Spherical Watermark: Encryption-Free, Lossless Watermarking for Diffusion Models

Xiaoxiao Hu[1], **Jiaqi Jin**[1], **Sheng Li**[1], **Wanli Peng**[2], **Xinpeng Zhang**[1], **Zhenxing Qian**[1,*]

[1]Fudan University, [2]China Agricultural University

Presenter: Yuhan Cao

ICLR 2026 · Submission Number 14944

1. Motivation: why do we need lossless watermarking?

Why AIGC watermarking matters

- Diffusion models raise **provenance and authenticity** concerns.
- Generated content needs **user-level traceability.**

Limitations of existing methods

- Lossy schemes introduce detectable distribution shift and fidelity drop, and can therefore be adversarially detected or removed.
- Existing lossless schemes still rely on per-image keys (Gaussian Shading, CVPR2024) or heavy decoding overhead (PRC Watermark, ICLR2025).

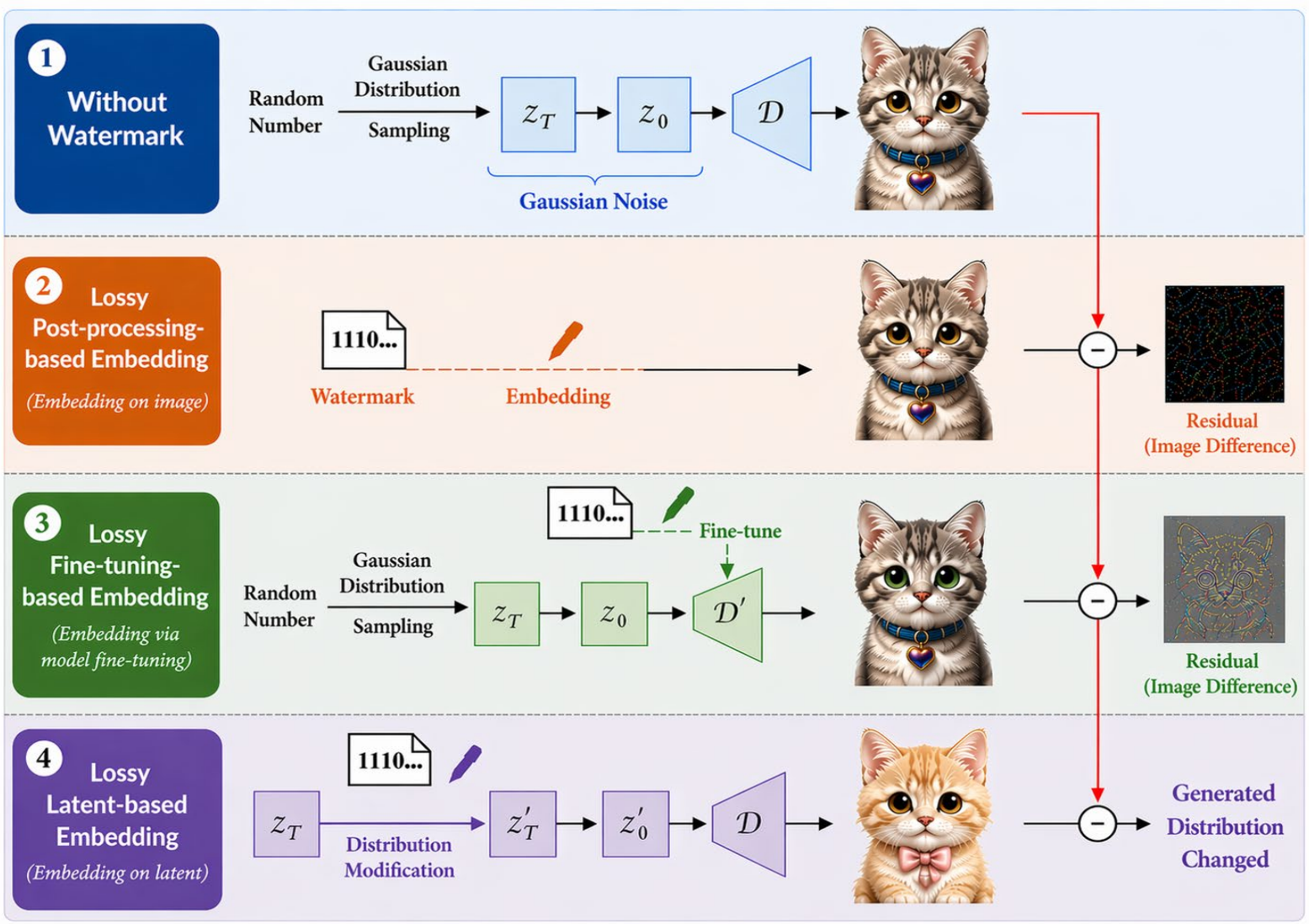
The desired goal

Lossless **Traceable** **Efficient**

Fig.1. Example watermarked images generated by Spherical Watermark on SD3 and FLUX.1-dev.

1. Motivation: existing AIGC watermarking methods

Existing lossy watermarking methods leave detectable traces or alter the generation process.



