BEYOND SINGLE WATERMARKS: ADAPTIVE TREE-RING WATERMARK COEXISTENCE FOR SEQUENTIAL EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Digital watermarks are essential for provenance tracking, ownership attribution, and misuse deterrence in AI-generated imagery. In realistic creative pipelines, images often undergo multiple editing rounds, where new watermarks may need to be embedded while earlier provenance signals remain detectable. Existing coexistence studies mainly focus on training-based encoder-decoder schemes, leaving training-free Tree-Ring-style watermarks underexplored despite their capacity, model compatibility, and deployment flexibility. In this paper, we study **multi-watermark coexistence** for continual image editing based on Tree-Ring-style watermarks. We introduce **SeqEdit-BM**, a multi-round editing benchmark for evaluating whether multiple watermarks can be repeatedly embedded, preserved, and independently detected across sequential edits. Using this benchmark, we show that Tree-Ring-style watermarks have strong **intrinsic compatibility** but suffer **progressive attenuation** during iterative diffusion reconstruction. We attribute this attenuation to trajectory mismatch between watermark injection and detection, and propose an adaptive watermarking method based on trajectory alignment to calibrate the watermark template before injection. We further formulate two allocation protocols for multi-watermark coexistence: single-slot multiplexing, where multiple marks share one spatial-spectral slot, and deterministic slot assignment, where keys map to a predefined partition of non-overlapping latent-frequency slots, with collisions allowed under blind writing. Experimental results on verification and identification tasks demonstrate that our method substantially improves multi-watermark coexistence and robustness against sequential editing, maintaining strong detection performance after five consecutive rounds while preserving image quality with no perceptible artifacts. These results suggest that Tree-Ring-style watermarks offer a practical foundation for provenance preservation in continual image editing.

1 INTRODUCTION

Digital watermarking has become a mature security primitive for supporting provenance tracking in AI-generated media. Most existing methods verify a single watermark in text (Kirchenbauer et al., 2023a;b; Liu et al., 2024; Pan et al., 2024), images (Fernandez et al., 2023; Wen et al., 2023; Li et al., 2025), or audio (Hua et al., 2016; Chen et al., 2023). This single-watermark setting does not cover collaborative AI media. As an image moves through model providers, editing tools, platforms, and end users, several parties may need to attach their own provenance records to the same asset ((C2PA), 2024). We call this requirement multi-watermark coexistence: multiple watermarks should remain independently detectable and attributable after later edits. Recent legislative proposals in major jurisdictions have also begun framing watermarking and disclosure for AI-generated content as legal obligations (U.S. Congress, 2023; Wicks, 2024), further increasing the need for reliable provenance across editing workflows (Petrov et al., 2026).

Prior work (Petrov et al., 2026) establishes the feasibility of coexistence among training-based watermarks. However, existing coexistence studies have not fully addressed the long-term preservation of training-free watermarks through sequential generative image editing. Tree-Ring-style watermarks (Wen et al., 2023; Ci et al., 2024; Li et al., 2025) are a good fit for multi-watermark co-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

existence because they place key-dependent patterns in the frequency domain of diffusion noise. That frequency space provides large watermark capacity, and the watermark can be added without retraining the editing model. However, capacity alone is not enough. Our empirical analysis shows that repeated editing and inversion gradually weaken the recovered frequency pattern, making progressive signal attenuation the main bottleneck for long-term coexistence (Figure 1).

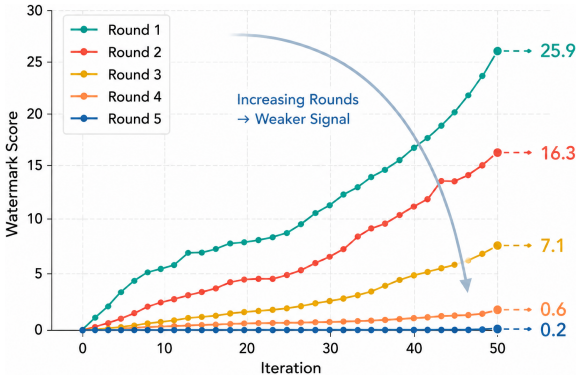

Figure 1: Watermark attenuation over five editing rounds. Higher watermark scores indicate a greater likelihood of successful detection.

We address this bottleneck with an adaptive Tree-Ring watermark that locally calibrates the pattern before embedding. The detector retains only the key, reconstructs a watermark-bearing noise latent, applies empty-text inversion, and compares the resulting frequency reference with the inverted query. To support multiple watermarks, we assign key-dependent patterns to predefined slots in the frequency-domain space. We introduce **SeqEdit-BM**, a benchmark adapted from ImgEdit (Ye et al., 2026), with five-step editing trajectories for standardized evaluation of iterative image watermarking. On SeqEdit-BM, we test two blind-writing protocols: deterministic slot assignment and single-slot multiplexing. Deterministic slot assignment selects a slot independently at each round, allowing collisions with earlier watermarks. Single-slot multiplexing places all watermarks in the same slot to directly test coexistence under repeated collisions. We also test two writing settings: blind writing, where the current editor has no watermark-side information, and informed writing, a controlled protocol in which it knows only the occupied watermark-bearing channels, not the previous keys or patterns. In summary, we make the following contributions:

- We introduce **SeqEdit-BM**, a dataset for multi-turn image editing. Its progressive and temporally coherent editing trajectories provide a standardized testbed for watermark coexistence.
- We evaluate blind watermark coexistence using deterministic slot assignment and single-slot multiplexing, together with a separate informed-writing setting. These protocols test tolerance to slot collisions and the effect of access to occupied-channel information.
- We identify the signal attenuation bottleneck inherent in existing Tree-Ring variants. To preserve watermark strength, we propose an **adaptive watermark** with a trajectory alignment strategy that counteracts watermark degradation.
- Extensive experiments on the SeqEdit-BM dataset demonstrate that our adaptive watermark achieves superior compatibility and robustness. It improves target-watermark detectability across successive edits under the evaluated protocols.

2 Related Work and Preliminaries

The injection of watermarks into generative models has a long history as a means to protect intellectual property and enable source attribution. In the context of diffusion models, watermarking approaches can be broadly categorized into two paradigms: training-based methods (Tang & Aoki, 1997; Wang et al., 1998; Al-Haj, 2007; Zhu et al., 2018) and training-free methods (Wen et al., 2023; Ci et al., 2024; Li et al., 2025; Arabi et al., 2024), exemplified by the Tree-Ring watermark. Training-based methods learn an encoder-decoder pipeline that embeds watermark information into the pixel, latent, or feature space while jointly optimizing imperceptibility and robustness to common distortions. While training-free Tree-Ring watermarks directly embeds a specific pattern into the frequency domain of the initial noise, which can be reliably detected by reversing the generation process. We have included a detailed introduction of these methods in Appendix G.

2.1 COEXISTENCE OF WATERMARKS

Most diffusion watermarking methods assume a single-watermark setting, in which each image carries only one watermark. However, this assumption does not adequately reflect practical provenance pipelines, where an image may pass through multiple parties and accumulate multiple watermark signals. Recent work has explored related multi-watermark settings. RingID (Ci et al., 2024) studies multi-key identification, where the detector identifies a single embedded key from a candidate set. Petrov et al. (Petrov et al., 2026) formally investigate watermark coexistence and show that watermarks from different schemes generally coexist more reliably than multiple watermarks from the same scheme. Their study primarily focuses on training-based methods and two-round coexistence, leaving the long-term preservation of training-free watermarks through sequential generative image editing insufficiently explored. We therefore study sequential multi-watermark coexistence. At each editing round, a party adds a new watermark, either without coordination or with access only to occupied channel information from previous rounds.

2.2 NOTATION

We use t for diffusion timesteps and $i \in \{1, \dots, N\}$ for editing rounds. A latent with c channels and spatial size $H \times W$ has a frequency representation in $\mathbb{C}^{H \times W \times c}$. Its channel set is $\mathcal{C} = \{0, \dots, c-1\}$; watermark-bearing channels $\mathcal{C}_w \subseteq \mathcal{C}$ are each partitioned into b annular slots. A watermark key is denoted by w, the null key by $w_\emptyset$, and the candidate set by $\mathcal{W} = \{w_1, \dots, w_K\}$. Fixed public configuration Γ specifies deterministic noise reconstruction $z(w; \Gamma)$, mask $M(w; \Gamma)$, and reference inversion R_θ (Section 4.2.1). For an image X, DDIM inversion recovers $x_{\text{inv}} = D_\theta^\dagger(X)$, and $s(X, w)$ scores key w. We use τ_i for a calibrated key-specific threshold (with a shared τ as a special case), $\delta_w(X)$ for verification, and $\text{ID}(\cdot)$ for identification. Let $X^{(0)}$ be the initial image and $X^{(i)}$ the output after instruction $\pi^{(i)}$ embeds key w_i. The available watermark information is $\mathcal{I}^{(i)}$, while $\mathcal{C}_{\text{occ}}^{(i-1)}$ denotes the set of occupied watermark-bearing channels known to the current party. The party does not observe previous keys, exact patterns, templates, or detector records. We write the editing pipeline as Ψ_θ. At round i, $x_t^{(i)}$ is the injection latent, $x_f^{(i)} = \mathcal{F}(x_t^{(i)})$ its Fourier representation, $\hat{x}_t^{(i)}$ the watermarked latent, and $x_{\text{inv}}^{(i)}$ the noise recovered from $X^{(i)}$.

2.3 DIFFUSION MODELS

For a data sample $x_0 \sim q(x)$, forward diffusion adds Gaussian noise over T steps:

$$q(x_t \mid x_0) = \mathcal{N}\big(x_t;\ \sqrt{\bar{\alpha}_t}\, x_0,\ (1 - \bar{\alpha}_t)\mathbf{I}\big), \tag{1}$$

where $\bar{\alpha}_t = \prod_{\ell=1}^{t}(1 - \beta_\ell)$ and $\beta_t \in (0, 1)$ is the variance schedule. Equivalently, $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$, with $\epsilon \sim \mathcal{N}(0, \mathbf{I})$.

Reverse Process. A noise predictor $\epsilon_\theta(x_t, t)$ gives the clean-sample estimate $\hat{x}_0 = (x_t - \sqrt{1 - \bar{\alpha}_t}\, \epsilon_\theta(x_t, t))/\sqrt{\bar{\alpha}_t}$. The deterministic DDIM update is

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\, \hat{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1}}\, \epsilon_\theta(x_t, t). \tag{2}$$

Starting from $x_T \sim \mathcal{N}(0, \mathbf{I})$, the complete denoising map is $x_0 = D_\theta(x_T)$.

Inversion Process. DDIM inversion approximately recovers the initial noise from x_0 through

$$x_{t+1} = \sqrt{\bar{\alpha}_{t+1}}\, \hat{x}_0 + \sqrt{1 - \bar{\alpha}_{t+1}}\, \epsilon_\theta(x_t, t), \tag{3}$$

yielding the approximate inverse mapping $x_T = D_\theta^\dagger(x_0)$.

2.4 FRAMEWORK FORMULATION

We model multi-turn watermarking as sequential watermark-aware editing. At round i,

$$X^{(i)} = \Psi_\theta\Big(X^{(i-1)}, \pi^{(i)}, w_i; \mathcal{I}^{(i)}\Big), \mathcal{I}^{(i)} = \begin{cases} \emptyset, & \text{blind writing,} \\ \mathcal{C}_{\text{occ}}^{(i-1)}, & \text{informed writing.} \end{cases} \tag{4}$$

The final image $X^{(N)}$ should preserve the accumulated key set $\{w_i\}_{i=1}^N$, which we evaluate using supplied candidate keys and fixed public configuration.

