing the watermarked latent noise $\hat{\mathbf{z}}_T$, the image generation process follows the standard procedure of the LDMs. Specifically, we utilize DDIM (Song et al., 2020) for denoising of $\hat{\mathbf{z}}_T$. Once the denoised latent $\hat{\mathbf{z}}_0$ is obtained, the watermarked image $\hat{\mathbf{x}}_0$ is generated by applying the LDM decoder $\mathcal{D}$: $\hat{\mathbf{x}}_0 = \mathcal{D}(\hat{\mathbf{z}}_0)$.

4.3 Watermark Verification

Diffusion Inversion. For watermark verification, we use the LDM encoder $\mathcal{E}$ to map the watermarked image $\hat{\mathbf{x}}_0$ back to the latent space, obtaining $\hat{\mathbf{z}}'_0 = \mathcal{E}(\hat{\mathbf{x}}_0)$. We then apply diffusion inversion over T timesteps, estimating the additive noise to recover $\hat{\mathbf{z}}'_T \approx \hat{\mathbf{z}}_T$. Here, DDIM inversion (Mokady et al., 2023) is used to approximate the original latent noise.

Robust Key Extraction. With $\hat{\mathbf{z}}'_T$ obtained, we partition it to isolate the key channel $\overline{\mathbf{z}}_T^{k'}$ containing the redundant key information and the shuffled channel. Using a fixed mapping function $\mathcal{M}$, we extract the redundant key information from predetermined positions in $\overline{\mathbf{z}}_T^{k'}$ to obtain both the key $\mathbf{k}'$ and its redundant bits $\{\mathbf{k}^{r'} \mid r \in [1, R]\}$. Each bit k'_m of $\mathbf{k}'$ is determined by a majority voting mechanism, wherein if more bits are zero than one among k'_m and $\{k_m^{r'} | r \in [1, R]\}$, k'_m is set to zero; otherwise, it is set to one.

Reshuffling Watermark Information and Verification. After recovering the key $\mathbf{k}'$, we use it as the seed for a pseudorandom number generator (PCG64) and reapply the Fisher-Yates shuffle algorithm to re-shuffle the latent noise, excluding the key channel. This reshuffled latent noise is then split to isolate $\hat{\mathbf{z}}_T^{w'}$ and $\mathbf{z}_T^{n'}$. Finally, based on the latent-based watermarking method employed, we extract and verify the watermark from $\hat{\mathbf{z}}_T^{w'}$.

5 Experiment

5.1 Setup

Latent Diffusion Models. We employed three widely-used Stable Diffusion models as base models: Stable Diffusion v1-5 (SD v1-5), Stable Diffusion v2-1 (SD v2-1), and SD-XL 1.0-base (SDXL 1.0). For customized models, we downloaded 60 checkpoints from Hugging Face (Face), fine-tuned from three base models (SD v1-5, SD v2-1, and SDXL 1.0), with each base model comprising 20 different checkpoints. Detailed on these 60 checkpoints is provided in the Appendix. Compared to previous work, our study covers the largest model set to date (60 models, *vs*. Tree-ring (Wen et al., 2023) with 1, Gaussian Shading (Yang et al., 2024b) with 3, and DiffuseTrace (Lei et al., 2024) with 2).

Image Generation Details. To generate images, we use prompts from the Stable-Diffusion-Prompts dataset (Gustavosta). The generated image resolution is 512×512 pixels, with latent noise dimensions set to 4×64×64 and a guidance scale of 7.5. We use DDIM sampling (Song et al., 2020) with 50 timesteps. In practice, the original prompts of the generated images are often not shared. Hence, we use an empty prompt for diffusion inversion (Mokady et al., 2023). In this process, we set the guidance scale to 1 and perform 50 timesteps of DDIM inversion.

Baselines. We evaluate three representative latent-noise-based watermarking methods: Tree-ring (Wen et al., 2023), Gaussian Shading (Yang et al., 2024b), and DiffuseTrace (Lei et al., 2024). For Gaussian Shading, we test both implementations, with and without the ChaCha20 (Bernstein et al., 2008) secure stream cipher, which shuffles the watermark sequence. Detail of these methods are in the Appendix.