2. Problem Formulation.

We want to design two procedures in the latent space for watermark embedding and extraction,

$$\text{Embed: } \boldsymbol{m} \in \{0,1\}^{l_m} \to \boldsymbol{z}_w \in \mathbb{R}^{l_x}, \qquad \text{Extract: } \hat{\boldsymbol{z}}_w \in \mathbb{R}^{l_x} \to \hat{\boldsymbol{m}} \in \{0,1\}^{l_m}.$$

- **Undetectability / Losslessness**: Any efficient adversary should fail to distinguish watermarked outputs from non-watermarked ones.

$$\left|\Pr[A'(\mathcal{G}(\boldsymbol{z}_w)) = 1] - \Pr[A'(\mathcal{G}(\boldsymbol{z})) = 1]\right| \leq \text{negl}(\rho).$$

- **Traceability / Exact Extraction**: The embedded watermark should be recovered from the generated image.

$$\Pr[\text{Extract}(\mathcal{G}^{-1}(\boldsymbol{O}_w)) = \boldsymbol{m}] \geq 1 - \text{negl}(\rho).$$

The key is to achieve both losslessness and traceability **while preserving the Gaussian prior**.

Compared with prior watermarking work

Per-image key	**No**
Weight changes	**No**
Theory goal	**Preserve Gaussian prior**
Empirical gain	**High fidelity + robustness**
Engineering gain	**Faster Embedding and extraction**

$\boldsymbol{m}$: secret message (watermark bits) $\boldsymbol{z}_w$ / $\boldsymbol{O}_w$: watermarked latent / image A': polynomial-time adversary

$\mathcal{G}$: diffusion generation, transform latent noise into image $\mathcal{G}^{-1}$: diffusion inversion, transform image into latent

3. Main challenges.

As illustrated in Fig. 2, existing lossless mapping-based watermarking methods also construct mappings in the latent space, but they rely on per-image keys and incur additional cryptographic encoding and decoding overhead. Eliminating these costs without sacrificing losslessness gives rise to **Challenges C1–C3**.

C1 · Preserve randomness

Removing per-image keys or encryption could not disrupt the Gaussian randomness required for lossless watermarking.

C2 · Enable lightweight integration

Seamlessly integrate the watermarking mechanism into the diffusion pipeline without introducing heavy overhead, i.e., iterative Belief-Propagation decoding.

C3 · Ensure reliable tracing

Watermarks should remain recoverable after common distortions on watermarked images. Common distortions include adversarial attacks and post-processing operations.