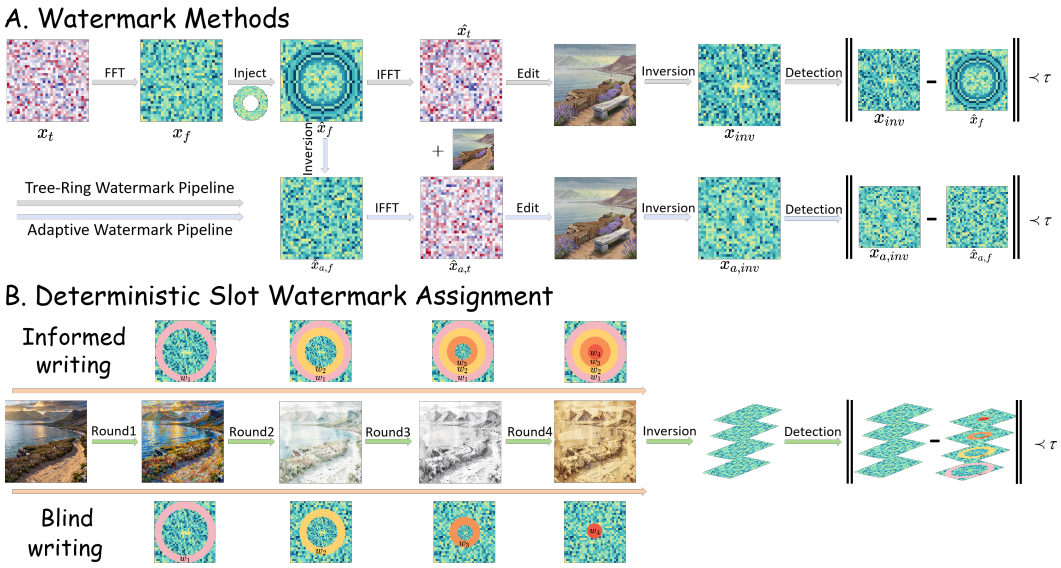

Figure 2: **A brief pipeline of the proposed adaptive watermarking scheme and multi-watermark framework.** The adaptive watermark calibrates an injection pattern by inverting a temporary watermarked image. Detection reconstructs a watermark-bearing noise latent from the key and applies empty-text inversion to obtain its reference (Section 4.2.1). The multi-watermark framework assigns different watermarks to radius-based annular slots in the frequency domain for independent verification.

3 Threat Model

We study provenance watermarking for multi-turn AI image editing. An image may pass through several editing services, platforms, or creators, and a new watermark may be added at each round. The final image should preserve all embedded watermarks so that each provenance claim can be verified independently. We consider two practical writing settings: *blind writing* and *informed writing*.

Watermarking Party. The watermarking party controls one editing round and embeds an imperceptible provenance signal into the edited output. This party may be a model owner, an editing service, a downstream platform, or a creator using an image editing tool. Across a sequence of edits, the same image may therefore accumulate multiple watermarks from different parties.

In the *blind writing* setting, the current editing party embeds its watermark independently, without knowing which watermarks are already present. With deterministic slot assignment, it samples from the full slot set without checking occupancy. We also test single-slot multiplexing, where every round uses the same slot, to evaluate coexistence under repeated collisions. In the *informed writing* setting, the current party knows which watermark-bearing channels are occupied through provenance information. This is a controlled upper bound, rather than a fully realistic information-sharing protocol. Only channel occupancy is disclosed, excluding prior keys, frequency patterns, phases, amplitudes, templates, and detector records. This allows channel-level collision avoidance without revealing the information needed to reproduce earlier watermarks.

Detector. Given an edited image and candidate watermark records, the detector tests whether each candidate watermark is present. The detector has access to the watermarking algorithm and the editing model. It does not rely on additional side information such as the original editing prompt, editing instructions, or guidance strength. The goal is to verify multiple embedded watermarks from the edited image without access to those editing conditions.

Attacker. The attacker aims to cause embedded watermarks to fail verification while preserving the usability of the edited image. To weaken watermark signals without substantially degrading perceptual quality, the attacker may apply transformations such as recompression and geometric distortions (Cheng et al., 2020). We also consider an adaptive averaging attacker who knows both

the watermarking algorithm and the verification rule, and may craft targeted perturbations to reduce selected scores. It exploits knowledge of the target watermark and detector to estimate a shared signal from multiple samples, then removes the estimated signal to construct a forgery.

4 METHOD

In this section, we present the injection and detection procedures of our adaptive watermark, followed by deterministic slot assignment for multi-watermark coexistence.

4.1 DESIGN RATIONALE

Tree-Ring watermarks suffer from attenuation under repeated editing and limited multi-watermark coexistence. RingID Ci et al. (2024) links detectability to distribution shifts from discarding the imaginary component after inverse Fourier transformation, while GaussMarker Li et al. (2025) identifies vulnerability to reconstruction attacks. We further observe that increasing latent channels reduces per-watermark energy and distorts the recovered frequency structure, alongside progressive attenuation across editing rounds (Figure 1). During inversion, our adaptive watermark exhibits steadily increasing cosine similarity and decreasing masked L_2 distance between the template and recovered noise, whereas Wind improves only slightly (Figure 3). We have provided a brief pipeline in Figure 2.

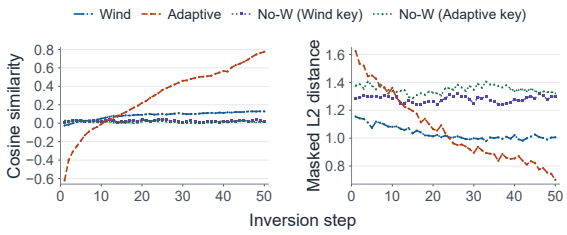

Figure 3: Retained inversion trajectories over 50 steps, averaged over 1,000 samples.

4.2 ADAPTIVE TREE-RING WATERMARK

At round i, we temporarily watermark $X^{(i-1)}$ using instruction $\pi^{(i)}$, then invert it with the same instruction to obtain $w_{\mathrm{a},f}^{(i)}$. This image-dependent tensor is used only for reinsertion; detection reconstructs a separate reference from w_i and Γ without retaining the tensor.

Let $M^{(i)} = M(w_i;\Gamma) \in \{0,1\}^{H\times W\times c}$ denote the key-derived mask and $\mathcal{F}$ the Fourier transform. Given diffusion noise $x_t^{(i)}$ with spectrum $x_f^{(i)} = \mathcal{F}(x_t^{(i)})$, we deterministically construct $p_f^{(i)} = P_{\mathrm{base}}(z(w_i;\Gamma);\Gamma)$. We inject this pattern into the masked region as $\tilde{x}_t^{(i)} = \mathcal{F}^{-1}\Big((1-M^{(i)})\odot x_f^{(i)} + M^{(i)}\odot p_f^{(i)}\Big)$ and generate the temporary image $X_{\mathrm{tmp}}^{(i)} = \mathcal{E}_\theta\Big(X^{(i-1)}, \pi^{(i)}, \tilde{x}_t^{(i)}\Big)$. Inverting this image with the same instruction yields the calibrated pattern $w_{\mathrm{a},f}^{(i)} = \mathcal{F}\Big(D_\theta^\dagger\Big(X_{\mathrm{tmp}}^{(i)};\pi^{(i)}\Big)\Big)$. We then reinsert this pattern into the original spectrum:

$$\hat{x}_t^{(i)} = \mathcal{F}^{-1}\Big((1-M^{(i)})\odot x_f^{(i)} + M^{(i)}\odot w_{\mathrm{a},f}^{(i)}\Big). \tag{5}$$

The final edited image is generated as $X^{(i)} = \mathcal{E}_\theta\Big(X^{(i-1)}, \pi^{(i)}, \hat{x}_t^{(i)}\Big)$. Our method follows the progression from Tree-Ring's frequency-domain embedding (Wen et al., 2023), through RingID's multi-key pattern design (Ci et al., 2024), to Wind's noise-matching approach (Arabi et al., 2024). Building on this foundation, we retain Wind-style matching and introduce local calibration and reinsertion together with inversion-based reference construction for sequential editing. Wind is therefore the immediate methodological baseline; Tree-Ring and RingID provide comparisons with earlier designs in this line of work.

4.2.1 WATERMARK DETECTION

Given a query image X, prompt-free inversion yields $x_{f,\mathrm{inv}} = \mathcal{F}\Big(D_\theta^\dagger(X;\pi_{\mathrm{det}}=\emptyset,\Gamma)\Big)$. Following Tree-Ring's seeded construction (Wen et al., 2023), each key w_i reconstructs Gaussian noise z_i,

mask $M^{(i)}$, and base pattern $p_f^{(i)}$. We insert the base pattern into the masked spectrum of z_i to obtain the watermark-bearing latent u_i. The detection reference is

$$r_f(w_i;\Gamma) = \mathcal{F}(R_\theta(u_i;\pi_{\text{ref}} = \emptyset,\Gamma)), \tag{6}$$

where R_θ applies prompt-free latent inversion directly to u_i using the query-inversion schedule.

We apply Wind-style sliding-window cosine matching (Arabi et al., 2024) to the masked query and reference: $c_i = \text{SlidingCos}\big(x_{f,\text{inv}}[M^{(i)}], r_f(w_i;\Gamma)[M^{(i)}]\big)$. The score and decision are

$$s(X,w_i) = \text{Top}_\rho\text{-Mean}(c_i), \qquad \delta_{w_i}(X) = \mathbb{1}[s(X,w_i) > \tau_i]. \tag{7}$$

We set $\rho = 3$, averaging the three largest window scores. Thresholds τ_i are calibrated on 1,000 unwatermarked editing outputs from separately sampled ImgEdit source images and held fixed during evaluation. Calibration and evaluation share no source images or editing trajectories; Appendix B.1 gives details.

4.3 Deterministic Slot Assignment Framework

To evaluate uncoordinated coexistence under blind writing, we define non-overlapping slots $\mathcal{J} = \{1,\dots,|\mathcal{C}_w|b\}$ across watermark-bearing channels $\mathcal{C}_w$, with b frequency bins per channel. Each round independently samples a key w_i and selects $j_i = \text{Slot}(w_i;\Gamma)$ from all slots through a fixed mapping. Slots remain eligible without occupancy checks, so collisions are allowed. The selected channel and frequency ring define the mask $\mathcal{M}$, restricting injection to that slot:

$$z_w = \mathcal{F}^{-1}\Big((1-\mathcal{M}) \odot \mathcal{F}(z_0) + \mathcal{M} \odot P\Big). \tag{8}$$

The image is generated as $I_w = G_\Theta(z_w, p_t)$. Only the key is saved; the slot, mask, and frequency reference are reconstructed from it using Γ.

A complementary single-slot stress test maps every key to one fixed slot.

Table 1: Coexistence results under the blind writing setting using the Deterministic Slot Assignment framework with Flux. Each entry is reported as AUC / TPR@1%FPR / ALL Detection Rate. The best value for each metric within the same editing round and evaluation condition is highlighted in yellow and bold.