Evaluation Metrics. To evaluate watermark probe attacks, we use the area under the ROC curve (AUC). Attack results on watermarking methods indicate stealthiness, calculated as (1 - AUC of watermark probe attack). We benchmark watermark effectiveness by reporting AUC and TPR at 1% FPR (noted as TPR@1%FPR) and bit accuracy for encoded information. For watermarked image quality, we use the CLIP score (Radford et al., 2021) between generated images and prompts, measured using OpenCLIP-ViT/G (Cherti et al., 2023) and the Fréchet Inception Distance (FID) (Heusel et al., 2017). FID, which evaluates feature similarity between generated and original images, is calculated from 5,000 images per base model generated using the MS-COCO-2017 dataset(Lin et al., 2014).

Table 1: Comparison of SWA-LDM and baselines. The watermark effectiveness is evaluated with AUC, TPR@1%FPR, and bit accuracy. The quality of the generated images is assessed using FID and CLIP scores. The stealthiness represents the failure rate of the proposed watermark presence attacks. Left to right are LDMs fine-tuned from SD v1-5/SD v2-1/SDXL 1.0.

Methods	Nonce Management	Metrics					
		AUC	TPR@1%FPR	Bit Acc.	FID ↓	CLIP-Score ↑	Stealthiness ↑
No watermark	✗	-	-	-	29.77/27.01/75.83	0.324/0.291/0.304	-
Tree-Ring	✗	0.999/0.999/0.999	0.987/0.996/0.998	-	30.53/28.32/78.97	0.325/0.296/0.305	0.208/0.212/0.227
DiffuseTrace	✗	0.999/0.983/0.840	0.989/0.944/0.434	0.978/0.951/0.692	30.15/26.83/83.68	0.324/0.296/0.302	0.204/0.218/0.296
Gaussian Shading	✗	1.000/1.000/1.000	1.000/1.000/1.000	0.999/0.999/0.999	31.58/29.82/70.39	0.325/0.297/0.305	0.005/0.019/0.084
G-S$_{ChaCha20}$	✓	1.000/1.000/1.000	1.000/1.000/1.000	0.999/0.999/0.999	29.69/27.21/75.83	0.324/0.297/0.304	0.427/0.505/0.478
SWA-LDM (T-R)	✗	0.999/0.997/0.996	0.999/0.991/0.993	-	30.24/27.43/70.21	0.324/0.297/0.305	0.475/0.495/0.474
SWA-LDM (D-T)	✗	0.999/0.978/0.810	0.983/0.942/0.354	0.974/0.945/0.666	29.80/26.90/76.89	0.323/0.295/0.301	0.496/0.497/0.504
SWA-LDM (G-S)	✗	0.999/0.997/0.998	0.999/0.995/0.998	0.999/0.997/0.998	30.53/27.28/75.29	0.324/0.297/0.304	0.469/0.513/0.508

Setup of Watermark probe Attack. The attacker generates 1,000 clean images using three base models (SD v1-5, SD v2-1, SDXL 1.0) and evaluates performance by averaging results across models. For the watermark feature extractor, we use a 12-layer CNN with convolutional and fully connected layers, ReLU activations, and layer normalization, outputting a 100-dimensional feature vector. Training uses SGD optimizer with a learning rate of 0.01, momentum of 0.9, and a scheduler with a 0.5 decay factor every 50 steps. Detailed architecture is in the Appendix.

Setup of SWA-LDM. We integrate SWA-LDM with three baseline methods: SWA-LDM with Tree-Ring (SWA-LDM(T-R)), SWA-LDM with DiffuseTrace (SWA-LDM(D-T)), and SWA-LDM with Gaussian Shading (SWA-LDM(G-S)). Each method uses a key channel count of 1 to construct an 8-bit key with 64 redundant bits. The number of watermark channels is set to 1 for SWA-LDM(T-R) and 3 for both SWA-LDM(D-T) and SWA-LDM(G-S).

5.2 Comparison to Baseline Methods

Stealthiness Comparison. We conduct watermark probe attack experiments across SWA-LDM and baselines. The attack performance, summarized in the "Stealthiness" column of Tab. 1, shows the average stealthiness achieved by each method against an attacker using different base models.

The results show that watermark probe attacks effectively detect watermarks in baseline methods. SWA-LDM improves stealthiness and provides defense against these attacks. Among baseline methods, Gaussian Shading is the most detectable, with the lowest stealthiness, while DiffuseTrace and Tree-Ring offer slight improvements but remain vulnerable. Gaussian Shading with ChaCha20 increases stealthiness but requires costly per-image nonce management. In contrast, SWA-LDM achieves ChaCha20-level stealthiness without nonce dependency, integrating smoothly with DiffuseTrace, Tree-Ring, and Gaussian Shading. Further analysis on the base model's impact on detection is in Appendix C.