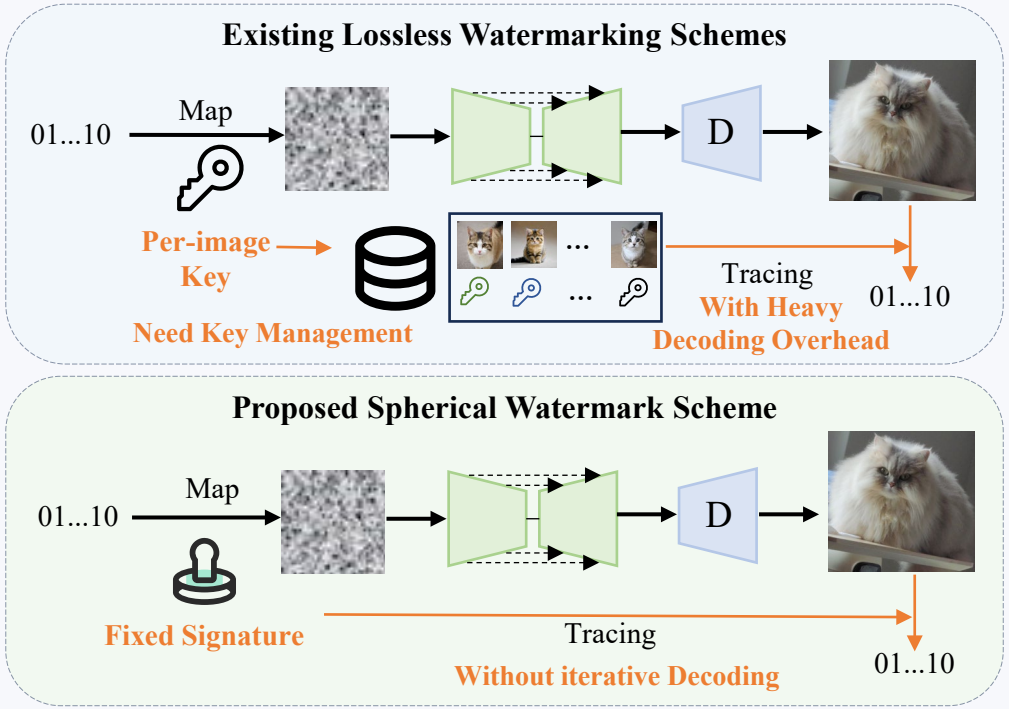
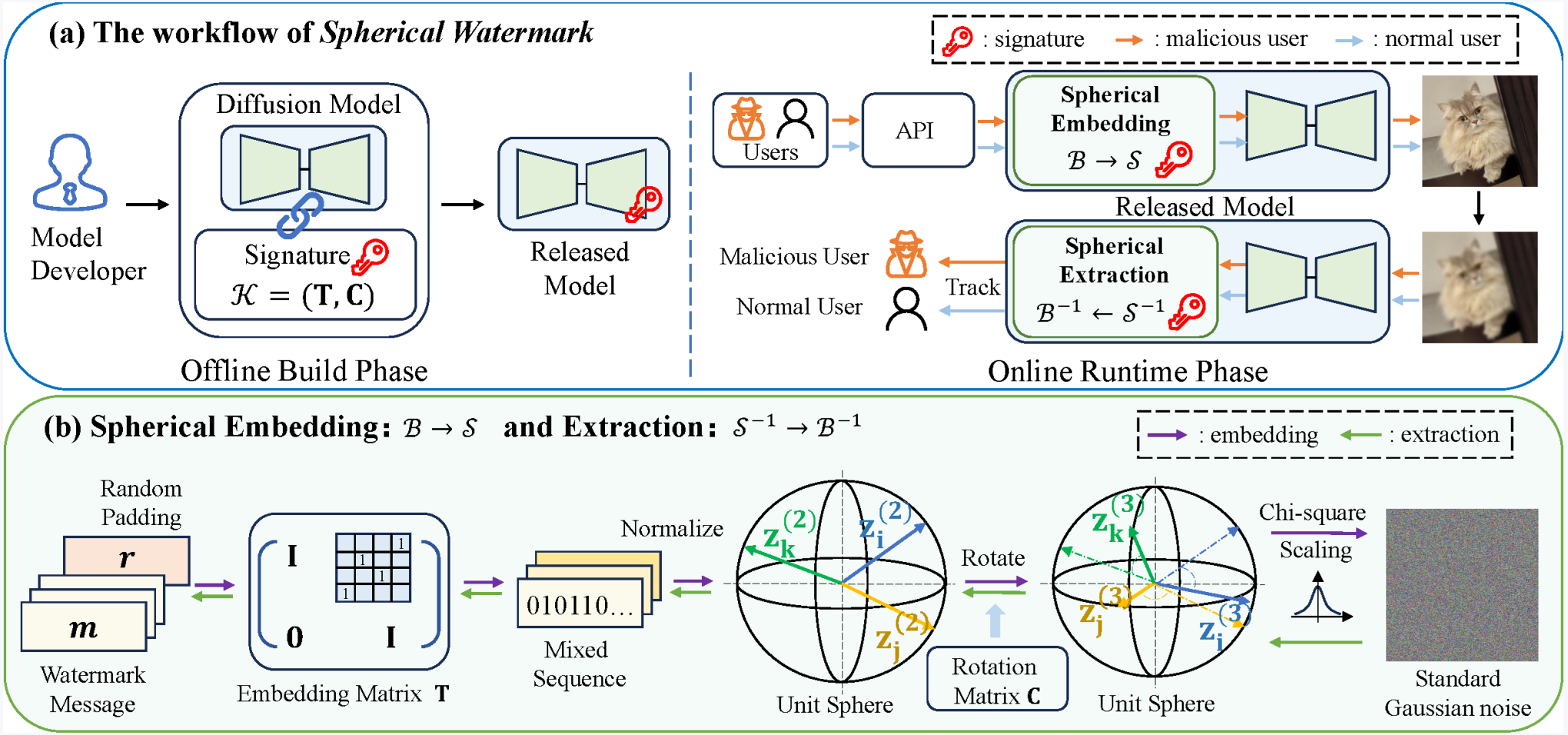


Fig.2. Comparison between existing lossless watermarking methods and our Spherical Watermark design.

4. Method overview.

$\mathcal{B}$: Binary Embedding Module
$\mathcal{S}$: Spherical Mapping Module

The complete transform is the signature K = (T, C): fixed offline, reused at runtime.



watermark bits → mixed binary sequence → unit-sphere mapping → rotated / scaled Gaussian latent → watermarked image

4. Module 1: Binary Embedding

The goal of the Binary Embedding module is to produce **high-entropy bitstream** from watermark bits.

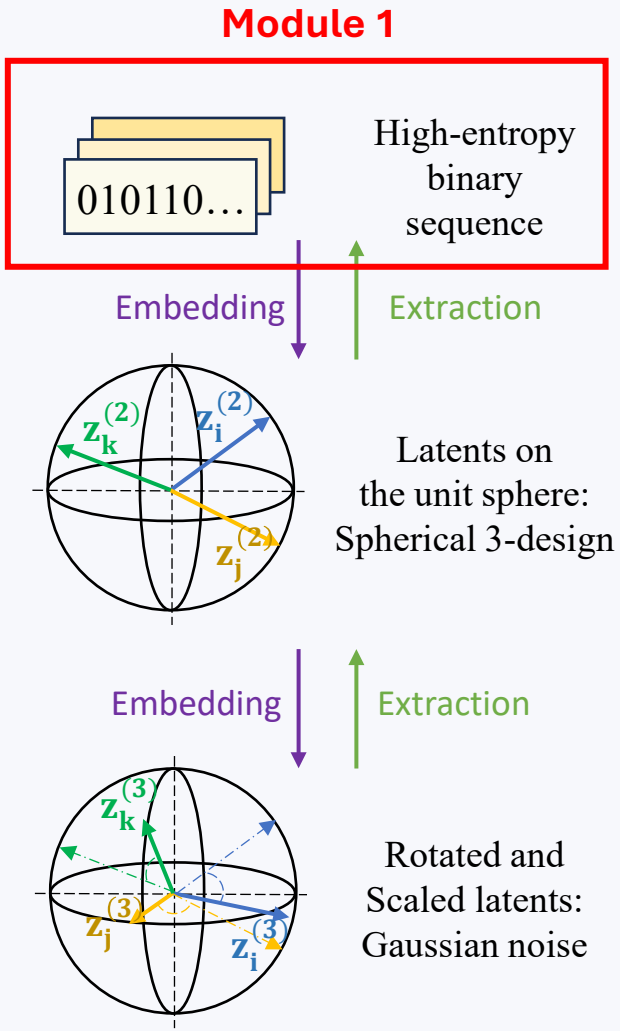
$$x = [m \quad m \quad \cdots \quad m \quad r]^{\top} \Rightarrow z^{(1)} = Tx$$

