MULTI-BIT WATERMARKING WITH SEMANTIC INTEGRITY FOR BINARY-TOKEN VISUAL GENERATIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The widespread deployment of large-scale image generation models raises urgent questions about provenance, attribution, and content authenticity. We present a training-free multi-bit watermarking framework for binary-token visual autoregressive models. The key idea is to treat the model's latent binary variables as redundant watermark carriers: each payload bit is assigned to many token-bit positions, and generation is steered only through positions that can be modified at low semantic cost. We investigate two visual autoregressive models with different generation mechanisms: Infinity, where selected uncertain bits at the finest scale are forced after sampling, and BiGR, where Bernoulli marginals are softly biased during bidirectional masked prediction. Extraction is purely feed-forward: the image is re-encoded through the frozen tokenizer and the payload is recovered by majority voting over the carrier sets. The framework supports variable-length payloads (16, 32, and 64 bits) and, for Infinity, achieves near-perfect bit accuracy on COCO-prompted images with negligible degradation to image quality as measured by FID, KID, and CLIP Score. Building on the multi-bit capacity, we introduce a *semantic integrity* mechanism: the watermark payload encodes a DINOHash perceptual fingerprint of the generated image, enabling post-hoc detection of semantic tampering. Because the watermark resides in latent token space, it can persist through pixel-level edits, while the perceptual hash of a tampered image diverges from the recovered payload—revealing manipulation.

1 INTRODUCTION

Generative image models have reached a level of photorealism that makes AI-synthesized content increasingly difficult to distinguish from photographs. While this brings remarkable creative possibilities, it also creates serious societal risks: deepfakes, unauthorized reproduction of copyrighted styles, and the spread of misleading visual content (Mirsky & Lee, 2021). Watermarking—embedding an imperceptible, recoverable signal into generated images—is a principled countermeasure that enables *provenance tracking* (attributing an image to a specific model or user), *copyright enforcement*, and *tamper detection*.

Most prior watermarking research targets *diffusion models* (Fernandez et al., 2023; Wen et al., 2023; Zhao et al., 2024), which operate in a continuous latent space amenable to gradient-based embedding. *Visual autoregressive models* (VARs) (Tian et al., 2024), by contrast, generate images through discrete latent tokens, often organized as binary or vector-quantized codes. This discrete structure makes many diffusion-oriented watermarking techniques inapplicable, but it also exposes a useful opportunity: if a generator predicts many binary token variables, then a small payload bit can be embedded redundantly across a large set of latent positions. Recent work has begun to exploit this structure for zero-bit detection (Kerner et al., 2025); we ask instead how much decodable *payload* such latent bit-planes can carry, and what that payload can be used for.

In this paper, we develop a unified multi-bit watermarking framework for binary-token visual autoregressive models. The framework assumes only that, at some generation stage, the model exposes per-bit probabilities or logits before token commitment, and that the generated image can later be re-encoded by the corresponding frozen tokenizer. For each payload bit, we assign a redundant car-

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.