Rounds	Methods	Clean	JPEG	Blur	Noise	Brightness	PSNR	SSIM
2	*Tree-Ring*	0.927 / 0.629 / 0.396	0.858 / 0.537 / 0.296	0.534 / 0.082 / 0.004	0.590 / 0.145 / 0.025	0.842 / 0.425 / 0.200	**35.060**	**0.977**
	RingID	0.982 / 0.830 / 0.700	0.933 / 0.591 / 0.368	0.590 / 0.113 / 0.014	0.655 / 0.136 / 0.021	0.893 / 0.455 / 0.236	33.833	0.972
	GaussMarker	0.960 / 0.677 / 0.468	0.899 / 0.518 / 0.279	0.556 / 0.059 / 0.004	0.687 / 0.150 / 0.025	0.873 / 0.450 / 0.175	18.471	0.710
	Wind	0.940 / 0.727 / 0.507	0.898 / 0.627 / 0.386	0.529 / 0.029 / 0.000	0.746 / 0.255 / 0.107	0.842 / 0.400 / 0.182	31.746	0.954
	Ours	**0.987 / 0.950 / 0.900**	**0.981 / 0.943 / 0.886**	**0.776 / 0.329 / 0.132**	**0.923 / 0.675 / 0.489**	**0.965 / 0.904 / 0.832**	31.555	0.952
3	*Tree-Ring*	0.857 / 0.436 / 0.093	0.873 / 0.337 / 0.032	0.559 / 0.057 / 0.000	0.572 / 0.050 / 0.000	0.808 / 0.275 / 0.018	**32.172**	**0.970**
	RingID	0.958 / 0.819 / 0.557	0.921 / 0.576 / 0.246	0.580 / 0.057 / 0.000	0.636 / 0.056 / 0.000	0.859 / 0.410 / 0.107	31.030	0.965
	GaussMarker	0.920 / 0.512 / 0.143	0.849 / 0.400 / 0.082	0.555 / 0.060 / 0.000	0.640 / 0.087 / 0.000	0.812 / 0.249 / 0.007	14.648	0.577
	Wind	0.896 / 0.619 / 0.250	0.860 / 0.469 / 0.121	0.632 / 0.074 / 0.000	0.728 / 0.155 / 0.007	0.810 / 0.446 / 0.157	28.195	0.922
	Ours	**0.970 / 0.917 / 0.818**	**0.963 / 0.883 / 0.746**	**0.788 / 0.315 / 0.036**	**0.871 / 0.625 / 0.332**	**0.943 / 0.843 / 0.671**	28.012	0.919
4	*Tree-Ring*	0.851 / 0.396 / 0.014	0.823 / 0.246 / 0.000	0.512 / 0.026 / 0.000	0.567 / 0.048 / 0.000	0.782 / 0.329 / 0.004	**29.940**	**0.963**
	RingID	0.913 / 0.497 / 0.068	0.884 / 0.556 / 0.114	0.521 / 0.063 / 0.000	0.637 / 0.103 / 0.000	0.867 / 0.361 / 0.021	28.871	0.956
	GaussMarker	0.881 / 0.379 / 0.029	0.818 / 0.268 / 0.007	0.544 / 0.023 / 0.000	0.617 / 0.078 / 0.000	0.812 / 0.360 / 0.032	12.584	0.473
	Wind	0.804 / 0.413 / 0.032	0.834 / 0.476 / 0.064	0.547 / 0.033 / 0.000	0.699 / 0.150 / 0.004	0.769 / 0.271 / 0.004	25.738	0.889
	Ours	**0.959 / 0.868 / 0.675**	**0.957 / 0.876 / 0.679**	**0.736 / 0.268 / 0.014**	**0.875 / 0.539 / 0.179**	**0.941 / 0.821 / 0.561**	25.398	0.883
5	*Tree-Ring*	0.776 / 0.240 / 0.000	0.799 / 0.186 / 0.000	0.528 / 0.019 / **0.000**	0.519 / 0.031 / 0.000	0.781 / 0.265 / 0.000	**28.141**	**0.954**
	RingID	0.894 / 0.552 / 0.064	0.829 / 0.441 / 0.014	0.494 / 0.062 / **0.000**	0.576 / 0.066 / 0.000	0.789 / 0.215 / 0.000	27.136	0.947
	GaussMarker	0.847 / 0.351 / 0.007	0.795 / 0.236 / 0.000	0.549 / 0.033 / **0.000**	0.597 / 0.080 / 0.000	0.764 / 0.229 / 0.000	11.136	0.408
	Wind	0.825 / 0.439 / 0.011	0.794 / 0.324 / 0.004	0.554 / 0.039 / **0.000**	0.671 / 0.114 / 0.000	0.787 / 0.274 / 0.004	23.837	0.852
	Ours	**0.944 / 0.841 / 0.561**	**0.941 / 0.816 / 0.496**	**0.739 / 0.216 / 0.000**	**0.837 / 0.530 / 0.157**	**0.905 / 0.758 / 0.407**	23.584	0.850

5 Experiments

In this section, we evaluate the proposed deterministic slot assignment framework under both the Informed writing and blind writing settings. We compare it against the single-channel multiplexing framework on four baselines as well as our proposed adaptive method. The informed-writing results are reported separately in Appendix D.2.

5.1 EXPERIMENTAL SETUP

Settings and data. We evaluate FLUX.2-Klein-4B (Labs, 2025), FireRed-Image-Edit (Team et al., 2026), and InstructPix2Pix (Brooks et al., 2023) at 512×512 resolution. The framework derives its capacity from each model's latent dimensions; unless stated otherwise, the ring width is 16. We construct SEQEDIT-BM from 280 randomly sampled ImgEdit images (Ye et al., 2026).

Baselines and implementation. We compare the complete pipelines of Tree-Ring (Wen et al., 2023), RingID (Ci et al., 2024), GaussMarker (Li et al., 2025), and Wind (Arabi et al., 2024) using their official implementations. Wind is the immediate baseline for our extension; the other methods establish performance across alternative and earlier designs. For each method, we evaluate 280 watermarked and unwatermarked images across five rounds. Generation and inversion use 50 DDIM steps. Wind and our detector share sliding-window cosine matching with a window of 256 and stride of 64. Our detector averages the three largest window scores ($\rho = 3$).

All experiments are repeated three times, and we report the mean results across the three runs.

Metrics. Following RingID (Ci et al., 2024), we report ROC-AUC and TPR at 1% FPR for verification and candidate-key ranking. Threshold calibration uses 1,000 unwatermarked editing outputs generated from source images randomly sampled from ImgEdit, disjoint from the 280 evaluation source images and their editing trajectories. Thresholds remain fixed during evaluation. We also report All-Detection Rate, the fraction of trajectories for which every watermark embedded by a given round is detected at that round. This metric counts missed true keys but does not penalize false-positive keys (Appendix E). PSNR (Hore & Ziou, 2010) and SSIM (Wang et al., 2004) measure image quality.

Attacks. Following Tree-Ring, we evaluate JPEG compression at quality 25, Gaussian blur with an 8×8 kernel, additive Gaussian noise with deviation 0.1, and brightness adjustment in $[0, 6]$.

Table 2: Watermark coexistence results under the blind writing setting with the Deterministic Slot Assignment framework, using FireRed-Image-Edit as the image editing model. Ring width is fixed to 16 for all entries. Each entry is reported as AUC / TPR@1%FPR / ALL detection rate. The best value for each metric within the same editing round and evaluation condition is highlighted in yellow and bold.

Rounds	Methods	Clean	JPEG	Blur	Noise	Brightness	PSNR	SSIM
2	*Tree-Ring*	0.965 / 0.855 / 0.710	0.964 / 0.780 / 0.600	0.698 / 0.125 / 0.030	0.716 / 0.225 / 0.070	0.903 / 0.745 / 0.600	30.201	0.915
	GaussMarker	0.982 / 0.960 / 0.920	0.981 / **0.945** / **0.890**	0.748 / **0.360** / 0.130	0.778 / 0.315 / 0.160	0.963 / 0.860 / 0.780	24.499	0.841
	Wind	0.965 / 0.875 / 0.760	0.942 / 0.745 / 0.510	0.712 / 0.160 / 0.011	0.724 / 0.170 / 0.039	0.901 / 0.630 / 0.430	26.826	0.886
	Ours	**0.992** / **0.970** / **0.940**	**0.984** / **0.945** / **0.890**	**0.774** / 0.315 / **0.150**	**0.907** / **0.580** / **0.380**	**0.966** / **0.875** / **0.800**	**31.204**	**0.953**
3	*Tree-Ring*	0.924 / 0.727 / 0.450	0.887 / 0.617 / 0.280	0.660 / 0.147 / 0.010	0.656 / 0.150 / 0.010	0.832 / 0.480 / 0.180	27.058	0.888
	GaussMarker	0.944 / 0.863 / 0.740	0.936 / 0.813 / 0.620	0.713 / **0.247** / **0.030**	0.687 / 0.203 / 0.050	0.905 / **0.767** / **0.570**	21.565	0.801
	Wind	0.889 / 0.657 / 0.350	0.883 / 0.597 / 0.290	0.650 / 0.077 / 0.000	0.654 / 0.160 / 0.011	0.849 / 0.450 / 0.140	23.979	0.854
	Ours	**0.963** / **0.887** / **0.760**	**0.954** / **0.833** / **0.670**	**0.740** / 0.200 / 0.020	**0.861** / **0.463** / **0.140**	**0.924** / 0.763 / 0.550	**27.130**	**0.907**
4	*Tree-Ring*	0.869 / 0.520 / 0.080	0.806 / 0.453 / 0.050	0.634 / 0.100 / 0.000	0.585 / 0.070 / 0.000	0.832 / 0.383 / 0.020	**25.407**	0.860
	GaussMarker	0.925 / 0.767 / 0.480	0.894 / 0.728 / 0.450	0.653 / **0.225** / **0.010**	0.613 / 0.165 / 0.020	0.897 / 0.690 / 0.350	20.193	0.768
	Wind	0.877 / 0.630 / 0.250	0.855 / 0.520 / 0.140	0.620 / 0.077 / 0.000	0.559 / 0.087 / 0.000	0.839 / 0.502 / 0.130	22.336	0.820
	Ours	**0.951** / **0.833** / **0.590**	**0.948** / **0.775** / **0.460**	**0.716** / 0.147 / 0.000	**0.840** / **0.472** / **0.120**	**0.935** / **0.770** / **0.490**	24.446	**0.866**
5	*Tree-Ring*	0.826 / 0.482 / 0.070	0.784 / 0.372 / 0.010	0.626 / 0.104 / **0.000**	0.603 / 0.124 / 0.000	0.775 / 0.472 / 0.030	**24.288**	**0.857**
	GaussMarker	0.906 / 0.748 / 0.410	0.864 / 0.626 / 0.240	0.634 / **0.186** / **0.000**	0.641 / 0.110 / 0.000	0.861 / 0.580 / 0.190	19.303	0.764
	Wind	0.819 / 0.332 / 0.000	0.800 / 0.336 / 0.000	0.629 / 0.154 / **0.000**	0.580 / 0.044 / 0.000	0.779 / 0.408 / 0.070	21.192	0.813
	Ours	**0.938** / **0.794** / **0.500**	**0.933** / **0.676** / **0.320**	**0.674** / 0.144 / **0.000**	**0.850** / **0.422** / **0.070**	**0.900** / **0.660** / **0.290**	22.829	0.836

5.2 COMPARISON RESULTS

Watermark compatibility. The single-slot-multiplexing results are reported in Appendix D.1, Table 5, while the deterministic-slot results appear in Table 1. Specifically, under a two-round successive editing scenario, RingID, GaussMarker, Wind, and our Adaptive watermark all achieve AUC scores exceeding 0.92. Remarkably, even after four rounds of editing, these methods retain AUC scores near 0.80, underscoring their exceptional robustness to iterative modifications. Table 2 reports the corresponding blind-writing results on FireRed-Image-Edit.

Blind-writing robustness. The robustness of our Adaptive watermark can be analyzed along two dimensions. First, successive editing inherently acts as a form of image compression and recon-

struction, during which watermark degradation is unavoidable. As shown in Table 1 under the deterministic slot assignment framework, the detection AUC of Tree-Ring watermarks drops from 0.927 in the second round to 0.776 by the fifth round. In stark contrast, our Adaptive watermark exhibits significantly slower decay, declining only marginally from 0.987 to 0.944 over the same period—a trend consistent across all experimental settings. Second, under blind writing with deterministic slot assignment on FLUX, our adaptive watermark improves AUC by an average of 12.5 percentage points over the strongest baseline in each evaluated condition, averaging across rounds 2–5 and four perturbations(Table 1).

Relative to Wind, our immediate baseline, the adaptive method better preserves watermark detectability across editing rounds. Under blind writing with deterministic slot assignment, Round-5 AUC increases from 0.825 to 0.944, while TPR@1%FPR rises from 0.439 to 0.841 and All-Detection Rate from 0.011 to 0.561. Together with the trajectory diagnostics in Figure 3, these results support the benefit of our adaptive injection within the Wind-style matching framework.