Repeated watermark + random padding — Invertible binary mixing

r is random padding follows Bernoulli(1/2). T is carefully designed sparse binary matrix, constructed by,

- **mixing each repeated watermark bit with a small subset of random padding bits**, so that the padding injects fresh randomness into the binary sequence;
- **assigning different repetitions of the same watermark bit to disjoint padding subsets**, which reduces correlation across repetitions while keeping the mapping invertible and lightweight.
- After binary embedding, each coordinate of the binary sequence follows Bernoulli(1/2). The sequence is **2-wise and 3-wise independent.** ***(Theorem 3.1)***

Random padding injects entropy into the repeated watermark bits.
Goal: Get Local Independence.



4. Module 2: Spherical Mapping

The goal of the Spherical Mapping module is transform the high-entropy bitstream into Gaussian noise input.

$$\boldsymbol{v} = 2\boldsymbol{z}^{(1)} - \mathbf{1}, \boldsymbol{z}^{(2)} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$$

Sign conversion and Normalize

$$\boldsymbol{z}^{(3)} = \boldsymbol{C}\boldsymbol{z}^{(2)},$$
$$r^2 \sim \chi^2(l_x), \boldsymbol{z}_w = r\boldsymbol{z}^{(3)}.$$

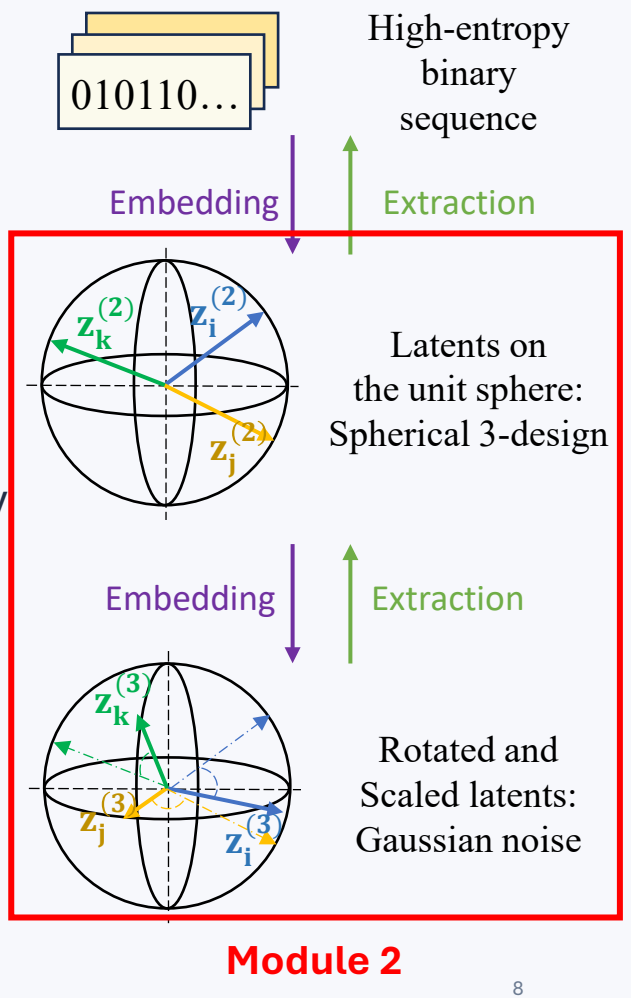
Rotate and scale

Diffusion Generation

$\boldsymbol{C}$ is a randomly sampled orthogonal matrix drawn from the Haar measure.

- **Approximate uniformity on the unit sphere**
  - ✓ After sign conversion and normalization, the latent codes lie on the unit sphere.
  - ✓ The resulting latent forms a spherical 3-design and is therefore approximately uniformly distributed on the unit sphere. ***(Definition 3.1 and Theorem 3.2)***

Transform the high-entropy binary sequence into Gaussian latent noise.
Goal: Preserve Gaussian prior.