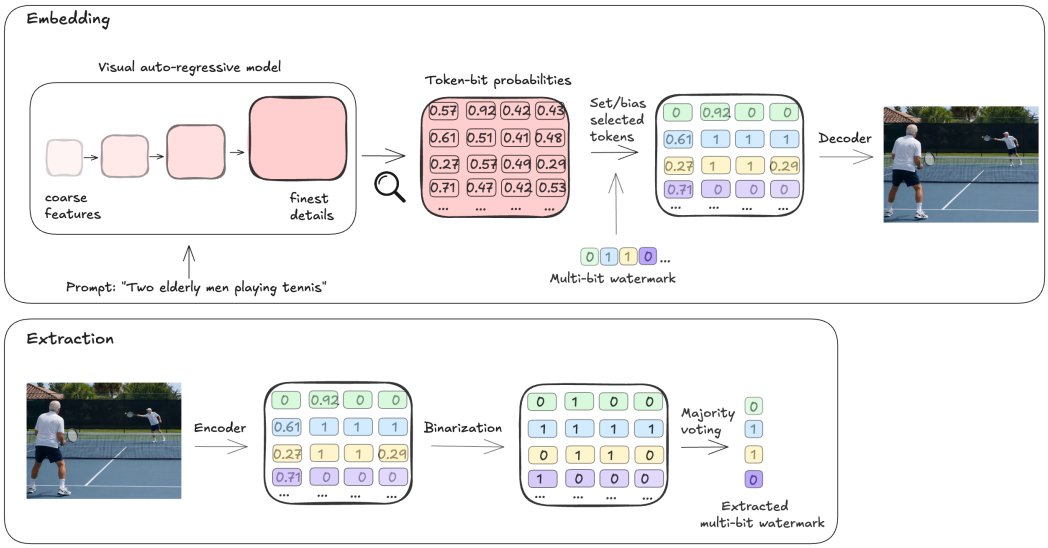

Figure 1: Overview of the framework. During generation, the model exposes probabilities or logits for binary latent token decisions. A deterministic embedding operator selects carrier positions and fixes or biases them toward the payload bits before decoding. For extraction, the image is re-encoded into latent token space, quantized to binary codes, and the payload is recovered by majority voting over the corresponding carrier sets.

rier set of latent token-bit positions. During generation, the sampler is intercepted and the carrier set is steered toward the desired payload bit, preferably at positions where the model is most uncertain (probability close to 0.5) or by applying a small pre-sampling bias. At verification time, the image is re-encoded and each payload bit is recovered by majority voting over its carrier set.

We instantiate this shared abstraction on two different host models. For **Infinity** (Han et al., 2025), a multi-scale residual binary VAR, the carrier space is the finest 64×64 scale with 32 binary bit-planes; embedding is implemented by forcing a selected subset of high-uncertainty sampled bits. For **BiGR** (Hao et al., 2025), a bidirectional masked-prediction model with a single-scale binary autoencoder, embedding is implemented by softly biasing Bernoulli marginals before the iterative token-selection step. Appendices A and B detail both instantiations.

Contributions.

1. **Watermarking abstraction for binary-token VARs.** We formulate embedding and extraction in terms of carrier sets, token-level uncertainty, model-specific sampler modulation, and re-encoding-based majority voting. Algorithms 1 and 2 capture the common procedure used for both Infinity and BiGR.

2. **Two concrete instantiations.** We provide separate implementation descriptions for Infinity and BiGR in Appendices A and B. Infinity uses force-based embedding at the final scale, whereas BiGR uses bias-based embedding inside the bidirectional masked-prediction loop.

3. **Scalable payload capacity.** Unlike zero-bit detection schemes for bitwise generators (Kerner et al., 2025), the scheme decodes an explicit message, supporting 16-, 32-, and 64-bit payloads by changing the carrier assignment and redundancy per bit, trading off embedding density, robustness, and image distortion.

4. **Semantic integrity via DINOHash.** We propose embedding a 64-bit DINOHash (Singhi et al., 2025; proteus-photos, 2024) perceptual fingerprint of the generated image as the watermark payload. At verification, a mismatch between the recovered payload and the current image hash flags semantic manipulation.

2 Related Work

Watermarking for generative models. Early neural watermarking methods such as HiDDeN (Zhu et al., 2018) and RivaGAN (Zhang et al., 2019) train encoder–decoder networks to hide messages in natural images. Stable Signature (Fernandez et al., 2023) fine-tunes the decoder of a latent diffusion model so that all generated images carry a fixed fingerprint extractable by a pre-trained decoder. Tree-Ring (Wen et al., 2023) embeds a ring pattern in the initial noise of diffusion models, enabling zero-shot detection without model access. Gaussian Shading (Yang et al., 2024) reinterprets the initial noise as a watermarked sample from a shifted Gaussian. Our work differs in targeting *discrete-token* autoregressive generators where neither a continuous noise space nor differentiable decoder fine-tuning is naturally available.