Coexistence under slot collisions. Single-slot multiplexing places all watermarks in the same latent channel and frequency slot across editing rounds. Table 5 reports the results under this repeated-collision setting with Flux. After five rounds on clean images, our method achieves an AUC of 0.978, TPR@1%FPR of 0.907, and All-Detection Rate of 0.700. The corresponding results under deterministic slot assignment are 0.944, 0.841, and 0.561 (Table 1). Thus, forcing all watermarks into one slot did not reduce these metrics relative to random placement in this setting. These results support tolerance to repeated slot collisions and show that the observed coexistence does not require collision-free allocation.

Identification performance. With 64 candidate keys, our adaptive watermark achieves Round-5 AUCs of 0.874 on clean images and 0.875 under JPEG compression. The strongest baselines reach 0.778 and 0.763, respectively. Appendix D.3 provides the complete curves, table, and analysis.

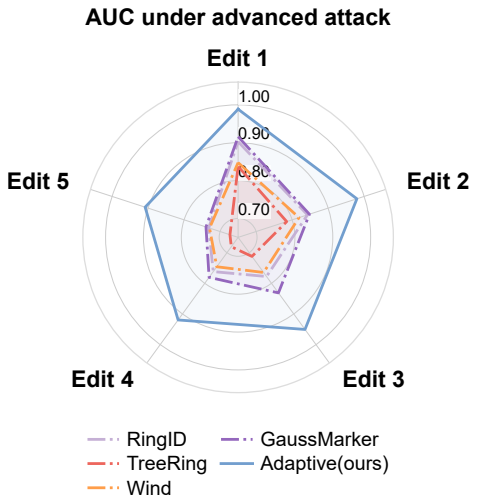

Figure 4: AUC under compression attacks.

5.3 Advanced Attack

Average attack. The averaging attack of Yang et al. (Yang et al., 2024) assumes that all images share an identical watermark template. By averaging a large number of inverted noises, the random noise component is canceled out while the watermark signal is preserved, enabling watermark recovery.

As shown in Figure 5, our adaptive watermark demonstrates substantially stronger resistance to averaging-based attacks than all baseline methods. Under the removal attack, our method achieves an AUC of 0.999, maintaining nearly perfect verification performance after the attacker attempts to estimate and subtract the watermark template. Under the more challenging forgery attack, our method still attains an AUC of 0.968, significantly outperforming all competing approaches. These results indicate that the watermark instances embedded by our method cannot be reliably recovered through averaging, effectively preventing both removal and forgery attacks.

Compression attack. Following the methodologies of Gaussmarker (Li et al., 2025), we employ pre-trained VAE compressors (Cheng et al., 2020) to perform the encoding and decoding process. As illustrated in Fig. 4, the Adaptive watermark demonstrates superior performance compared to other baselines, effectively overcoming reconstruction attacks, which were previously considered a significant vulnerability (Li et al., 2025; Umrajkar & Singh, 2025).

5.4 Ablation Study

Detection margin relative to Wind. Our detector retains Wind-style sliding-window cosine matching and uses the inversion-derived reference in Equation 6. Figure 6 compares the cosine similarity

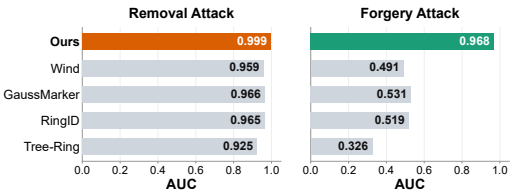

Figure 5: AUC under the averaging attack.

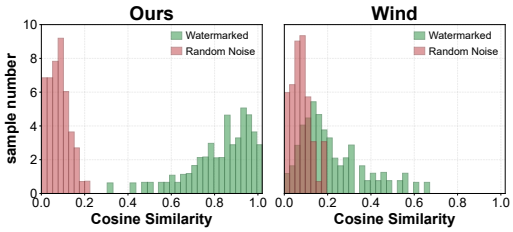

Figure 6: Cosine similarities for 100 samples.

Table 3: Ablation study of the adaptive watermark. (a) Candidate-key set size for identification, with ring width fixed at 16. (b) Ring width. All entries report AUC. At each editing round, AUC remains largely unchanged across the evaluated candidate-key set sizes and ring widths.

(a) Candidate-key set size

Keys	Round	Clean	JPEG	Blur	Noise	Bright.
32	2	0.960	0.963	0.659	0.841	0.911
	3	0.946	0.929	0.730	0.807	0.912
	4	0.875	0.923	0.627	0.714	0.919
	5	0.885	0.877	0.547	0.674	0.830
64	2	0.959	0.965	0.696	0.800	0.931
	3	0.949	0.941	0.738	0.604	0.919
	4	0.916	0.878	0.617	0.882	0.926
	5	0.874	0.875	0.598	0.808	0.802
128	2	0.961	0.959	0.697	0.800	0.934
	3	0.954	0.935	0.734	0.729	0.885
	4	0.871	0.872	0.611	0.677	0.897
	5	0.877	0.855	0.589	0.766	0.858

(b) Ring width

Width	Round	Clean	JPEG	Blur	Noise	Bright.
8	2	0.986	0.982	0.752	0.898	0.962
	3	0.973	0.963	0.738	0.888	0.947
	4	0.962	0.946	0.731	0.878	0.939
	5	0.945	0.939	0.687	0.879	0.919
16	2	0.987	0.981	0.776	0.923	0.965
	3	0.970	0.963	0.788	0.871	0.943
	4	0.959	0.957	0.736	0.875	0.941
	5	0.944	0.941	0.739	0.837	0.905
32	2	0.986	0.981	0.768	0.893	0.961
	3	0.973	0.960	0.723	0.862	0.944
	4	0.964	0.957	0.721	0.858	0.939
	5	0.943	0.936	0.693	0.857	0.909

distributions of 100 randomly selected samples for Wind and our method against random noise. Our method exhibits greater separation between watermarked and random-noise scores, consistent with improved detectability from the adaptive pipeline under the shared matching rule.

Candidate-key set size. To assess the effect of candidate-set size on identification, we evaluate 32, 64, and 128 candidate keys. Table 3(a) shows that identification AUC remains broadly stable as the candidate set grows. These results indicate that our method maintains reliable candidate-key ranking across the evaluated search spaces.

Ring width. Ring width controls watermark energy and embedding capacity, both of which affect detection accuracy. We evaluate ring widths of 8, 16, and 32 using FLUX.2-Klein-4B. Table 3(b) reports the results across editing rounds. Clean-image AUC remains broadly stable across the evaluated ring widths, indicating limited sensitivity to this parameter within the tested range.

6 Conclusion

This paper addresses multi-watermark provenance in sequential image editing, where new watermarks must be added without erasing earlier ones. We identify the limitations of single-watermark and Tree-Ring watermark designs under blind writing, and introduce SeqEdit-BM, two coexistence frameworks, and an adaptive watermark that aligns latent frequency signals with diffusion inversion trajectories. Our experiments show that Tree-Ring-style watermarks can coexist more reliably than training-based schemes, but still suffer progressive attenuation during iterative reconstruction. By calibrating the template along the local DDIM inversion path before injection, our adaptive watermark reduces this drift and improves robustness under informed and blind writing, reconstruction attacks, and common post-processing. These results support practical provenance for generative AI systems in which images move across multiple services or creators.

Reproducibility Statement

Sections 4.2 and 4.2.1 and Appendix B describe injection and the revised key-based detection data flow. Section 5.1 and Appendix D retain the reported experimental settings and results. Appendix F documents SeqEdit-BM construction and the trajectory-generation prompt. We will release the code after acceptance.

AI Use Statement

We used Gemini 2.5 to propose multi-turn image-editing instructions for the SeqEdit-BM construction and used generative AI tools to assist with language editing of this manuscript. AI-generated instructions and text were reviewed by the authors, who verified the dataset constraints, experimental descriptions, equations, tables, and claims.

References

Ali Al-Haj. Combined dwt-dct digital image watermarking. *Journal of computer science*, 3(9): 740–746, 2007.

Kasra Arabi, Benjamin Feuer, R Teal Witter, Chinmay Hegde, and Niv Cohen. Hidden in the noise: Two-stage robust watermarking for images. *arXiv preprint arXiv:2412.04653*, 2024.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18392–18402, 2023.

Tu Bui, Shruti Agarwal, Ning Yu, and John Collomosse. Rosteals: Robust steganography using autoencoder latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 933–942, 2023.

(C2PA). Content credentials: C2PA technical specification, version 2.4. Technical report, 2024. URL https://spec.c2pa.org/specifications/specifications/2.4/specs/C2PA_Specification.html. Section 2.3.4, 10.3.2.2.

Guangyu Chen, Yu Wu, Shujie Liu, Tao Liu, Xiaoyong Du, and Furu Wei. Wavmark: Watermarking for audio generation, 2023.

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7939–7948, 2020.

Hai Ci, Pei Yang, Yiren Song, and Mike Zheng Shou. Ringid: Rethinking tree-ring watermarking for enhanced multi-key identification. In *European conference on computer vision*, pp. 338–354. Springer, 2024.

Tianshuo Cong, Xinlei He, and Yang Zhang. SSLGuard: A watermarking scheme for self-supervised learning pre-trained encoders. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 579–593. ACM, 2022. doi: 10.1145/3548606.3559355. URL https://doi.org/10.1145/3548606.3559355.

Tianshuo Cong, Delong Ran, Zesen Liu, Xinlei He, Jinyuan Liu, Yichen Gong, Qi Li, Anyu Wang, and Xiaoyun Wang. Have you merged my model? on the robustness of large language model IP protection methods against model merging. In *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis*, pp. 69–76. ACM, 2024. doi: 10.1145/3689217.3690614. URL https://doi.org/10.1145/3689217.3690614.

Google DeepMind. Gemini 2.5. https://deepmind.google/technologies/gemini/, 2025.

Pierre Fernandez, Alexandre Sablayrolles, Teddy Furon, Hervé Jégou, and Matthijs Douze. Watermarking images in self-supervised latent spaces. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3054–3058, 2022.

Pierre Fernandez, Guillaume Couairon, Herv
'e Jegou, Teddy Furon, and Matthijs Douze. The stable signature: Rooting watermarks in latent diffusion models. *arXiv preprint arXiv:2303.15435*, 2023.

Xinlei He, Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. MGTBench: Benchmarking machine-generated text detection. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2251–2265. ACM, 2024. doi: 10.1145/3658644.3670344. URL https://doi.org/10.1145/3658644.3670344.

Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pp. 2366–2369. IEEE, 2010.

Guang Hua, Jiwu Huang, Yun Q Shi, Jonathan Goh, and Vrizlynn LL Thing. Twenty years of digital audio watermarking—a comprehensive review. *Signal processing*, 128:222–242, 2016.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning (ICML)*, pp. 17061–17084. PMLR, 2023a.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634*, 2023b.

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. https://bfl.ai/blog/flux-2, 2025.

Kecen Li, Zhicong Huang, Xinwen Hou, and Cheng Hong. Gaussmarker: Robust dual-domain watermark for diffusion models. *arXiv preprint arXiv:2506.11444*, 2025.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip S. Yu. A survey of text watermarking in the era of large language models. *ACM Computing Surveys*, 2024.

Leyi Pan, Aiwei Liu, Zheng He, Zhiwei Gao, Xunxiang Zhao, Yijian Lu, Boren Zhou, Shuo Liu, Xuming Hu, Lijie Wen, Irwin King, and Philip S. Yu. Markllm: An open-source toolkit for llm watermarking. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, 2024.

Aleksandar Petrov, Shruti Agarwal, Philip Torr, Adel Bibi, and John Collomosse. On the coexistence and ensembling of watermarks. *Advances in Neural Information Processing Systems*, 38:107058–107094, 2026.

Yiting Qu, Xinyue Shen, Xinlei He, Michael Backes, Savvas Zannettou, and Yang Zhang. Unsafe diffusion: On the generation of unsafe images and hateful memes from text-to-image models. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3403–3417. ACM, 2023. doi: 10.1145/3576915.3616679. URL https://doi.org/10.1145/3576915.3616679.