Module 2

4. Module 2: Spherical Mapping

The goal of the Spherical Mapping module is transform the high-entropy bitstream into Gaussian noise input.

$$\boldsymbol{v} = 2\boldsymbol{z}^{(1)} - \mathbf{1}, \boldsymbol{z}^{(2)} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$$

Sign conversion and Normalize

$$\boldsymbol{z}^{(3)} = \boldsymbol{C}\boldsymbol{z}^{(2)},$$
$$r^2 \sim \chi^2(l_x), \boldsymbol{z}_w = r\boldsymbol{z}^{(3)}.$$

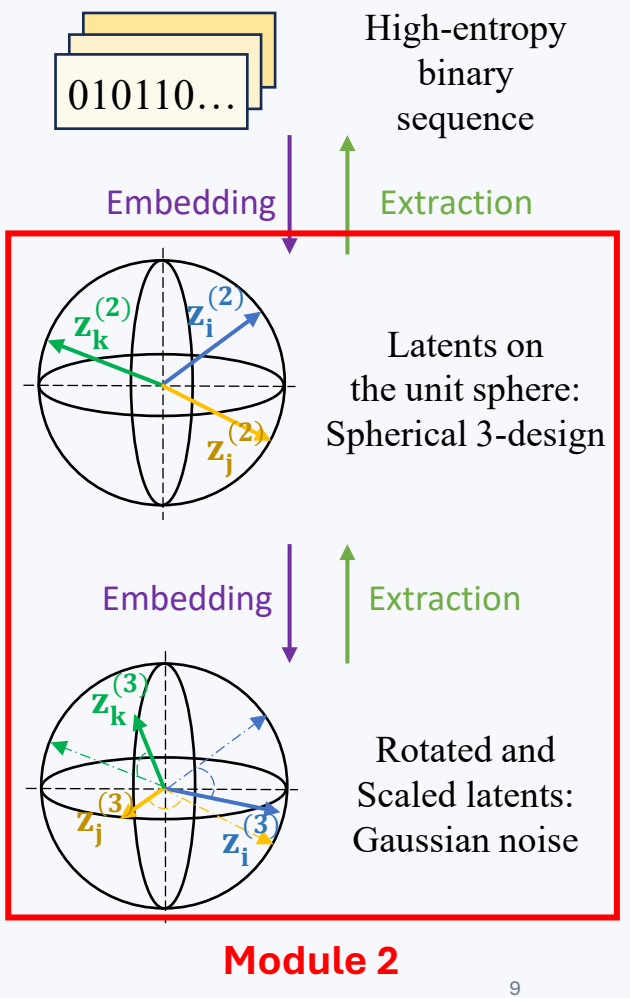
Rotate and scale

Diffusion Generation

$\boldsymbol{C}$ is a randomly sampled orthogonal matrix drawn from the Haar measure.

- **Recover Gaussian latent noise,**
  - ✓ Orthogonal rotation preserves the spherical 3-design property of the latent distribution. ***(Lemma 3.3)***
  - ✓ Chi-square scaling converts the rotated sample into Gaussian latent, yielding a final latent approximately distributed as standard multivariate Gaussian noise. ***(Lemma 3.4)***

Transform the high-entropy binary sequence into Gaussian latent noise.
Goal: Preserve Gaussian prior.



5. Experimental setup: models, datasets, metrics, baselines

Implementation details

- Backbone: Stable Diffusion v1.5 / v2.1, 512×512 generation, latent size = 4×64×64.
- Generation: 50-step DPM-Solver++; inversion: 50-step DDIM inversion.
- Experiments were run in PyTorch on four RTX 4090 GPUs.
- Traditional Baselines: DwtDct, DwtDctSvd, RivaGAN
- Latent-based Baselines: Tree-Ring, Gaussian Shading, PRC Watermark

Datasets & metrics

Prompts	**COCO + Stable Diffusion Prompts, 1000 each**
Undetectability	**FID + latent/image classifier accuracy**
Traceability	**ACC + TPR@1%FPR**
Robustness	**Post-processing + adversarial attacks**

Visual Quality: representative watermarked images generated by different methods.

6. Result I: undetectability

Key message

We train latent-level and image-level classifiers to decide whether a sample is watermarked.