Watermarking Effectiveness Comparison. For the evaluation of watermark effectiveness, As detailed in Sec. 5.1, each base model (SD v1-5, SD v2-1, SDXL 1.0) is fine-tuned to produce 20 checkpoints, each generating 1,000 images, resulting in 60,000 watermarked and 60,000 clean images per method. As shown in Tab. 1, SWA-LDM maintains AUC, TPR@1%FPR, and bit accuracy comparable to original methods, with slight metric decreases due to key construction from latent noise for enhanced stealthiness. SWA-LDM also has minimal impact on FID and CLIP scores, preserving LDM-generated image quality.

6 Conclusion

In conclusion, we address critical vulnerabilities in latent-based watermarking methods for Latent Diffusion Models (LDMs) by exposing their susceptibility to detection through constant watermarks. We introduce a novel watermark probe attack that operates solely on generated images, setting a new standard in the field and highlighting the urgent need for enhanced watermarking strategies. To counter these vulnerabilities, we present SWA-LDM, a plug-and-play component that enables the creation of stealthy, image-dependent watermarks without incurring additional management costs. Comprehensive experiments validate the effectiveness of SWA-LDM in improving watermark stealthiness without compromising other watermark metrics.

REFERENCES

Daniel J Bernstein et al. Chacha, a variant of salsa20. In *Workshop record of SASC*, volume 8, pp. 3–5. Citeseer, 2008.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2829, June 2023.

Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital watermarking and steganography*. Morgan kaufmann, 2007.

I. Csiszar. I-Divergence Geometry of Probability Distributions and Minimization Problems. *The Annals of Probability*, 3(1):146 – 158, 1975. doi: 10.1214/aop/1176996454. URL https://doi.org/10.1214/aop/1176996454.

Manuel Eberl. Fisher–yates shuffle. *Archive of Formal Proofs*, September 2016. ISSN 2150-914x. https://isa-afp.org/entries/Fisher_Yates.html, Formal proof development.

Hugging Face. Huggingface: The ai community building the future. https://huggingface.co/.

Weitao Feng, Wenbo Zhou, Jiyan He, Jie Zhang, Tianyi Wei, Guanlin Li, Tianwei Zhang, Weiming Zhang, and Nenghai Yu. AquaLoRA: Toward white-box protection for customized stable diffusion models via watermark LoRA. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13423–13444. PMLR, 21–27 Jul 2024.

Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22466–22477, October 2023.

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

Gustavosta. Stable diffusion dataset. https://huggingface.co/datasets/Gustavosta/Stable-Diffusion-Prompts.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

Civit AI Inc. Civitai: The home of open-source generative ai. https://civitai.com/.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Liangqi Lei, Keke Gai, Jing Yu, and Liehuang Zhu. Diffusetrace: A transparent and flexible watermarking scheme for latent diffusion model, 2024. URL https://arxiv.org/abs/2405.02696.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PlKWVd2yBkY.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

Midjourney. Midjourney. https://www.midjourney.com/.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text Inversion for Editing Real Images using Guided Diffusion Models . In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6038–6047, Los Alamitos, CA, USA, June 2023. IEEE Computer Society. doi: 10.1109/CVPR52729.2023.00585. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.00585.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

J.J.K. Ó Ruanaidh, W.J. Dowling, and F.M. Boland. Watermarking digital images for copyright protection. *IEE Proceedings - Vision, Image, and Signal Processing*, pp. 250, Jan 1996. doi: 10.1049/ip-vis:19960711. URL http://dx.doi.org/10.1049/ip-vis:19960711.

Melissa E. O'Neill. Pcg: A family of simple fast space-efficient statistically good algorithms for random number generation. Technical Report HMC-CS-2014-0905, Harvey Mudd College, Claremont, CA, September 2014.

J.J.K. O'Ruanaidh and T. Pun. Rotation, scale and translation invariant digital image watermarking. In *Proceedings of International Conference on Image Processing*, Nov 2002. doi: 10.1109/icip.1997.647968. URL http://dx.doi.org/10.1109/icip.1997.647968.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Md Maklachur Rahman. A dwt, dct and svd based watermarking technique to protect the image piracy. *International Journal of Managing Public Sector Information and Communication Technologies*, 4(2):21, 2013.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

Mehrdad Saberi, Vinu Sankar Sadasivan, Keivan Rezaei, Aounon Kumar, Atoosa Chegini, Wenxiao Wang, and Soheil Feizi. Robustness of ai-image detectors: Fundamental limits and practical attacks. In *ICLR*, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020.