Visual autoregressive models. VQVAE (van den Oord et al., 2017) introduced learned discrete codebooks for image compression. VQGAN (Esser et al., 2021) coupled vector quantization with adversarial training to enable high-fidelity image synthesis via transformer-based token prediction. VAR (Tian et al., 2024) reframes autoregressive generation as next-*scale* prediction over multi-resolution token maps, achieving ImageNet benchmarks competitive with diffusion models. Infinity (Han et al., 2025) extends VAR to text-conditioned generation at high resolution using binary-coded residual tokens and classifier-free guidance. BiGR (Hao et al., 2025) instead uses a bidirectional masked-prediction transformer over a binary-autoencoder latent space. These two generators form the test cases for our watermarking abstraction.

Perceptual hashing and content authentication. Traditional perceptual hashes (pHash, dHash) capture low-frequency image structure but lack robustness to semantic-preserving transformations. DINOHash (Singhi et al., 2025; proteus-photos, 2024) computes compact binary fingerprints from DINOv2 ViT-S/14 (Oquab et al., 2024) features, achieving strong tolerance to benign edits while remaining sensitive to semantic content changes—properties we exploit for tampering detection.

Training-free and content-bound watermarking. Several methods embed watermarks into image generation without training, e.g., by modifying sampling distributions (Kirchenbauer et al., 2023) or latent-space initialization (Wen et al., 2023). Closest to our setting, BitMark (Kerner et al., 2025) biases the logits of a bitwise autoregressive generator with a green/red list, and IndexMark (Tong et al., 2025) swaps generated VQ indices for similar codebook entries; both decide detection by a statistical test on the token distribution rather than decoding a payload. For diffusion models, SEAL (Arabi et al., 2026) binds content by seeding the initial noise from a hash of a caption embedding, but the binding is verified by matching inverted noise patches rather than decoded.

3 Background: Binary-Token Visual Autoregressive Models

Binary latent tokenizers. The models considered in this work represent an image through a frozen encoder– decoder pair. The encoder maps an image x to a binary latent representation $Z \in \{0,1\}^{N\times C}$, where N is the number of spatial tokens and C is the number of binary bit-planes per token; the decoder maps Z back to pixels. The values of N and C are model-specific. For example, Infinity uses a multi-scale residual binary tokenizer and exposes $N = 4{,}096$ tokens with $C = 32$ bit-planes at its finest scale, whereas BiGR uses a single-scale binary autoencoder whose token count depends on image resolution and whose bit-depth is model-dependent.

Autoregressive or masked token generation. A visual autoregressive generator produces the latent code by repeatedly predicting and committing token variables. Depending on the architecture, the active variables may be processed causally, scale by scale, or through bidirectional masked refinement. At each generation step t, the model exposes logits or Bernoulli probabilities P_t over the currently active binary carrier entries $\mathcal{A}_t \subseteq \Omega$, where $\Omega = \{(i,c) : 1 \le i \le N, 1 \le c \le C\}$ is the set of all candidate token-bit positions. The sampler then commits some or all entries in $\mathcal{A}_t$. Our framework inserts a small hook at this sampler boundary: it either modifies the probabilities before sampling or modifies selected sampled bits immediately after sampling, while leaving the model weights unchanged.

Low-cost carrier positions. The semantic cost of changing a binary carrier entry is related to the model's confidence. When a Bernoulli probability $p_{i,c}$ is close to 0.5, the model assigns similar likelihood to both bit values, so steering the bit toward a payload value is less likely to distort the final image. When $p_{i,c}$ is close to 0 or 1, overriding the bit conflicts with a confident model decision and can have a stronger perceptual effect. This motivates the uncertainty cost

$$\rho_{i,c} = |p_{i,c} - 0.5|, \tag{1}$$

which is used directly by the Infinity instantiation and indirectly by the BiGR instantiation through confidence-based masked-token selection.