Tree-Ring: 100%

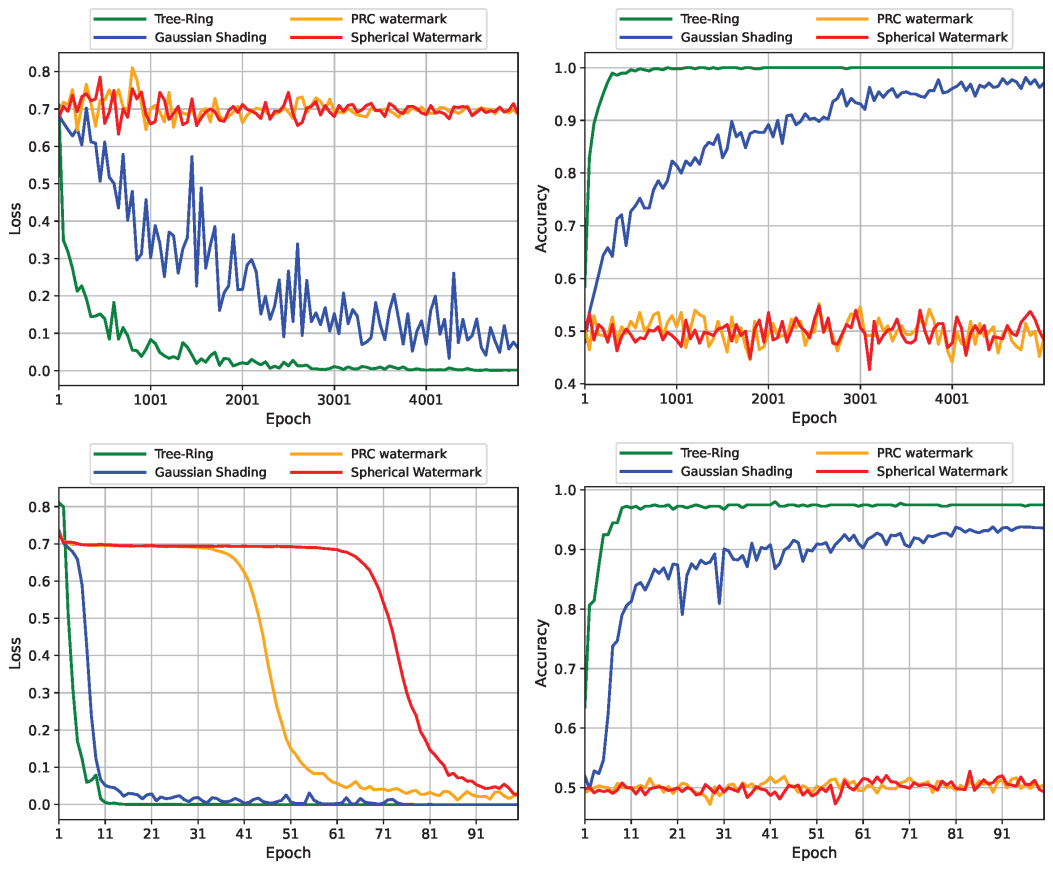
Gaussian Shading: 97%

PRC Watermark: ≈ 50%

Ours: ≈ 50%

So to a learned detector, Spherical Watermark is nearly indistinguishable from unwatermarked samples.

FID also stays essentially identical to the original model output. Example: on COCO / SD v2.1, Original = 46.8146 and Ours = 46.8132.



Latent-level Classification

Image-level Classification

Method	COCO		SDP	
	SD v1.5	SD v2.1	SD v1.5	SD v2.1
Original	$48.1256_{1.3744}$	$46.8146_{1.0617}$	$49.7041_{0.5425}$	$46.4060_{0.5231}$
DwtDct	$48.2975_{1.3918}$	$46.9771_{1.0702}$	$49.9853_{0.5385}$	$46.7304_{0.5163}$
DwtDctSvd	$48.7179_{1.4075}$	$47.4049_{1.0121}$	$51.0160_{0.6162}$	$47.5044_{0.6439}$
RivaGan	$48.7956_{1.3952}$	$47.6124_{1.1012}$	$51.2773_{0.6320}$	$47.8298_{0.6748}$
Tree-Ring	$49.3318_{1.5108}$	$47.8721_{1.1320}$	$50.6491_{1.0197}$	$47.3917_{0.7127}$
Gaussian Shading	$50.6968_{1.3200}$	$49.4379_{1.0326}$	$51.5221_{0.8773}$	$48.2539_{0.4859}$
PRC Watermark	$48.1348_{1.3074}$	$46.7544_{1.0748}$	$49.5250_{0.7651}$	$46.4157_{0.3445}$
Ours	$48.1224_{1.5489}$	$46.8132_{1.0962}$	$49.3894_{0.7475}$	$46.4311_{0.3695}$

FID Comparison

6. Result II: robustness and efficiency

WHY LOSSLESSNESS MATTERS

If a watermark is detectable, an attacker can train a detector and attack it directly. Lossless schemes are clearly stronger than lossy ones under adversarial perturbations.

Lossy Gaussian Shading with fixed key (ACC):

100.00 clean / 88.06 adversarial

Lossless Spherical Watermark with fixed key (ACC):