Tensor.art. Tensor.art. https://tensor.art/.

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-rings watermarks: invisible fingerprints for diffusion images. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Pei Yang, Hai Ci, Yiren Song, and Mike Zheng Shou. Can simple averaging defeat modern watermarks? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Zijin Yang, Kai Zeng, Kejiang Chen, Han Fang, Weiming Zhang, and Nenghai Yu. Gaussian shading: Provable performance-lossless image watermarking for diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12162–12171, 2024b. doi: 10.1109/CVPR52733.2024.01156.

Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *arXiv preprint arXiv:1909.01285*, 2019.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3813–3824, 2023. doi: 10.1109/ICCV51070.2023.00355.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator, 2023. URL https://arxiv.org/abs/2204.13902.

Yuqing Zhang, Yuan Liu, Zhiyu Xie, Lei Yang, Zhongyuan Liu, Mengzhou Yang, Runze Zhang, Qilong Kou, Cheng Lin, Wenping Wang, and Xiaogang Jin. Dreammat: High-quality pbr material generation with geometry- and light-aware diffusion models. *ACM Trans. Graph.*, 43(4), jul 2024. ISSN 0730-0301. doi: 10.1145/3658170. URL https://doi.org/10.1145/3658170.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: a unified predictor-corrector framework for fast sampling of diffusion models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.

A Experimental Details

All experiments are implemented using PyTorch 2.0.1 and the Diffusers 0.24.0 library, running on a single NVIDIA A800 GPU.

A.1 Baseline Methods

We detail the specific configurations of different baseline watermarking methods used in our experiments.

- For Tree-Ring (Wen et al., 2023), it embeds a carefully constructed watermark pattern in the Fourier space of the initial latent noise. Following the original paper, we set the watermark pattern to multiple concentric rings, where each ring maintains a constant value drawn from a Gaussian distribution. This design ensures rotation invariance and resilience against various image transformations while minimally deviating from an isotropic Gaussian distribution. The radius of the watermark pattern is set to 16 to balance generation quality and verification performance. We embed the watermark into one latent channel and vary the constant values along the rings to generate distinct watermarks.
- For Gaussian Shading (Yang et al., 2024b), we adopt the parameters recommended in the original paper to balance watermark capacity and robustness. Specifically, the watermark size is $1/8$ of the latent height, $1/8$ of the latent width, and one channel. For generated images with resolution $3 \times 512 \times 512$, the corresponding latent noise dimensions are $4 \times 64 \times 64$, resulting in watermark dimensions of $1 \times 8 \times 8$. During embedding, the watermark is redundantly replicated and inserted into three latent noise channels to enhance robustness.
- For DiffuseTrace (Lei et al., 2024), We use the publicly available code to obtain the DiffuseTrace Encoder-Decoder architecture, and pre-train the Encoder-Decoder to generate 3-channel latent noise (dimensions $3 \times 64 \times 64$) containing the DiffuseTrace watermark. To meet the input requirement of $4 \times 64 \times 64$ latent noise for Stable Diffusion (SD) models, we concatenate the 3-channel watermarked latent noise with a $1 \times 64 \times 64$ lantent noise

Table 2: Detailed architecture of the Watermark Feature Extractor. The table lists the parameters for each layer, including input channels, output channels, kernel size, stride, and activation function.

Layer	Type	Input Channels	Output Channels	Kernel Size	Stride
1	Conv2D	3	32	3×3	2
2	ReLU	-	-	-	-
3	Conv2D	32	32	3×3	1
4	ReLU	-	-	-	-
5	Conv2D	32	64	3×3	2
6	ReLU	-	-	-	-
7	Conv2D	64	64	3×3	1
8	ReLU	-	-	-	-
9	Conv2D	64	64	3×3	2
10	ReLU	-	-	-	-
11	Conv2D	64	128	3×3	2
12	ReLU	-	-	-	-
13	Conv2D	128	128	3×3	2
14	ReLU	-	-	-	-
15	Flatten	-	-	-	-
16	Dense	-	512	-	-
17	ReLU	-	-	-	-
18	LayerNorm	-	512	-	-
19	Dense	-	100	-	-
20	Sigmoid	-	-	-	-

sampled from a Gaussian distribution. To ensure compatibility with different SD models, we fine-tune the Encoder-Decoder for each specific SD model to initialize latent noise tailored to the model. Following the original implementation, the bit length of the watermark is set to 48 during both training and testing.