Carrier redundancy. A single payload bit is not embedded into a single token bit. Instead, each payload bit w_k is assigned to a carrier set $\mathcal{C}_k \subseteq \Omega$ containing many token-bit entries. This redundancy is crucial because re-encoding through a tokenizer is imperfect and because some image transformations perturb individual bits. Extraction therefore uses an aggregate vote over $\mathcal{C}_k$ rather than trusting any single latent position.

4 Watermarking Method

4.1 Overview

Let $\mathbf{w} = (w_0, w_1, \ldots, w_{K-1}) \in \{0,1\}^K$ be a K-bit watermark payload. The method embeds $\mathbf{w}$ into the binary latent carrier space of a visual autoregressive generator at inference time. The generator and tokenizer are frozen. The only required access is a sampler hook that exposes the probabilities or logits of active binary token variables before they are committed.

The method has three model-independent components. First, a deterministic carrier assignment $a : \Omega \to \{0, \ldots, K-1\}$ partitions token-bit positions into redundant carrier sets $\mathcal{C}_k = \{e \in \Omega : a(e) = k\}$, one set per payload bit. Second, during generation, active carrier entries are steered toward their assigned payload value by a model-specific modulation rule. Third, during extraction, the image is re-encoded by the frozen tokenizer and each payload bit is recovered by majority voting over its carrier set.

4.2 Embedding

Algorithm 1 gives the general embedding procedure. The algorithm deliberately separates the common watermark logic from the model-specific sampler intervention. The procedure Select chooses the entries to carry a payload bit, typically by selecting the entries with the smallest uncertainty cost in Eq. 1 or by using all currently active entries in the carrier group. The procedure PreSampleBias optionally modifies probabilities or logits before sampling, and PostSampleForce optionally overwrites selected sampled bits after sampling. We ablate the carrier-selection criterion and the selection ratio for Infinity in Appendix K.

The abstraction covers both implementations used in this paper. For Infinity, $\mathcal{A}_t$ is the set of binary entries at the finest scale; Select chooses the M entries in each carrier set with probability closest to 0.5; PreSampleBias is the identity; and PostSampleForce sets selected sampled bits to w_k. For BiGR, $\mathcal{A}_t$ is the set of still-masked token bits at a refinement step; Select uses the watermark carrier group; PreSampleBias shifts Bernoulli marginals toward w_k; and PostSampleForce is not used. The corresponding implementation details are given in Appendices A.3 and B.3.

4.3 Extraction

Given a possibly transformed image $\hat{x}$, extraction is performed by re-encoding $\hat{x}$ through the frozen tokenizer and aggregating votes over the same carrier assignment used during embedding. Let $\hat{Z} \in \{0,1\}^{N \times C}$ be the recovered binary latent code, and let $\mathcal{V}_k$ denote the vote set for payload bit k. In the simplest case, $\mathcal{V}_k = \mathcal{C}_k$. If an implementation stores or deterministically reconstructs a smaller selected subset, $\mathcal{V}_k$ may instead be that subset.

The extracted payload is $\hat{\mathbf{w}} = (\hat{w}_0, \ldots, \hat{w}_{K-1})$, and bit accuracy is $\mathrm{BA} = \frac{1}{K} \sum_k \mathbb{1}[\hat{w}_k = w_k]$.

