A. Appendix of Paper "Towards Robust Content Watermarking Against Removal and Forgery Attacks"

A.1. Experimental Details

Watermarking hyperparameter assignment. After gaining the label $y^c \in [C]$ for C-classes clustering, we need to assign corresponding injection timestep t and pattern location offset l. Specifically, we denote the injection timestep range in $[T_1, T_2]$, the pattern location offset $l = (l_x, l_y)$ range in $[l_x^1, l_x^2] \times [l_y^1, l_y^2]$. The injection timestep t is set as

$$t = T_1 + [y^c \bmod (T_2 - T_1)],$$

the pattern position offset is set as

$$l_x = l_x^1 + \left[y^c \bmod (l_x^2 - l_x^1)\right], l_y = l_y^1 + \left[\left(y^c // (l_y^2 - l_y^1)\right) \bmod (l_y^2 - l_y^1)\right].$$

The watermarking pattern after location offset should be

$$\text{Offset}(W, l)_{i,j} = W_{i+l_x, j+l_y}.$$

In this paper, we set the clustering class $C = 1024$, the timestep range in $[10, 20]$, and the pattern position offset be $[-12, 12]^2$ by default.

Watermark injection. After getting the watermarking pattern $W^o = \text{Offset}(W, l)$, we add the watermarking pattern into the t-step latent z_t. Building upon with the method proposed by Wen et al. [58], denote the watermarking mask in the frequency domain be M, M is also deployed with location offset l similar as W, denoted as

$$M^o_{i,j} = M_{i+l_x, j+l_y}.$$

The watermarking channel is 0 and the radius is set to be 20. Then we replace the corresponding pixel $k = (i_k, j_k)$ of z_t by W^o under mask M^o in the frequency domain, i.e.

$$\mathcal{F}(z_t)_k = \begin{cases} W^o_k, & \text{if } k \in M^o, \\ \mathcal{F}(\mathcal{M}_{T \to t}(p, z_T))_k, & \text{otherwise.} \end{cases}$$

Finally, the watermarked t-step latent z_t^w is obtained by inverse Fourier transformation:

$$z_t^w = \mathcal{F}^{-1}(\mathcal{F}(z_t)),$$

after the modification on $\mathcal{F}(z_t)$.

To make the notation simple, in our Algorithm 2, we donate the above injection process as

$$z_t^w = z_t \oplus \text{Offset}(W, l).$$

Parameter selector. For a new prompt p_{new} and the corresponding generated image I_{new}, we need to assign its watermarking parameter with our parameter selector $f = h \circ g$, where g is a pre-trained CLIP feature extractor, and f is the classifier trained by existing features and their assigned labels.

To reduce the computational complexity, we use a very simple two-layer neural network to align extracted features with their watermarking parameters, where the hidden dimension is same as the input feature dimension, and apply ReLU activation, Batch Normalization with $p = 0.5$ dropout rate.

A.2. Details on Removal and Forgery Attacks

Imp-Removal. In Müller et al. [41], they try to remove watermarks by optimizing the follow equation:

$$\mathcal{L}(\delta) = \|\mathcal{M}_{0 \to T}(p, z_0^w + \delta) + z_T^w\|_2,$$

where $\mathcal{M}$ is the surrogate diffusion model, p is the text prompt, $z_0^w = \mathcal{E}(x^w)$ derive from the obtained watermarked image x^w with encoder $\mathcal{E}$, $z_T^w = \mathcal{M}_{0 \to T}(p, z_0^w)$ is an estimation of T-step latent after DDIM inversion from z_0^w. Tne Imp-Removal

attack aims to induce watermarked latent to the opposite position in timestep T, which could be vulnerable to our two-sided detection. Finally, the image after Imp-Removal attack $\hat{x}^w$ is decoded by decoder $\mathcal{D}$ from the perturbed latent:

$$\hat{x}^w = \mathcal{D}(z_0^w + \delta).$$

Following the setting provided in Müller et al. [41], we set the optimization steps be 150, with learning rate be 0.01. Furthermore, it is noteworthy that, Müller et al. [41] considers the surrogate model scenario, where the attack model be Stable-Diffusion 2.1 (SD 2.1), and the victim models be different, like SD 2.1-Anime (SD 2.1 Finetuned on Anime), Stable Diffusion XL [44], etc. In this paper, we consider a stronger case for attackers, that they can use the same diffusion model as surrogate, inducing a white-box rather than grey-box scenario. As our evaluation based on this white-box scenario, the robustness of our watermarking will be not bad than those from grey-box scenario.