Weili Tang and Yoshinao Aoki. A dct-based coding of images in watermarking. In *Proceedings of ICICS, 1997 International Conference on Information, Communications and Signal Processing. Theme: Trends in Information Systems Engineering and Wireless Multimedia Communications (Cat.*, volume 1, pp. 510–512. IEEE, 1997.

Super Intelligence Team, Changhao Qiao, Chao Hui, Chen Li, Cunzheng Wang, Dejia Song, Jiale Zhang, Jing Li, Qiang Xiang, Runqi Wang, Shuang Sun, Wei Zhu, Xu Tang, Yao Hu, Yibo Chen, Yuhao Huang, Yuxuan Duan, Zhiyi Chen, and Ziyuan Guo. Firered-image-edit-1.0 technical report, 2026. URL https://arxiv.org/abs/2602.13344.

Ved Umrajkar and Aakash Kumar Singh. Detection limits and statistical separability of tree ring watermarks in rectified flow-based text-to-image generation models. *arXiv preprint arXiv:2504.03850*, 2025.

U.S. Congress. Advisory for AI-Generated Content Act. https://www.govinfo.gov/app/details/BILLS-118s2765is/summary, 2023. S. 2765, 118th Cong., 1st Sess.

Houng-Jyh Mike Wang, Po-Chyi Su, and C-C Jay Kuo. Wavelet-based digital image watermarking. *Optics Express*, 3(12):491–496, 1998.

Kaibo Wang, Jianda Mao, Tong Wu, and Yang Xiang. Towards a golden classifier-free guidance path via foresight fixed point iterations. *Advances in Neural Information Processing Systems*, 38: 124994–125030, 2026.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *arXiv preprint arXiv:2305.20030*, 2023.

B. Wicks. California Digital Content Provenance Standards. `https://legiscan.com/CA/text/AB3211/id/2984195`, 2024. Cal. Assem. Bill No. 3211, 2023–2024 Reg. Sess.

Pei Yang, Hai Ci, Yiren Song, and Mike Zheng Shou. Can simple averaging defeat modern watermarks? In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Yang Ye, Xianyi He, Zongjian Li, Shenghai Yuan, Zhiyuan Yan, Bohan Hou, Li Yuan, et al. Imgedit: A unified image editing dataset and benchmark. *Advances in Neural Information Processing Systems*, 38, 2026.

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

Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 657–672, 2018.

APPENDIX OVERVIEW

The appendix is organized as follows:

- Appendix A gives the formal task formulation for watermark verification and identification.
- Appendix B gives the complete watermark assignment and detection algorithms.
- Appendix C provides additional analysis of Tree-Ring watermark coexistence.
- Appendix D reports supplementary evaluations, including single-slot multiplexing results in Appendix D.1, informed-writing results in Appendix D.2, identification experiments in Appendix D.3.
- Appendix E discusses robustness and image-quality evaluation limitations.
- Appendix F describes the construction of SEQEDIT-BM.
- Appendix G summarizes additional related work.

A TASK FORMULATION

We follow the verification and identification settings of Tree-Ring (Wen et al., 2023) and RingID (Ci et al., 2024), and formalize their extension to the multi-watermark setting below.

Watermark verification. Given an image X and candidate key w, the verifier recovers x_{inv} and applies

$$\begin{aligned} H_1 &: X \text{ contains } w, \\ H_0 &: X \text{ does not contain the tested watermark key } w, \\ \delta_w(X) &= \mathbb{1}[s(X, w) > \tau]. \end{aligned} \tag{9}$$

Here, H_0 includes any image without the tested key, including images carrying other keys. For threshold calibration, however, we use only the 1,000 unwatermarked negative images described in Section 4.2.1. The revised verifier receives a candidate key and Γ, reconstructs its noise and mask, and obtains the reference through Equation 6. It tests each supplied key independently in a one-versus-rest manner; no frequency template is supplied.

Watermark identification. Given X and candidate set $\mathcal{W}$, identification ranks candidate keys by the scores $s(X, w_k)$ to recover one key or a subset:

$$\begin{aligned} \mathrm{ID}(X) &= \arg\max_{w_k \in \mathcal{W}} s(X, w_k), \\ \mathrm{ID}_{\mathrm{multi}}(X) &= \{w_k \in \mathcal{W} \mid s(X, w_k) > \tau_k\}. \end{aligned} \tag{10}$$

Identification AUC measures the score ranking for target-key attribution in a one-versus-rest setting. Each candidate score uses its own reconstructed mask and reference; the query inversion is shared for a fixed Γ. In this formulation, target-absent negatives may carry other watermarks. A systematic evaluation of false attribution under such negatives is left to future work (Appendix E).

B Detailed Algorithms

Algorithms 1 and 2 describe adaptive watermark injection under blind writing and adaptive detection, respectively. The comparison evaluates each method's complete embedding and detection pipeline. Our detector builds on Wind's sliding-window cosine matching and adds the reference construction in Equation 6; the adaptive injection is specified in Algorithm 1. Local injection calibration and reference construction form a paired embedding–detection design, evaluated together as a complete pipeline. Wind is the immediate methodological baseline, while Tree-Ring and RingID represent earlier designs. Their reported results use their respective detection procedures.

Algorithm 1. Adaptive Watermark Injection

Input: previous image $X^{(i-1)}$, instruction $\pi^{(i)}$, independently sampled key w_i, public configuration Γ, and editing pipeline $\mathcal{E}_\theta$.
Derive $z_i \leftarrow \mathrm{Noise}(w_i; \Gamma)$, $j_i \leftarrow \mathrm{Slot}(w_i; \Gamma)$, and $M^{(i)} \leftarrow M(w_i; \Gamma)$. Random slots allow reuse; single-slot mode fixes j_i.
Construct $P_{\mathrm{base}} \leftarrow P_{\mathrm{base}}(z_i; \Gamma)$ using the deterministic prototype rule.
Obtain editing latent $x_t^{(i)}$ using the editing pipeline; set $x_f^{(i)} \leftarrow \mathcal{F}(x_t^{(i)})$ and $\tilde{x}_t^{(i)} \leftarrow \mathcal{F}^{-1}((1 - M^{(i)}) \odot x_f^{(i)} + M^{(i)} \odot P_{\mathrm{base}})$.
Generate $X_{\mathrm{tmp}}^{(i)} \leftarrow \mathcal{E}_\theta(X^{(i-1)}, \pi^{(i)}, \tilde{x}_t^{(i)})$ and compute $w_{\mathrm{a},f}^{(i)} \leftarrow \mathcal{F}(D_\theta^\dagger(X_{\mathrm{tmp}}^{(i)}; \pi^{(i)}))$.
Reinsert the local pattern: $\hat{x}_t^{(i)} \leftarrow \mathcal{F}^{-1}((1 - M^{(i)}) \odot x_f^{(i)} + M^{(i)} \odot w_{\mathrm{a},f}^{(i)})$.
return $X^{(i)} \leftarrow \mathcal{E}_\theta(X^{(i-1)}, \pi^{(i)}, \hat{x}_t^{(i)})$; persist w_i as the watermark reference and discard the local calibration intermediate.

Algorithm 2. Adaptive Watermark Detection

Input: query image I^*, candidate keys $\{w_k\}_{k=1}^K$, and public configuration Γ (model, reconstruction, scoring, and calibrated thresholds τ_k).
Invert the query once: $\tilde{z}^* \leftarrow D_\theta^\dagger(I^*; \pi_{\mathrm{det}} = \emptyset, \Gamma)$; $x_f^* \leftarrow \mathcal{F}(\tilde{z}^*)$, and $\mathcal{D} \leftarrow \emptyset$.
for each candidate key w_k, reconstruct $z_k \leftarrow \mathrm{Noise}(w_k; \Gamma)$ and $M_k \leftarrow M(w_k; \Gamma)$.
    Construct $p_{f,k} \leftarrow P_{\mathrm{base}}(z_k; \Gamma)$ and the patterned latent $u_k \leftarrow \mathrm{Re}\,\mathcal{F}^{-1}((1 - M_k) \odot \mathcal{F}(z_k) + M_k \odot p_{f,k})$; run the detector's latent inversion directly on u_k: $r_{f,k} \leftarrow \mathcal{F}(R_\theta(u_k; \pi_{\mathrm{ref}} = \emptyset, \Gamma))$ (Appendix B.1).
    Align and score: $S_k \leftarrow \mathrm{Top}_3\text{-Mean}(\mathrm{SlidingCos}(x_f^*[M_k], r_{f,k}[M_k]))$.
    if $S_k > \tau_k$, update $\mathcal{D} \leftarrow \mathcal{D} \cup \{w_k\}$.
end for.
return detected keys $\mathcal{D}$ and scores $\{S_k\}_{k=1}^K$; report "watermarked" iff $\mathcal{D} \neq \emptyset$.

B.1 KEY RECONSTRUCTION AND REFERENCE INVERSION

Seeded noise and base patterns. We follow the seeded noise construction in the official Tree-Ring implementation (Wen et al., 2023): `set_random_seed` initializes the pseudorandom state from the watermark seed, and `get_random_latents` samples Gaussian noise with the pipeline's native latent shape and scaling. In our notation, $\mathrm{Noise}(w;\Gamma)$ resets this generator to the seed specified by w before sampling; the same key and fixed implementation settings reproduce the same tensor. This watermark seed is distinct from the per-image generation seed. Tree-Ring constructs its ring template by applying a centered two-dimensional FFT to the sampled noise and assigning a shared sampled coefficient to each concentric ring in each watermark channel. RingID (Ci et al., 2024) instead specifies ring values through a discrete key combination and applies its pattern-construction operations; its key is not simply another Gaussian draw. We retain the base-pattern rule used at injection when reconstructing P_{base} for detection. Any key-dependent pattern choices must therefore be reproducible from w and the fixed configuration Γ. Writing $z_i = z(w_i;\Gamma)$ and $p_f^{(i)} = P_{\mathrm{base}}(z_i;\Gamma)$, the watermark-bearing reference latent is $u_i = \mathrm{Re}\,\mathcal{F}^{-1}((1-M^{(i)})\odot\mathcal{F}(z_i) + M^{(i)}\odot p_f^{(i)})$. The public configuration Γ fixes the model, sampler, latent shape, base-pattern construction, and slot mapping.

The additional inversion pass. Let $\mathcal{V}_\theta(X)$ denote the deterministic image-to-latent encoding used by the detector. The image and latent interfaces satisfy

$$D_\theta^\dagger(X;\emptyset,\Gamma) = R_\theta(\mathcal{V}_\theta(X);\emptyset,\Gamma). \tag{11}$$

Query inversion initializes this routine with $\mathcal{V}_\theta(X)$; reference inversion initializes the same routine with the reconstructed watermark-bearing latent u_i. The reference call acts directly on the patterned latent without generating or encoding a reference image. Both calls follow the detector's full inversion schedule from its input endpoint toward its noise endpoint, with empty-text conditioning and the 50 steps specified in Section 5.1. Empty-text conditioning uses the text encoder's embedding of the empty string. The reference call treats u_i as the initial state of that schedule, even though it was constructed from noise; it does not start from the noise endpoint and denoise a reference image. For the DDIM notation of Section 2.3, the updates are

$$\begin{aligned} h_{t_0} &= u_i, \qquad e_\ell = \epsilon_\theta(h_{t_\ell}, t_\ell;\emptyset),\\ \widehat{h}_{0,\ell} &= \frac{h_{t_\ell} - \sqrt{1-\bar\alpha_{t_\ell}}\, e_\ell}{\sqrt{\bar\alpha_{t_\ell}}},\\ h_{t_{\ell+1}} &= \sqrt{\bar\alpha_{t_{\ell+1}}}\,\widehat{h}_{0,\ell} + \sqrt{1-\bar\alpha_{t_{\ell+1}}}\, e_\ell, \qquad \ell = 0,\dots,L-1, \end{aligned} \tag{12}$$

where $t_0,\dots,t_L$ are the query detector's inversion endpoints and $L = 50$; scheduler-specific scaling and endpoint conventions are shared between the two calls. The output is $R_\theta(u_i;\emptyset,\Gamma) = h_{t_L}$, followed by a centered FFT to obtain $r_f(w_i;\Gamma)$. For each editing model, reference construction reuses that model's query-inversion implementation and settings. This is an additional transformation of the candidate reference relative to native Tree-Ring/RingID matching; it is not a second inversion of the query image. The image-dependent reinsertion tensor $w_{\mathrm{a},f}^{(i)}$ is neither an input to this procedure nor assumed equal to its output.