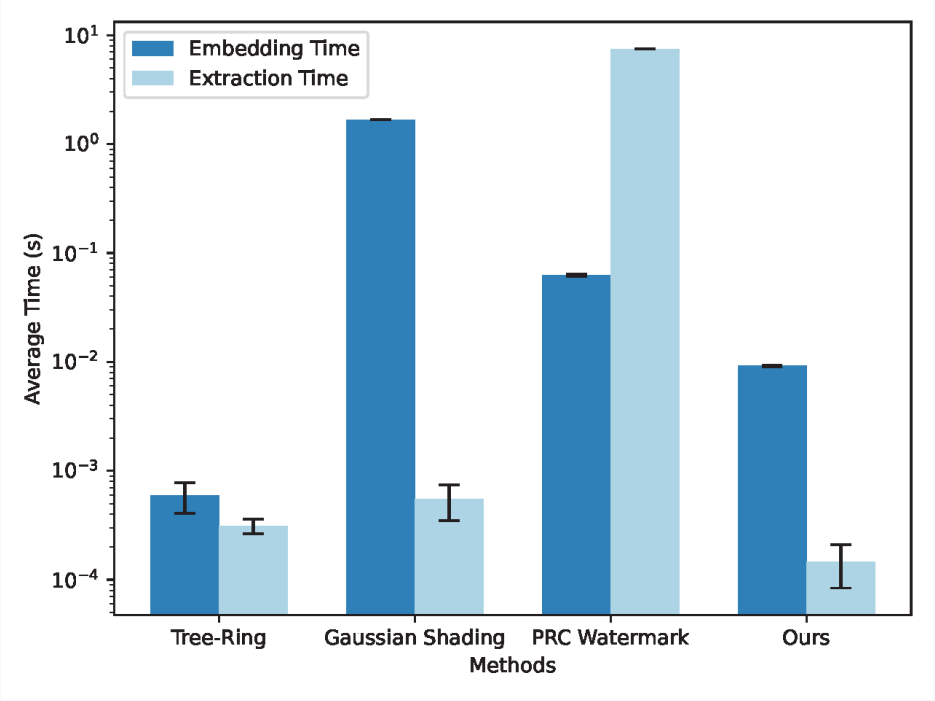
99.99 clean / 98.12 adversarial

Key Message

The experiment also reports extraction roughly **four orders of magnitude faster** than PRC Watermark.

Method	Metrics					
	ACC (Clean)	ACC (Post.)	ACC (Adv.)	TPR (Clean)	TPR (Post.)	TPR (Adv.)
DwtDct	$90.14_{1.15}$	$64.75_{1.08}$	$49.28_{0.00}$	$92.80_{3.14}$	$52.23_{3.41}$	$16.15_{0.02}$
DwtDctSvd	$100.00_{0.00}$	$93.21_{0.17}$	$48.95_{0.01}$	$100.00_{0.00}$	$91.94_{0.68}$	$17.05_{0.02}$
RivaGan	$99.68_{0.10}$	$96.78_{0.22}$	$52.31_{0.01}$	$100.00_{0.00}$	$99.13_{0.22}$	$26.75_{0.02}$
Tree-Ring	-	-	-	$100.00_{0.00}$	$98.85_{0.31}$	$6.71_{0.02}$
Gaussian Shading	$100.00_{0.00}$	$98.43_{0.04}$	$88.06_{0.11}$	$100.00_{0.00}$	$99.97_{0.04}$	$99.23_{0.00}$
PRC Watermark	$100.00_{0.00}$	$93.52_{0.20}$	$97.69_{0.07}$	$100.00_{0.00}$	$87.03_{0.39}$	$95.38_{0.00}$
Ours	$99.99_{0.01}$	$95.02_{0.08}$	$98.12_{0.04}$	$100.00_{0.00}$	$97.50_{0.18}$	$99.83_{0.00}$

Tracing Performance under Common Distortions.



Embedding and Extraction Time Comparison

(Y-axis at log-scale)

Q & A ?

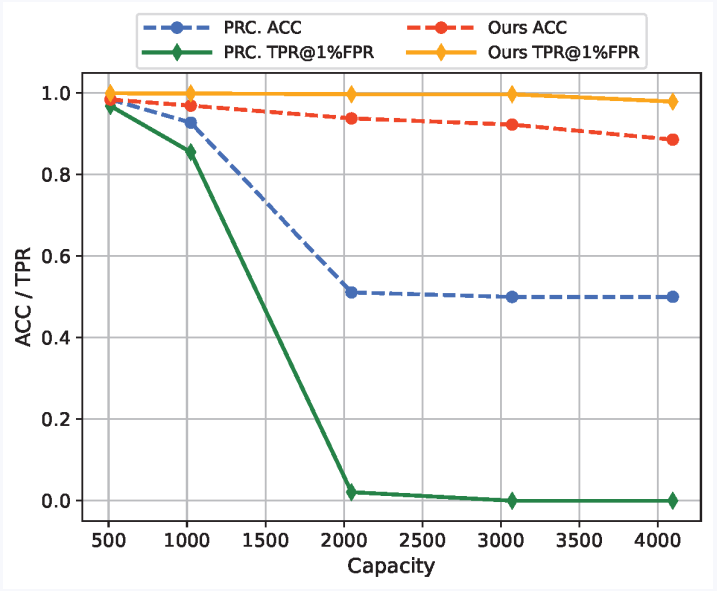
Thanks for listening.

6. Ablation Result

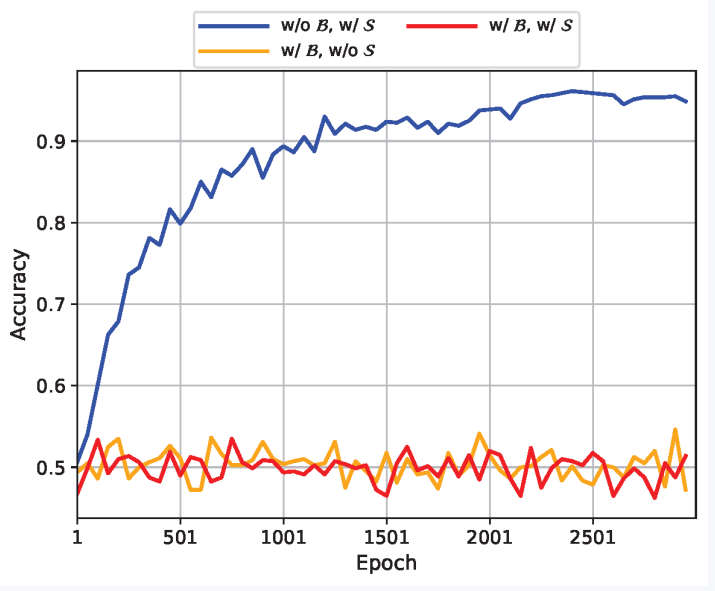
Which components matter most?