A.2 The Architecture of WFE

In Sec. 3.4, we have introduced the Watermark Feature Extractor (WFE). Here, we provide details of its architecture, as shown in Tab. 2. The WFE processes input images with a resolution of 256×256 through a series of 3×3 convolutional layers with stride 2, progressively reducing the spatial dimensions to an 8×8 feature map. Each convolutional layer is followed by a ReLU activation to introduce non-linearity. The resulting feature map is flattened and passed through two dense layers: the first projects it to a hidden dimension of 512, stabilized by LayerNorm, and the second produces a 100-dimensional watermark feature vector. A sigmoid activation function is applied to the output, ensuring that the values are in the range [0, 1], suitable for representing watermark features.

A.3 Evaluated Models

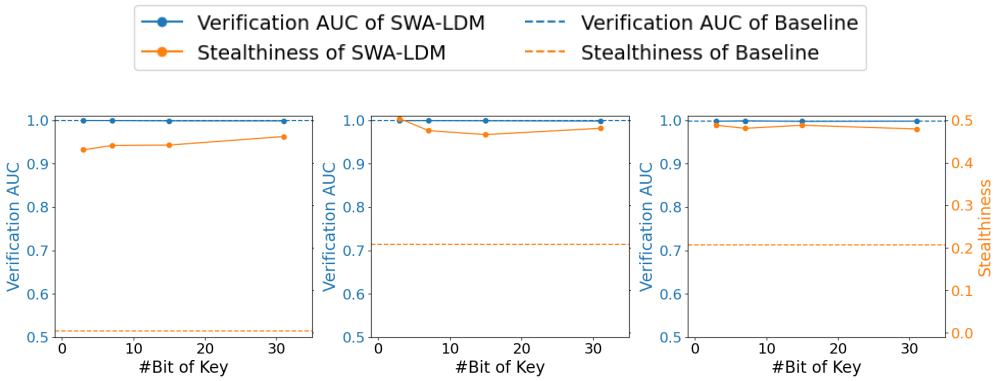
We utilized three widely-used Stable Diffusion models as base models: Stable Diffusion v1-5 (SD v1-5), Stable Diffusion v2-1 (SD v2-1), and SDXL 1.0-base (SDXL 1.0). Additionally, we downloaded 60 checkpoints from Hugging Face (Face), which include models fine-tuned from these base models or equipped with adapters. These customized models, comprising either fine-tuned versions or base models enhanced with adapters, were used in our experiments. A list of the adapters and fine-tuned models can be found in Tab. 3.

A.4 Image Perturbation Settings

In Appendix B, we evaluate the robustness of SWA-LDM against seven common image perturbations, which simulate potential attacks. The types of perturbations and their respective parameter ranges are detailed as follows:

Table 3: The names of the 60 checkpoints used in our experiment

Type	base on runwayml/stable-diffusion-v1-5	base on stabilityai/stable-diffusion-2-1	base on stabilityai/stable-diffusion-xl-base-1.0
Adapters	latent-consistency/lcm-lora-sdv1-5	sahibnanda/anime-night-vis-sd	alvdansen/BandW-Manga
	Melonie/text_to_image_finetuned	sahibnanda/anime-real-vis-night	nerijs/pixel-art-xl
	Kvikontent/midjourney-v6	dlcvproj/cartoon_sd_lora	latent-consistency/lcm-lora-sdxl
	h1t/TCD-SD15-LoRA	jainr3/sd-diffusiondb-pixelart-v2-model-lora	alvdansen/littletinies
	ostris/depth-of-field-slider-lora	dlcvproj/retro_sd_lora	Pclanglais/Mickey-1928
	Norod78/sd15-megaphone-lora	lora-library/lora-dreambooth-sample-dog	artificialguybr/ColoringBookRedmond-V2
	rocifier/painterly	lora-library/artdecodsgn	fofr/sdxl-emoji
	artificialguybr/pixelartredmond-1-5v-pixel-art-loras-for-sd-1-5	nakkati/output_dreambooth_model_preservation	alimama-creative/slam-lora-sdxl
	patrickvonplaten/lora_dreambooth_dog_example	Mousewrites/charturnerhn	Adrenex/chamana
	artificialguybr/stickers-redmond-1-5-version-stickers-lora-for-sd-1-5	lora-library/alf	alvdansen/midsommarcartoon
Finetunes	mhdang/dpo-sd1.5-text2image-v1	ptx0/pseudo-flex-v2	mhdang/dpo-sdxl-text2image-v1
	iamkaikai/amazing-logos-v2	Vishnou/sd-laion-art	Bakanayatsu/Pony-Diffusion-V6-XL-for-Anime
	stablediffusionapi/counterfeit-v30	n6ai/graphic-art	Bakanayatsu/ponyDiffusion-V6-XL-Turbo-DPO
	Bakanayatsu/cuteyukimix-Adorable-kemiaomiao	artificialguybr/freedom	Lykon/dreamshaper-xl-lightning
	iamanaiart/meinamix_meinaV11	bguisard/stable-diffusion-nano-2-1	Lykon/dreamshaper-xl-v2-turbo
	stablediffusionapi/maturemalemix-v14	cloudwithraj/dogbooth	Lykon/AAM_XL_AnimeMix
	Lykon/DreamShaper	bghira/pseudo-flex-v2	Linaqruf/animagine-xl-2.0
	Lykon/AnyLoRA	WildPress/simba_model	fluently/Fluently-XL-v4
	simbolo-ai/bagan	nishant-glance/model-sd-2-1-priorp-unet-2000-lr2e-ab	Eugeoter/artiwaifu-diffusion-1.0
	Lykon/AbsoluteReality	yuanbit/max-15-1e-6-1500	ehristoforu/Visionix-alpha