Scoring and threshold calibration. Sliding-window cosine matching is applied without geometric registration, and the score averages the three largest window similarities. For threshold calibration, we use 1,000 unwatermarked editing outputs generated from source images randomly sampled from ImgEdit separately from the evaluation set. Each sample is an individual editing-step output rather than an entire trajectory. The negatives contain no embedded watermark; images carrying other keys are excluded from the calibration pool. Calibration and evaluation share no source images or editing trajectories, and no evaluation outputs are used for calibration. The calibrated thresholds remain fixed during evaluation. We select each threshold to target a nominal FPR of 1% on the independent set of 1,000 calibration negatives. Using the same fixed threshold, we compute TPR on test positives and the empirical FPR on test negatives under the corresponding evaluation condition, without recalibrating on the test set. The test-set FPR may therefore differ from 1%; in the reported TPR@1%FPR notation, 1% denotes the nominal calibration FPR rather than a guaranteed

test-set FPR. A shared threshold is the special case $\tau_i = \tau$ for all keys. Detection uses the query image, candidate keys, fixed public configuration Γ, and calibrated thresholds. The image-dependent injection-calibration tensor $w_{\mathrm{a},f}^{(i)}$ is not retained for detection.

C Additional Analysis of Tree-Ring Watermark Coexistence

Our method is inspired by training-free Tree-Ring watermarks, which provide a frequency-domain representation suitable for multiple watermark records. This section gives theoretical intuition for why prompt-free inversion can retain information about a watermark injected during editing. It does not guarantee local injection calibration, alignment with a key-derived reference, or multi-watermark coexistence.

C.1 Theoretical Intuition for Prompt-Free Detection

A Fixed-Point View of Classifier-Free Guidance (CFG). Wang et al. (Wang et al., 2026) study a fixed-point interpretation of CFG in which conditional and unconditional denoising trajectories become closely aligned. Their analysis motivates examining mismatch between conditional editing and prompt-free query inversion. It does not establish that reference inversion of key-reconstructed noise matches the recovered query signal. This interpretation is explanatory only: it does not assume that the two trajectories are identical for the models or settings evaluated in this paper. They define the *golden path* as latent states where conditional and unconditional denoising maps align. Formally, let $f^u_{t\to 0}$ and $f^c_{t\to 0}$ denote the unconditional and conditional denoising mappings from timestep t to timestep 0, respectively, and let $\mathcal{M}_t$ denote the probability manifold at timestep t. The golden path is defined as

$$\mathcal{G}_t := \{x_t \in \mathcal{M}_t \mid f^u_{t\to 0}(x_t) = f^c_{t\to 0}(x_t)\}\,. \tag{13}$$

The theorem in Wang et al. (Wang et al., 2026) bounds a conditional–unconditional gap under that paper's fixed-point assumptions and its iteration and subproblem parameters. Those assumptions and parameters are not established for our local calibration or for the reference-noise operator R_θ. We therefore do not transfer that bound to our method and do not claim a theoretical guarantee of prompt-free verification or multi-watermark coexistence.

As an empirical check, we apply Tree-Ring watermarking to 1,000 samples and recover the initial noise using both conditional and unconditional branches. The similarities describe condition mismatch in that experiment and do not prove condition-agnostic detection.

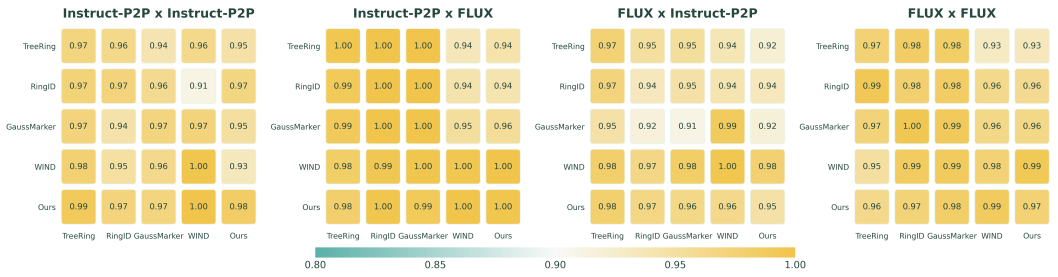

Figure 7: Two-round editing watermark coexistence AUC.

C.2 Tree-Ring Watermark Coexistence is Agnostic to Editing Models and Watermarking Methods

The compatibility of watermarks embedded by different models and methods is critical for multi-actor provenance chains. We aim to examine the advantages of Tree-Ring watermarks over prior training-based approaches. To demonstrate the flexibility and generality of Tree-Ring watermark coexistence, we evaluate combinations of two editing models (FLUX.2-Klein-4B and InstructPix2Pix) together with five Tree-Ring watermarking methods. Figure 7 reports the results under a two-round editing setting, where each entry denotes the AUC of all embedded watermarks after sequential edits. All results are computed over 100 samples. Across all combinations of editing models and watermarking methods, the coexistence AUC consistently exceeds 90%, demonstrating that Tree-Ring watermarks remain reliably detectable even after iterative edits. These results extend the study of watermark coexistence to training-free Tree-Ring-style methods; they do not constitute a matched comparison with prior training-based approaches (Petrov et al., 2026).

The evaluated Tree-Ring-style methods also exhibit intra-method compatibility in this two-round setting. This observation is intuitive, as Tree-Ring watermarks naturally provide larger embedding capacity by injecting watermark signals into a dedicated channel, thereby reducing interference among multiple watermarks.

Table 4: Watermark coexistence results of FireRed-Image-Edit under the blind writing setting. Each entry is reported as AUC / TPR@1%FPR / ALL detection rate. The best result within the same edit round is highlighted in yellow and bold.

Framework	Edit Rounds	Methods	Clean	JPEG	Blur	Noise	Brightness	PSNR	SSIM
Deterministic Slot Assignment	**2**	*Tree-Ring*	0.965 / 0.855 / 0.710	0.964 / 0.780 / 0.600	0.698 / 0.125 / 0.030	0.716 / 0.225 / 0.070	0.903 / 0.745 / 0.600	30.201	0.915
		GaussMarker	0.982 / 0.960 / 0.920	0.981 / **0.945** / **0.890**	0.748 / **0.360** / 0.130	0.778 / 0.315 / 0.160	0.963 / 0.860 / 0.780	24.499	0.841
		Wind	0.965 / 0.875 / 0.760	0.942 / 0.745 / 0.510	0.712 / 0.160 / 0.011	0.724 / 0.170 / 0.039	0.901 / 0.630 / 0.430	26.826	0.886
		Ours	**0.992** / **0.970** / **0.940**	**0.984** / **0.945** / **0.890**	**0.774** / 0.315 / **0.150**	**0.907** / **0.580** / **0.380**	**0.966** / **0.875** / **0.800**	**31.204**	**0.953**
	3	*Tree-Ring*	0.924 / 0.727 / 0.450	0.887 / 0.617 / 0.280	0.660 / 0.147 / 0.010	0.656 / 0.150 / 0.010	0.832 / 0.480 / 0.180	27.058	0.888
		GaussMarker	0.944 / 0.863 / 0.740	0.936 / 0.813 / 0.620	0.713 / **0.247** / **0.030**	0.687 / 0.203 / 0.050	0.905 / **0.767** / **0.570**	21.565	0.801
		Wind	0.889 / 0.657 / 0.350	0.883 / 0.597 / 0.290	0.650 / 0.077 / 0.000	0.654 / 0.160 / 0.011	0.849 / 0.450 / 0.140	23.979	0.854
		Ours	**0.963** / **0.887** / **0.760**	**0.954** / **0.833** / **0.670**	**0.740** / 0.200 / 0.020	**0.861** / **0.463** / **0.140**	**0.924** / 0.763 / 0.550	**27.130**	**0.907**
	4	*Tree-Ring*	0.869 / 0.520 / 0.080	0.806 / 0.453 / 0.050	0.634 / 0.100 / 0.000	0.585 / 0.070 / 0.000	0.832 / 0.383 / 0.020	**25.407**	0.860
		GaussMarker	0.925 / 0.767 / 0.480	0.894 / 0.728 / 0.450	0.653 / **0.225** / **0.010**	0.613 / 0.165 / 0.020	0.897 / 0.690 / 0.350	20.193	0.768
		Wind	0.877 / 0.630 / 0.250	0.855 / 0.520 / 0.140	0.620 / 0.077 / 0.000	0.559 / 0.087 / 0.000	0.839 / 0.502 / 0.130	22.336	0.820
		Ours	**0.951** / **0.833** / **0.590**	**0.948** / **0.775** / **0.460**	**0.716** / 0.147 / 0.000	**0.840** / **0.472** / **0.120**	**0.935** / **0.770** / **0.490**	24.446	**0.866**
	5	*Tree-Ring*	0.826 / 0.482 / 0.070	0.784 / 0.372 / 0.010	0.626 / 0.104 / **0.000**	0.603 / 0.124 / 0.000	0.775 / 0.472 / 0.030	**24.288**	**0.857**
		GaussMarker	0.906 / 0.748 / 0.410	0.864 / 0.626 / 0.240	0.634 / **0.186** / **0.000**	0.641 / 0.110 / 0.000	0.861 / 0.580 / 0.190	19.303	0.764
		Wind	0.819 / 0.332 / 0.000	0.800 / 0.336 / 0.000	0.629 / 0.154 / **0.000**	0.580 / 0.044 / 0.000	0.779 / 0.408 / 0.070	21.192	0.813
		Ours	**0.938** / **0.794** / **0.500**	**0.933** / **0.676** / **0.320**	**0.674** / 0.144 / **0.000**	**0.850** / **0.422** / **0.070**	**0.900** / **0.660** / **0.290**	22.829	0.836

D Additional Experimental Results

D.1 Blind Writing Results under Single-Slot Multiplexing

Table 5 reports the complete blind-writing results for the single-slot multiplexing framework with Flux. All watermarks share the same latent channel and frequency slot, directly testing their ability to remain detectable under repeated co-location and sequential editing. This protocol forces a slot collision at every subsequent round and tests whether coexistence requires separate embedding locations.

Table 5: Watermark coexistence results under the blind writing setting using the Single-Slot Multiplexing framework with Flux. Each entry is reported as AUC / TPR@1%FPR / ALL Detection Rate. The best result is highlighted in yellow and bold.