Imp-Forgery. In Müller et al. [41], they want to forge existing watermarking method by optimizing the following loss:

$$\mathcal{L}(\delta) = \|\mathcal{M}_{0\to T}(p, z_0^c + \delta) - z_T^w\|_2,$$

where $z_0^c = \mathcal{E}(x^c)$ derive from the clean image x^c ready to be forged, $z_T^w = \mathcal{M}_{0\to T}(p, z_0^w)$ is an estimation of T-step latent after DDIM inversion from z_0^w, and $z_0^w = \mathcal{E}(x^w)$ derive from the obtained watermarked image x^w. Other notations are similar to Imp-Removal. Similarly, the optimization steps are 150 with learning rate is 0.01 following the setting of Müller et al. [41]. We also evaluate on the white-box scenario in Imp-Forgery attack. The image after Imp-Forgery attack $\hat{x}^w$ is decoded by decoder $\mathcal{D}$ from the perturbed latent:

$$\hat{x}^w = \mathcal{D}(z_0^c + \delta).$$

Avg-Removal and Avg-Forgery. In Yang et al. [60], they find that watermarking patterns can be revealed by averaging a collection of watermarked and non-watermarked images. Specifically, they propose averaging over N images and calculating their residual as:

$$\delta = \frac{1}{N}\left(\sum_{i=1}^{N} x_{w,i} - \sum_{i=1}^{N} x_{c,i}\right),$$

where $\{x_{c,i}\}_{i\in[N]}$ are clean non-watermarked images, $\{x_{w,i}\}_{i\in[N]}$ are watermarked images.
In Avg-Removal attack, they modify the watermarked image x^w to $\hat{x}^w$ by

$$\hat{x}^w = x^w - \delta,$$

where in Avg-Forgery attacks, they modify the clean image x^c to $\hat{x}^w$ by

$$\hat{x}^w = x^c + \delta.$$

In this paper, we first generate 100 pairs of non-watermarked and watermarked images to extract the residual δ, then conduct Avg-Removal and Avg-Forgery attacks for each watermarked/non-watermarked image.

VAE-Removal. In Jain et al. [26], similar to Müller et al. [41], they try to remove and forge watermarks with a single watermarked image. Unlike Müller et al. [41] uses a surrogate diffusion model to optimize perturbations in the T-step noise latent space, Jain et al. [26] directly optimize attacks on latent image space with only the surrogate VAE encoder. Specifically, in VAE-Removal attack, they optimize injected noise δ by:

$$\min_{\delta} \|\mathcal{E}(x^w + \delta) - \mathcal{E}(\mu_{x^w})\|_2 + \lambda\|\delta\|_2,$$

where x^w is the watermarked image ready to be removed, $\mathcal{E}$ is the VAE encoder, λ is the balancing hyperparameter, and μ_{x^w} is the plain image with all values equal to the mean of the watermarked image x^w.
In this paper, followed the setting of Jain et al. [26], we choose the VAE encoder from Stable Diffusion v1.4, and set λ be 5×10^4.

VAE-Forgery. Similar to VAE-Removal attacks, VAE-Forgery proposed by Jain et al. [26] also optimize the perturbation at the latent image space. Specifically, VAE-Forgery works as:

$$\min_{\delta} \|\mathcal{E}(x^c + \delta) - \mathcal{E}(x^w)\|_2 + \lambda\|\delta\|_2,$$

where x^c is the clean non-watermarked image ready to be forged, x^w is the obtained watermarked image, other notations are similar to VAE-Removal.
Same as VAE-Removal, the VAE encoder is from Stable Diffusion v1.4, and the hyperparameter $\lambda = 5 \times 10^4$.

A.3. Different Intermediate Steps for Watermarking Injection

To further investigate the impact of intermediate steps for watermarking injection, we evaluate different ranges of injection steps from $t = 5$ to $t = 45$, the overall diffusion steps is $T = 50$. As our watermarking injection dynamically select $t \in [10, 20]$, to ensure the consistency, we evaluate different steps with the same range of variation, from $[5, 15]$ to $[35, 45]$. We use Imp-Removal and Imp-Forgery as the removal and forgery attacks, results are shown in Figure 5.
It reveals that although the detection AUC remains sufficiently high for original watermarking without attacks across different intermediate steps, the performance under removal and forgery attacks have changed considerably. As steps go larger, Detection AUC becomes worse for both removal and forgery attacks, implying a relative small injection steps when designing robust watermarking.

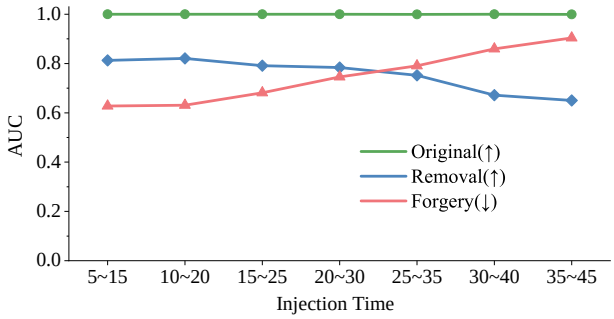

Figure 5. Different injection steps.

A.4. Potential Side-Channel Attacks on ISTS

A simple side-channel attack of our ISTS watermarking is the potential leakage of the public CLIP extractors and even the parameter selector model. However, we believe that information extracted from CLIP features or the parameter selector model of potentially leaked watermarked images does not compromise the security of our ISTS watermarking scheme. This is because the assigned labels y_p^c can be obfuscated using a (pseudo-)random permutation $\mathcal{R}$ controlled by a secret key. Given a large number of clusters (e.g., 1024), it becomes infeasible for an attacker to recover the injection parameters (t, l), as they cannot infer the permuted label $\mathcal{R}(y_p^c)$ because 2^{1024} is a infeasible large number for any attackers. Moreover, in practical deployments, watermarking methods are typically applied to proprietary models, where adversaries are limited to black-box access via APIs and do not have visibility into the model's internal structures or algorithms.