(a) SWA-LDM(G-S). (b) SWA-LDM(T-R). (c) SWA-LDM(D-T).

Figure 4: Performance of SWA-LDM with varying bit number of key. The effectiveness is demonstrated through AUC and stealthiness metrics, where (a) compares SWA-LDM (G-S) with Gaussian Shading and (b) compares SWA-LDM (T-R) with Tree-Ring. (c) compares SWA-LDM (D-T) with DiffuseTrace.

- JPEG Compression, where the image is compressed using quality factors (QF) set to {100, 90, 80, 70, 60, 50, 40, 30, 20, 10};
- Random Crop, which retains a randomly selected region covering {80%, 90%} of the original image area, discarding the rest;
- Random Drop, where randomly selected regions covering {10%, 20%, 30%, 40%, 50%} of the image area are replaced with black pixels;
- Resize and Restore (Resize), where the image is resized to {20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%} of its original dimensions and then restored to the original size;
- Gaussian Blur (GauBlur), applied with blur radii r set to {1, 2, 3, 4};
- Median Filter (MedFilter), using kernel sizes k of {1, 3, 5, 7, 9, 11};
- Brightness Adjustment, which modifies the image brightness using brightness factors {0, 2, 4, 6}.

B Benchmarking Watermark Robustness

To evaluate the robustness of SWA-LDM, we assess its performance under seven common image perturbations as potential attacks: JPEG compression, random crop, random drop, resize and restore(Resize), Gaussian blur (GauBlur), median filter (MedFilter), brightness adjustments. The parameter ranges are shown in the Appendix. For each parameter setting of every perturbation, we used 2,000 images generated by the SD v1-5 to evaluate performance. The average verification AUC for each perturbation is reported in Tab. 4, which compares the robustness of various

Table 4: Watermark Verification AUC under each image perturbation. Cr. & Dr. refers to random crop and random drop.

Methods	JPEG	Cr. & Dr.	Resize	GauBlur	MedFilter	Brightness	Avg
Tree-Ring	0.987	0.993	0.992	0.985	0.988	0.991	0.990
DiffuseTrace	0.962	0.993	0.985	0.966	0.969	0.922	0.968
Gaussian Shading	0.999	1.000	1.000	1.000	1.000	0.999	0.999
G-S$_{ChaCha20}$	0.999	1.000	1.000	1.000	1.000	0.999	0.999
SWA-LDM (T-R)	0.952	0.955	0.983	0.951	0.969	0.946	0.957
SWA-LDM (D-T)	0.939	0.974	0.977	0.939	0.950	0.965	0.959
SWA-LDM (G-S)	0.965	0.982	0.988	0.973	0.978	0.937	0.972

Table 5: Impact of different clean SD models on the watermark probe attacks. Left to right are target LDMs fine-tuned from SD v1-5/SD v2-1/SDXL 1.0.