Edit Rounds	Methods	Clean	JPEG	Blur	Noise	Brightness	PSNR	SSIM
2	*Tree-Ring*	0.865 / 0.548 / 0.332	0.807 / 0.470 / 0.261	0.534 / 0.077 / 0.011	0.672 / 0.182 / 0.061	0.766 / 0.259 / 0.093	**32.982**	**0.963**
	RingID	0.957 / 0.805 / 0.646	0.888 / 0.645 / 0.443	0.497 / 0.059 / 0.004	0.726 / 0.225 / 0.079	0.882 / 0.539 / 0.318	31.797	0.954
	GaussMarker	0.966 / 0.848 / 0.714	0.917 / 0.668 / 0.461	0.595 / 0.025 / 0.004	0.703 / 0.184 / 0.050	0.875 / 0.566 / 0.332	28.576	0.924
	Wind	0.931 / 0.611 / 0.364	0.862 / 0.480 / 0.264	0.536 / 0.023 / 0.000	0.669 / 0.150 / 0.029	0.834 / 0.457 / 0.236	31.496	0.954
	Ours	**0.996** / **0.986** / **0.971**	**0.993** / **0.952** / **0.904**	**0.683** / **0.148** / **0.064**	**0.867** / **0.484** / **0.357**	**0.970** / **0.889** / **0.836**	31.417	0.953
3	*Tree-Ring*	0.788 / 0.383 / 0.075	0.758 / 0.406 / 0.114	0.480 / 0.075 / 0.000	0.623 / 0.102 / 0.007	0.729 / 0.287 / 0.046	**29.322**	**0.935**
	RingID	0.914 / 0.651 / 0.296	0.868 / 0.524 / 0.189	0.566 / 0.105 / 0.000	0.733 / 0.164 / 0.021	0.801 / 0.542 / 0.193	28.210	0.923
	GaussMarker	0.890 / 0.620 / 0.246	0.824 / 0.419 / 0.100	0.441 / 0.018 / 0.000	0.674 / 0.136 / 0.007	0.785 / 0.370 / 0.089	25.166	0.879
	Wind	0.891 / 0.599 / 0.246	0.830 / 0.436 / 0.111	0.495 / 0.039 / 0.000	0.594 / 0.054 / 0.000	0.784 / 0.293 / 0.050	27.865	0.917
	Ours	**0.990** / **0.951** / **0.875**	**0.987** / **0.943** / **0.850**	**0.691** / **0.111** / **0.014**	**0.876** / **0.389** / **0.207**	**0.956** / **0.846** / **0.711**	27.729	0.916
4	*Tree-Ring*	0.752 / 0.479 / 0.064	0.716 / 0.281 / 0.018	0.540 / 0.050 / **0.000**	0.584 / 0.088 / 0.000	0.676 / 0.273 / 0.025	**26.570**	**0.898**
	RingID	0.856 / 0.504 / 0.054	0.808 / 0.307 / 0.007	0.563 / **0.069** / **0.000**	0.676 / 0.233 / 0.014	0.792 / 0.406 / 0.032	25.622	0.886
	GaussMarker	0.864 / 0.546 / 0.086	0.808 / 0.433 / 0.043	0.506 / 0.037 / **0.000**	0.645 / 0.163 / 0.004	0.806 / 0.403 / 0.054	22.651	0.832
	Wind	0.795 / 0.296 / 0.007	0.759 / 0.281 / 0.011	0.450 / 0.014 / **0.000**	0.649 / 0.057 / 0.000	0.743 / 0.237 / 0.007	25.118	0.881
	Ours	**0.984** / **0.947** / **0.832**	**0.980** / **0.915** / **0.736**	**0.703** / 0.034 / **0.000**	**0.899** / **0.664** / **0.418**	**0.963** / **0.862** / **0.700**	24.889	0.875
5	*Tree-Ring*	0.714 / 0.441 / 0.021	0.699 / 0.351 / 0.007	0.536 / **0.124** / **0.000**	0.642 / 0.116 / 0.000	0.625 / 0.198 / 0.004	**24.470**	**0.864**
	RingID	0.806 / 0.402 / 0.007	0.806 / 0.308 / 0.007	0.541 / 0.074 / **0.000**	0.653 / 0.159 / 0.000	0.779 / 0.296 / 0.000	23.740	0.850
	GaussMarker	0.789 / 0.354 / 0.004	0.776 / 0.373 / 0.011	0.527 / 0.055 / **0.000**	0.637 / 0.099 / 0.000	0.748 / 0.314 / 0.004	21.146	0.796
	Wind	0.776 / 0.314 / 0.000	0.756 / 0.219 / 0.000	0.521 / 0.011 / **0.000**	0.624 / 0.119 / 0.000	0.723 / 0.139 / 0.000	23.387	0.851
	Ours	**0.978** / **0.907** / **0.700**	**0.968** / **0.874** / **0.632**	**0.642** / 0.044 / **0.000**	**0.878** / **0.455** / **0.186**	**0.955** / **0.816** / **0.554**	23.046	0.843

D.2 Informed Writing Results

Table 6 reports the controlled informed-writing verification results, where each embedding operation is assigned with knowledge of occupied channels to alleviate frequency-slot collisions. In this

controlled protocol, a new entity is given only the occupied watermark-bearing channel, not the previous entity's key, exact frequency pattern, phase, amplitude, template, or detector record. Thus, the setting models limited coordination for collision avoidance rather than disclosure of a reproducible watermark. In the clean setting, our method achieves AUC values of 1.000 across rounds 2–5. At round 2, its TPR@1%FPR and All-Detection Rate are 0.998 and 0.996, respectively; both reach 1.000 in rounds 3–5. After five rounds, its All-Detection Rate is 0.996 under JPEG compression, 0.668 under additive noise, and 0.954 under brightness adjustment. Blur remains challenging: despite an AUC of 0.800, its round-5 TPR@1%FPR is 0.324 and All-Detection Rate is 0.004. These results support watermark coexistence under the controlled informed-writing protocol, with robustness depending on the applied perturbation.

Table 6: Watermark coexistence results under the Informed Writing setting with Flux. Each entry is reported as AUC / TPR@1%FPR / All-Detection Rate. For each editing round and condition, the highest value of each metric is highlighted in yellow and bold; ties are highlighted equally.

Edit Rounds	Methods	Clean	JPEG	Blur	Noise	Brightness
2	*Tree-Ring*	0.986 / 0.857 / 0.721	0.963 / 0.693 / 0.479	0.603 / 0.079 / 0.007	0.690 / 0.079 / 0.000	0.929 / 0.632 / 0.386
	RingID	0.990 / 0.891 / 0.786	0.952 / 0.671 / 0.443	0.593 / 0.123 / 0.014	0.677 / 0.164 / 0.025	0.916 / 0.541 / 0.264
	GaussMarker	0.997 / 0.975 / 0.950	0.977 / 0.866 / 0.782	0.635 / 0.113 / 0.014	0.806 / 0.498 / 0.357	0.951 / 0.768 / 0.661
	Wind	0.935 / 0.721 / 0.500	0.894 / 0.621 / 0.382	0.524 / 0.025 / 0.000	0.745 / 0.256 / 0.111	0.810 / 0.405 / 0.186
	Ours	**1.000 / 0.998 / 0.996**	**0.999 / 0.995 / 0.989**	**0.800 / 0.373 / 0.150**	**0.953 / 0.766 / 0.646**	**0.990 / 0.964 / 0.950**
3	*Tree-Ring*	0.981 / 0.842 / 0.568	0.953 / 0.826 / 0.571	0.583 / 0.152 / 0.007	0.724 / 0.138 / 0.004	0.922 / 0.642 / 0.261
	RingID	0.987 / 0.943 / 0.829	0.960 / 0.769 / 0.454	0.600 / 0.080 / 0.000	0.679 / 0.088 / 0.000	0.922 / 0.637 / 0.257
	GaussMarker	0.995 / 0.956 / 0.875	0.984 / 0.924 / 0.807	0.680 / 0.257 / 0.011	0.812 / 0.440 / 0.211	0.960 / 0.849 / 0.696
	Wind	0.899 / 0.614 / 0.254	0.865 / 0.415 / 0.125	0.662 / 0.075 / 0.000	0.748 / 0.144 / 0.011	0.826 / 0.444 / 0.150
	Ours	**1.000 / 1.000 / 1.000**	**1.000 / 0.996 / 0.989**	**0.809 / 0.390 / 0.071**	**0.946 / 0.802 / 0.664**	**0.991 / 0.973 / 0.950**
4	*Tree-Ring*	0.986 / 0.892 / 0.618	0.967 / 0.829 / 0.464	0.588 / 0.124 / 0.000	0.652 / 0.121 / 0.000	0.956 / 0.709 / 0.257
	RingID	0.983 / 0.869 / 0.550	0.963 / 0.836 / 0.486	0.545 / 0.097 / 0.000	0.694 / 0.150 / 0.000	0.956 / 0.718 / 0.275
	GaussMarker	0.998 / 0.979 / 0.921	0.987 / 0.936 / 0.814	0.610 / 0.146 / 0.000	0.873 / 0.572 / 0.304	0.979 / 0.903 / 0.739
	Wind	0.812 / 0.433 / 0.029	0.832 / 0.478 / 0.068	0.556 / 0.033 / 0.000	0.712 / 0.150 / 0.007	0.752 / 0.274 / 0.007
	Ours	**1.000 / 1.000 / 1.000**	**1.000 / 0.999 / 0.996**	**0.770 / 0.370 / 0.021**	**0.967 / 0.863 / 0.689**	**0.997 / 0.991 / 0.968**
5	*Tree-Ring*	0.978 / 0.859 / 0.404	0.953 / 0.714 / 0.171	0.554 / 0.045 / 0.000	0.670 / 0.039 / 0.000	0.951 / 0.734 / 0.229
	RingID	0.984 / 0.913 / 0.625	0.955 / 0.834 / 0.404	0.523 / 0.104 / 0.000	0.642 / 0.133 / 0.004	0.938 / 0.664 / 0.132
	GaussMarker	0.997 / 0.979 / 0.907	0.987 / 0.941 / 0.786	0.615 / 0.150 / 0.000	0.906 / 0.629 / 0.318	0.979 / 0.899 / 0.711
	Wind	0.830 / 0.446 / 0.018	0.798 / 0.322 / 0.004	0.562 / 0.062 / 0.000	0.681 / 0.141 / 0.000	0.768 / 0.252 / 0.007
	Ours	**1.000 / 1.000 / 1.000**	**1.000 / 0.999 / 0.996**	**0.800 / 0.324 / 0.004**	**0.959 / 0.840 / 0.668**	**0.993 / 0.983 / 0.954**

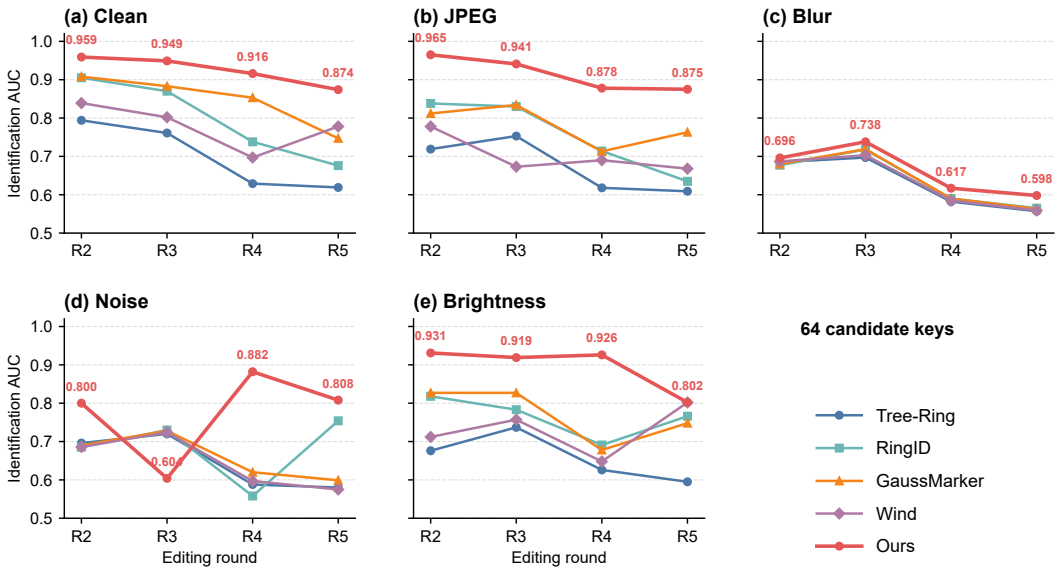

Figure 8: Watermark identification AUC across attacks and editing rounds. Each target key is selected from 64 candidates using the highest watermark score.

Table 7: Watermark identification AUC under deterministic slot assignment (key size = 64). The best result in each column within the same edit round is highlighted in bold.