Binary Embedding supports undetectability.

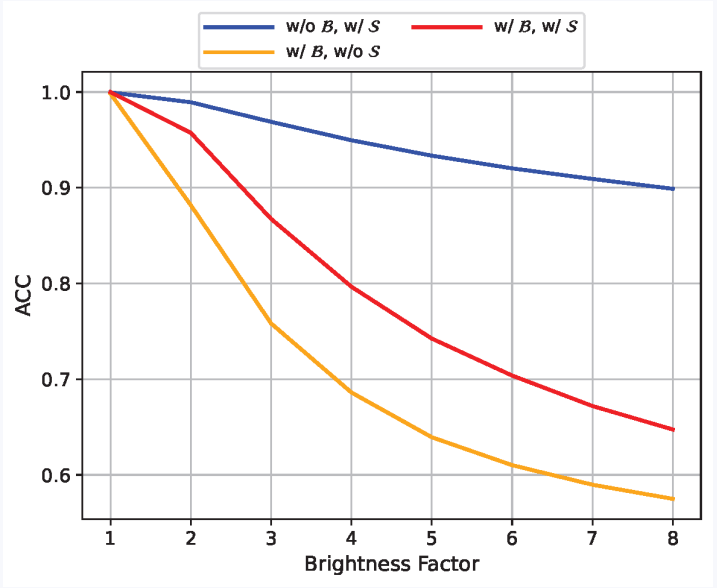
Spherical Mapping enhance robustness.



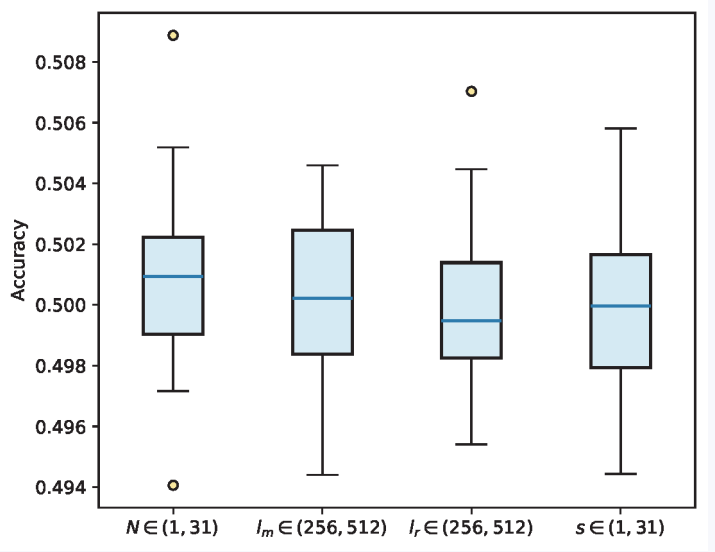
Watermark Length



Undetectability on Modules



Robustness on Modules



Parameter Settings

6. Experiments on Re-generation or Editing Attacks

Method	l_m	Re-generation			Editing			
		Regen-Diff	Rinse-2xDiff	Regen-VAE	MagicBrush	UltraEdit	InstructP2P.	GAN-edit
DwtDct	32	$49.96_{0.73}$	$49.91_{0.46}$	$50.27_{0.72}$	$50.25_{1.28}$	$49.28_{1.96}$	$50.56_{1.38}$	$49.88_{2.21}$
DwtDctSvd	32	$50.20_{0.93}$	$49.69_{0.74}$	$48.98_{0.58}$	$49.22_{1.76}$	$49.62_{1.62}$	$48.53_{0.88}$	$48.28_{3.44}$
RivaGan	32	$56.77_{0.56}$	$54.19_{0.53}$	$50.89_{0.33}$	$67.31_{1.24}$	$52.06_{2.33}$	$60.56_{1.52}$	$99.84_{0.14}$
TrustMark	100	$71.36_{0.46}$	$59.94_{0.30}$	$93.87_{0.33}$	$86.74_{1.85}$	$68.16_{1.18}$	$81.68_{3.54}$	$95.85_{1.34}$
Robust.	64	$96.90_{0.18}$	$93.22_{0.23}$	$94.15_{0.63}$	$94.36_{1.54}$	$80.09_{0.68}$	$96.44_{1.10}$	$98.33_{0.61}$
PRC	100	$99.26_{0.41}$	$94.22_{0.88}$	$75.91_{1.25}$	$94.14_{2.33}$	$81.53_{3.09}$	$83.53_{7.82}$	$100.00_{0.00}$
Ours	100	$99.63_{0.17}$	$97.70_{0.37}$	$87.48_{0.45}$	$93.96_{2.94}$	$86.11_{1.61}$	$92.53_{2.38}$	$100.00_{0.00}$