Methods	SD version used to generate the clean images		
	SD v1-5	SD v2-1	SD-XL v1.0
Tree-ring	0.240/0.255/0.275	0.163/0.158/0.255	0.223/0.223/0.153
DiffuseTrace	0.183/0.229/0.303	0.207/0.213/0.303	0.223/0.213/0.284
Gaussian Shading	0.010/0.015/0.100	0.000/0.030/0.085	0.005/0.012/0.068
G-S$_{ChaCha20}$	0.445/0.546/0.500	0.383/0.400/0.435	0.453/0.570/0.500
SWA-LDM (T-R)	0.481/0.518/0.485	0.478/0.498/0.468	0.465/0.470/0.470
SWA-LDM (D-T)	0.478/0.528/0.520	0.491/0.484/0.463	0.520/0.479/0.530
SWA-LDM (G-S)	0.438/0.475/0.528	0.500/0.515/0.495	0.468/0.548/0.503

watermarking methods, both with and without the integration of SWA-LDM. Results indicate that SWA-LDM maintains robust watermark verification under moderate image perturbations, demonstrating its robustness. However, incorporating SWA-LDM impacts the original robustness of these watermarking methods, especially under high-intensity distortions. This occurs because SWA-LDM requires complete recovery of each bit in the key to retrieve the watermark, which can reduce robustness. Nevertheless, unless the image undergoes quality-compromising levels of perturbation, watermark remains practical.

C Ablation Studies

Impact of the clean SD model on watermark probe attack. We evaluated whether the effectiveness of the watermark probe attack is influenced by the base model used by the attacker to generate clean images. Results shown in Tab. 5 indicate that the choice of base model has minimal impact on attack performance, demonstrating that the watermark detection attack remains effective without requiring knowledge related to the target model.

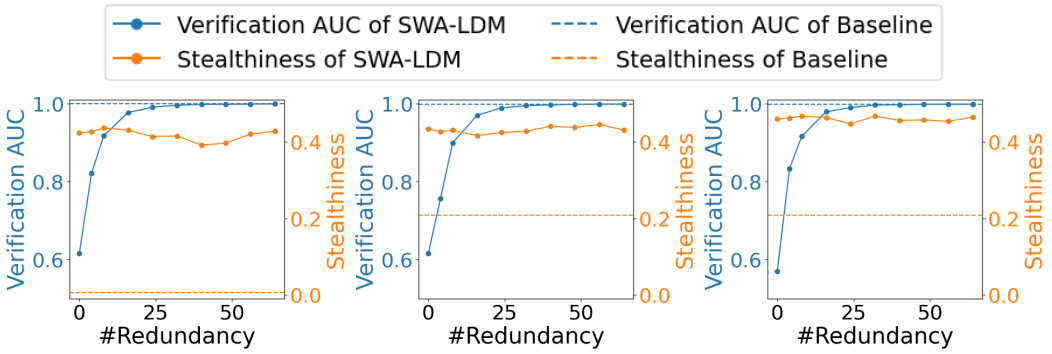
Impact of image quantity on watermark probe attack. Following the setup in Sec. 5.1, we varied the number of images generated by the watermark probe attacker to assess its effect on attack performance. Results shown in Tab. 6, indicate that within our sampled range, the watermark probe attack's effectiveness remains stable regardless of image quantity.

Impact of key redundancy on stealthiness and verification performance. Following the setup in Section 5.1, we evaluate how varying key redundancy levels affects watermark stealthiness and verification AUC. Results in Fig. 5 show that with minimal redundancy (4 redundancies), SWA-LDM achieves a verification AUC around 0.8, compared to a near-perfect verification AUC of 1 for watermarking methods without SWA-LDM, indicating an 80% key recovery success rate. As redundancy increases to 8, the recovery probability improves to 90%, and with redundancy over 40, SWA-LDM achieves near-complete key recovery without compromising verification AUC. Across all redundancy levels, SWA-LDM maintains consistently high stealthiness.

Impact of bit number of key on stealthiness and verification performance. In Sec. 4.2, we have introduced how SWA-LDM employs a pseudorandom number generator (PRNG) and a shuffle

Table 6: Impact of image quantities on the watermark probe attacks. Results are shown as stealthiness.