Framework	Edit Rounds	Methods	Ring Width	Clean	JPEG	Blur	Noise	Brightness
Deterministic Slot Assignment	**2**	*Tree-Ring*	16	0.794	0.719	0.685	0.696	0.676
		RingID	16	0.905	0.838	0.677	0.685	0.818
		GaussMarker	16	0.908	0.812	0.680	0.690	0.827
		Wind	16	0.839	0.778	0.687	0.686	0.712
		Ours	16	**0.959**	**0.965**	**0.696**	**0.800**	**0.931**
	3	*Tree-Ring*	16	0.761	0.753	0.697	0.720	0.737
		RingID	16	0.870	0.830	0.718	**0.730**	0.783
		GaussMarker	16	0.883	0.834	0.719	0.728	0.827
		Wind	16	0.802	0.673	0.703	0.724	0.757
		Ours	16	**0.949**	**0.941**	**0.738**	0.604	**0.919**
	4	*Tree-Ring*	16	0.629	0.618	0.582	0.588	0.626
		RingID	16	0.738	0.714	0.590	0.558	0.691
		GaussMarker	16	0.853	0.713	0.590	0.620	0.678
		Wind	16	0.697	0.690	0.585	0.597	0.648
		Ours	16	**0.916**	**0.878**	**0.617**	**0.882**	**0.926**
	5	*Tree-Ring*	16	0.619	0.609	0.557	0.580	0.595
		RingID	16	0.676	0.635	0.565	0.754	0.766
		GaussMarker	16	0.747	0.763	0.563	0.599	0.748
		Wind	16	0.778	0.668	0.559	0.575	**0.802**
		Ours	16	**0.874**	**0.875**	**0.598**	**0.808**	**0.802**

D.3 Watermark Identification under Deterministic Slot Assignment

Table 7 reports watermark identification performance under deterministic slot assignment with a candidate key set size of 64. We evaluate Tree-Ring, RingID, GaussMarker, Wind, and our adaptive watermark across rounds 2–5 under clean conditions and four perturbation settings.

Figure 8 and Table 7 show that our method retains the strongest identification performance in most conditions. After five rounds, it reaches AUCs of 0.874 on clean images and 0.875 under JPEG compression. The strongest baselines reach 0.778 and 0.763, respectively. Under additive noise, our method obtains 0.808, compared with the best baseline value of 0.754.

Existing Tree-Ring-style methods degrade more sharply with repeated editing, especially under compression.

D.4 Blind Writing Results on FireRed-Image-Edit

Table 4 presents watermark coexistence results under the blind writing setting using FireRed-Image-Edit. In this setting, each editing round is performed without access to previously embedded watermark information, making watermark coexistence substantially more challenging than informed writing.

Despite the absence of watermark-side information, the proposed adaptive watermark consistently achieves the strongest overall performance. In particular, our method maintains superior robustness under JPEG compression, additive noise, and brightness perturbations while preserving competitive image quality in terms of PSNR and SSIM.

As the number of editing rounds increases, the performance gap between our method and existing baselines becomes increasingly pronounced. This observation suggests that trajectory alignment effectively mitigates the cumulative attenuation introduced by repeated editing and inversion, allowing multiple watermarks to coexist more reliably throughout long editing sequences.

D.5 Discussion on Watermark Coexistence

Deterministic slot assignment partitions the frequency-domain watermark space into distinct slots and maps independently sampled keys to this full set at each round. The slots define possible embedding locations; their allocation does not guarantee that different watermarks occupy different locations. Blind writing therefore requires no record of previous slot occupancy and permits collisions.

Unlike previous approaches that rely on heuristic channel allocation or architecture-specific modifications, the proposed framework is model agnostic and naturally adapts to different latent resolutions and channel configurations. By decoupling watermark placement from the underlying diffusion architecture, the framework can be integrated into a wide range of Tree-Ring-style watermarking methods.

The single-slot results directly evaluate repeated collisions by placing every watermark in the same frequency region. On clean images after five rounds, the adaptive watermark retains an AUC of 0.978 and an All-Detection Rate of 0.700 under this protocol. These observations support coexistence under shared-slot placement in the tested setting, without relying on collision avoidance.

D.6 Qualitative Visualization

To further validate the practical utility of multi-watermark coexistence under sequential editing, we visualize the complete iterative editing pipeline in Figure 9. Across five editing rounds, our method preserves visual quality, semantic consistency, and structural integrity without introducing noticeable artifacts.



Figure 9: Visualization of multi-turn watermarking. Watermarked images generated by our method over five editing rounds are compared with watermark-free edited images.

E LIMITATIONS

False attribution among coexisting watermarks. Our evaluation focuses on retaining detection of embedded keys. All-Detection Rate does not penalize detections of absent keys: an image containing keys A and B can count as a success even if the detector also reports an absent key C. Moreover, thresholds calibrated on fully unwatermarked images do not establish false-positive control for images carrying other keys. Systematically evaluating this setting requires additional controls across key combinations, slot collisions, and editing histories. We leave this evaluation, including false-attribution rates and exact recovery of the embedded key set, to future work.

Quality evaluation. We mainly report PSNR and SSIM as image-quality metrics. However, these metrics do not always align well with human perceptual judgment, and future work could incorporate additional perceptual metrics. Nevertheless, as shown in Figure 9, the visual results after multiple rounds of editing are similar across methods, suggesting that sequential watermarking does not introduce noticeable quality degradation.

Limited direct comparison with training-based methods. Prior work establishes coexistence among training-based watermarks, while the long-term preservation of training-free watermarks through sequential generative image editing remains insufficiently explored. Our experiments address this latter setting. We do not provide a direct, matched comparison with training-based methods because of limited computational resources. Our primary target is training-free watermarking, which avoids per-method training. Accordingly, comparisons with training-based approaches are contextual rather than conclusive; a controlled evaluation under identical models, data, and attack budgets remains an important direction for future work.

F SEQEDIT-BM DATASET CONSTRUCTION

To construct SEQEDIT-BM, we employ Gemini-2.5 (DeepMind, 2025) to automatically extend a single-turn editing instruction into a multi-turn editing trajectory. Given an input image and a predefined edit type, the prompt requires the model to generate a sequence of coherent editing instructions that can be executed consecutively.

Because the trajectories are generated with an LLM, we treat automatic generation as a proposal stage rather than as evidence of dataset validity. We apply the constraints described below before retaining a trajectory: each instruction must be actionable, consistent with the assigned edit category, applicable to the preceding output, and free of contradictory or unrelated operations. Trajectories that fail these checks are regenerated or removed. These checks do not establish that the benchmark fully represents real user workflows.

The prompt enforces several constraints to improve dataset quality. First, each editing instruction must operate on the output of the previous editing round rather than the original image, ensuring a realistic sequential editing process. Second, all turns are restricted to the same edit category, preventing semantic drift across editing rounds. Third, the generated trajectory must preserve scene continuity and avoid contradictory modifications, allowing watermark persistence to be evaluated under long editing chains.

The prompt and filtering procedure also impose compliance constraints. Instructions must describe ordinary image-editing operations, must not request illegal, harmful, deceptive, or privacy-invasive content, and must not include executable code, external tool calls, or hidden instructions. The retained dataset contains only the source image identifier, edit category, and filtered editing instructions; model-generated metadata and rejected candidates are not used in evaluation. These checks improve procedural compliance, but they do not establish that the benchmark fully represents real user workflows.

These constraints are particularly important for our watermark coexistence study. Since multiple watermarks are injected across different editing rounds, the benchmark must provide progressive yet consistent modifications while maintaining sufficient visual diversity. The resulting trajectories provide a controlled testbed for evaluating watermark attenuation, watermark coexistence, and watermark identification under continual image editing. We have provided the detailed prompt in Figure 10.

Prompt Used for SEQEDIT-BM Trajectory Generation

```
You are an expert dataset curator for multi-turn image editing benchmarks.

You are given:
1. An input image.
2. A required edit type for every turn.

Assigned edit type for all turns:
"{primary_edit_type}"

Required edit type schedule:
{turn_type_lines}

Your task:
Generate exactly {turns} sequential editing instructions for a
continuous multi-turn image editing pipeline.

Hard requirements:
- Output ONLY one JSON object.
- The JSON object must contain exactly these keys:
  "turn1" ... "turnN".
- turn1 must be a natural first edit based on the input image.
- turn2 to turnN must edit the result of the previous turn.
- All turns must follow the assigned edit type.
- Do not undo previous edits.
- Do not create contradictions.
- Keep instructions concise and realistic.
- Maintain subject identity and scene continuity.
- Avoid impossible edits and unrelated scene changes.

Image understanding requirements:
- Inspect the image content before generating instructions.
- Choose realistic editable targets.
- Build a coherent editing trajectory.

Quality target:
- Visually specific
- Progressive
- Natural
- High quality

Return JSON only.
```

Figure 10: Prompt used to generate multi-turn editing trajectories in SEQEDIT-BM. Given an input image and an edit type, the model produces a coherent sequence of editing instructions that preserve scene continuity while progressively modifying the image.

G Additional Related Work

G.1 Training-based Watermarks

HiDDeN (Zhu et al., 2018). HiDDeN is a representative end-to-end watermarking method. It embeds a secret message into an image through a learned encoder while keeping the visual changes imperceptible. A corresponding decoder is used to recover the message from the watermarked image. The embedded watermark remains detectable after common image transformations such as noise corruption, cropping, and JPEG compression.

RivaGAN (Zhang et al., 2019). RivaGAN extends neural watermarking with attention mechanisms and adversarial training. The attention module helps locate appropriate embedding regions, while adversarial learning improves the robustness and visual quality of the embedded watermark.

RoSteALS (Bui et al., 2023). RoSteALS performs watermark embedding in the latent space of a pretrained autoencoder rather than directly in the image space. Leveraging latent representations reduces learning complexity and enhances robustness against image transformations.

SSL (Fernandez et al., 2022). SSL introduces a feature-space watermarking framework based on self-supervised representations. It embeds watermarks using carrier vectors in the learned representation space rather than directly manipulating image pixels, enhancing robustness.

G.2 Tree-Ring Inspired Watermarks

Tree-Ring (Wen et al., 2023). The Tree-Ring approach embeds the watermark directly into the initial noise latent at the beginning of the generative process. By leveraging the invertibility of diffusion models and the inherent stability of the frequency domain, it enables robust watermark detection through frequency domain analysis after reconstructing the initial noise.

RingID (Ci et al., 2024). RingID was later proposed, attributing the detectability of the Tree-Ring watermark to the distribution shift caused by discarding the imaginary part during the inverse Fast Fourier Transform (IFFT) process. It found that this distribution shift can help in the verification task but fails in the identification task. It then proposed Spatial Shift, Lossless Imprinting, and Making A Rounder Ring to enhance the watermark identification rate.

GaussMarker (Li et al., 2025). GaussMarker proposes a robust dual-domain watermarking framework for diffusion models. It employs a pipelined injector to embed watermarks consistently across both the spatial and frequency domains. Furthermore, it introduces a model-agnostic Gaussian Noise Restorer to refine the extracted noise from tampered images, which significantly enhances detection robustness.

Wind (Arabi et al., 2024). WIND proposes a two-stage watermarking framework. During generation, it embeds watermark information by augmenting the initial noise with Fourier-domain patterns that encode the corresponding noise group. For watermark detection, Wind first retrieves the relevant noise group and subsequently searches within this group for an initial noise pattern that best matches the target image. Tree-Ring relies on the p-value or ℓ_2 distance for detection. In contrast, WIND adopts sliding-window cosine similarity for watermark matching and improves detection robustness.

G.3 Security, Ownership Protection, and Generated-Content Detection

Model ownership protection. SSLGuard (Cong et al., 2022) watermarks self-supervised encoders and uses shadow training to improve robustness to model stealing. Cong et al. (Cong et al., 2024) examine model merging, finding the evaluated quantization watermark less robust than instructional fingerprinting. Both concern model ownership; we study image-level provenance through sequential edits.

Generative-model misuse and content attribution. Unsafe Diffusion (Qu et al., 2023) studies unsafe text-to-image generation and image editing for hateful meme creation. MGTBench (He et al., 2024) benchmarks machine-generated text detection and source attribution under adversarial modifications, including paraphrasing. These studies contextualize the need for robust provenance, while addressing different tasks from preserving deliberately embedded image-watermark keys.