Methods	Clean Image Quantity			
	500	1,000	1,500	2,000
Tree-ring	0.256/0.331/0.194	0.208/0.212/0.227	0.212/0.172/0.214	0.224/0.172/0.186
DiffuseTrace	0.220/0.208/0.325	0.204/0.218/0.296	0.244/0.204/0.288	0.221/0.214/0.300
Gaussian Shading	0.039/0.014/0.081	0.005/0.019/0.084	0.011/0.004/0.071	0.049/0.008/0.076
G-S$_{ChaCha20}$	0.423/0.468/0.438	0.427/0.505/0.478	0.438/0.486/0.533	0.423/0.478/0.546
SWA-LDM (T-R)	0.440/0.459/0.521	0.475/0.495/0.474	0.413/0.480/0.515	0.509/0.483/0.454
SWA-LDM (D-T)	0.491/0.530/0.470	0.496/0.497/0.504	0.525/0.475/0.500	0.498/0.516/0.479
SWA-LDM (G-S)	0.428/0.480/0.527	0.469/0.513/0.508	0.410/0.485/0.520	0.500/0.456/0.528



(a) SWA-LDM(G-S). (b) SWA-LDM(T-R). (c) SWA-LDM(D-T).

Figure 5: Performance of SWA-LDM with varying numbers of redundancies. The effectiveness is demonstrated through AUC and stealthiness metrics, where (a) compares SWA-LDM (G-S) with Gaussian Shading and (b) compares SWA-LDM (T-R) with Tree-Ring. (c) compares SWA-LDM (D-T) with DiffuseTrace.

algorithm to randomize the watermarked latent noise. The key is derived from the latent noise and serves as the seed for the PRNG. To analyze the impact of key bit number on watermark stealthiness and verification performance, we have further evaluated SWA-LDM across a range of key lengths from 4 to 32 bits, based on the setup described in Sec. 5.1. Results in Fig. 4, indicate that both watermark stealthiness and verification AUC remain consistent regardless of the key's bit number within this range. These findings suggest that the choice of key length does not compromise the effectiveness or concealment of the watermark, providing flexibility in the design of the key construction process.

Impact of key construction. As described in Sec. 4.2, the key is constructed by sampling each bit from the sign of specific latent variables. To assess its importance, we have replaced this mechanism with fixed latent variables as the key. Following the experimental setup in Sec. 5.1, the results in Tab. 7 show that removing the key construction significantly degrades performance. As analyzed in Sec. 4.2, during diffusion inversion, the reconstructed latent noise may not perfectly match the original latent noise, especially when the image experiences perturbations. These mismatches prevent accurate key reconstruction, making watermark verification infeasible. This underscores the critical role of key construction in maintaining robust watermarking.

Impact of sampling methods. We tested five commonly used sampling methods. As shown in Tab. 8, our method demonstrates stable watermark verification AUC across different sampling methods.

Impact of inversion step. In practice, the specific denoising step used in generation is often unknown, which can result in a mismatch with the inversion step. However, as shown in Tab. 9, this step mismatch does not affect the performance of our watermarking approach.

Table 7: Impact of key construction on watermark verification AUC. The table compares results with (✓) and without (✗) the proposed key construction mechanism. Left to right are LDMs fine-tuned from SD v1-5/SD v2-1/SDXL 1.0.

Key Construction	**Watermark Methods**		
	SWA-LDM (T-R)	SWA-LDM (D-T)	SWA-LDM (G-S)
✗	0.517/0.532/0.521	0.482/0.495/0.500	0.491/0.502/0.496
✓	0.999/0.997/0.996	0.999/0.978/0.810	0.999/0.997/0.998

Table 8: Verification AUC with different sampling methods, including DDIM Song et al. (2020), UniPC Zhao et al. (2024), PNDM Liu et al. (2022), DEIS Zhang & Chen (2023), and DPMSolver Lu et al. (2022); Song et al. (2020).

Watermark Methods	**Sampling Methods**				
	DDIM	UniPC	PNDM	DEIS	DPMSolver
SWA-LDM (T-R)	1.000	0.972	1.000	1.000	1.000
SWA-LDM (D-T)	0.999	0.999	0.999	1.000	0.999
SWA-LDM (G-S)	1.000	1.000	1.000	1.000	1.000

Table 9: Verification AUC of SWA-LDM (T-R)/SWA-LDM (D-T)/SWA-LDM (G-S) with different denoising and inversion step.

Denoising Step	**Inversion Step**			
	10	25	50	100
10	0.999/0.999/1.000	1.000/1.000/1.000	1.000/1.000/0.999	1.000/1.000/0.999
25	1.000/0.999/1.000	1.000/0.999/1.000	1.000/1.000/1.000	1.000/1.000/1.000
50	1.000/1.000/1.000	1.000/1.000/1.000	1.000/1.000/1.000	1.000/1.000/1.000
100	1.000/1.000/1.000	1.000/1.000/1.000	1.000/1.000/1.000	1.000/1.000/1.000