Algorithm 1 Watermark Embedding

Require: Generator $\mathcal{G}$, prompt y, seed r, payload $\mathbf{w} \in \{0,1\}^K$, carrier assignment a, strength parameters η
Ensure: Watermarked image x_w

Initialize $\mathcal{G}$ with prompt y and seed r
for each generation step t **do**
  Run $\mathcal{G}$ to the sampler/commit boundary and expose probabilities P_t over active carrier entries $\mathcal{A}_t$
  $\widetilde{P}_t \leftarrow P_t$
  for $k = 0$ **to** $K-1$ **do**
    $\mathcal{D}_{k,t} \leftarrow \{e \in \mathcal{A}_t : a(e) = k\}$
    if $\mathcal{D}_{k,t} \neq \emptyset$ **then**
      Compute costs ρ_e for $e \in \mathcal{D}_{k,t}$
      $\mathcal{S}_{k,t} \leftarrow \textsc{Select}(\mathcal{D}_{k,t}, \rho; \eta)$
      $\widetilde{P}_t \leftarrow \textsc{PreSampleBias}(\widetilde{P}_t, \mathcal{S}_{k,t}, w_k; \eta)$
    end if
  end for
  Commit tokens using $\widetilde{P}_t$ and the original generation schedule
  Optionally apply PostSampleForce to committed entries in $\bigcup_k \mathcal{S}_{k,t}$
end for
Decode the final latent code with the frozen decoder to obtain x_w
return x_w

Algorithm 2 Watermark Extraction & Detection

Require: Image $\hat{x}$, frozen tokenizer encoder E, carrier assignment a, optional reference payload $\mathbf{w}$ and threshold θ
Ensure: Extracted bits $\hat{\mathbf{w}}$, detection score S, decision d

$\hat{Z} \leftarrow E(\hat{x})$ {Recover binary carrier code}
for $k = 0$ **to** $K-1$ **do**
  $\mathcal{V}_k \leftarrow \textsc{VoteSet}(a, k)$
  $v_k \leftarrow |\mathcal{V}_k|^{-1} \sum_{e \in \mathcal{V}_k} \hat{Z}_e$
  $\hat{w}_k \leftarrow \mathbb{1}[v_k > 0.5]$
end for
if reference payload $\mathbf{w}$ is provided **then**
  $S \leftarrow \sum_{k=0}^{K-1} (2w_k - 1)(v_k - 0.5)$
  $d \leftarrow \mathbb{1}[S > \theta]$
else
  $S \leftarrow$ UNDEFINED; $d \leftarrow$ UNDEFINED
end if
return $\hat{\mathbf{w}}, S, d$

All reported bit accuracies and detection scores are measured on the 8-bit PNG artifact: each generated image is quantized to 8 bits per channel and reloaded before being re-encoded through the tokenizer, rather than extracting from the unquantized floating-point output.

4.4 Watermark Detection

To decide whether an image is watermarked with a known payload, we compute the signed detection score

$$S = \sum_{k=0}^{K-1} (2w_k - 1)(v_k - 0.5), \tag{2}$$

which is large and positive when the soft votes align with the reference payload. We calibrate a threshold θ on the null distribution of unwatermarked generations to target a 1% false-positive rate (FPR). An image is classified as watermarked if $S > \theta$.

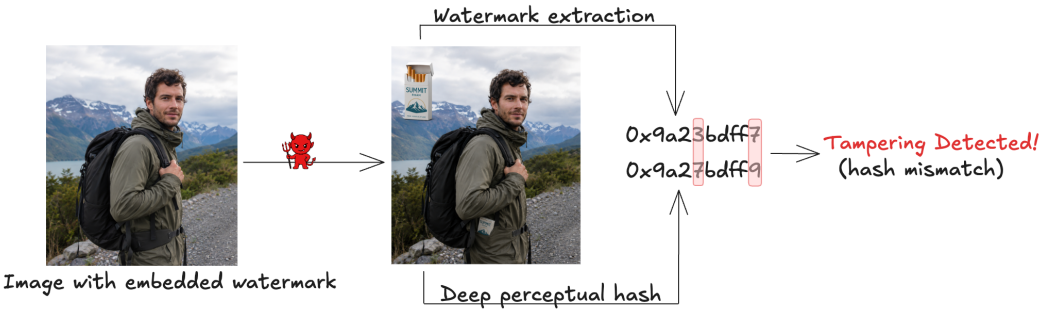

Figure 2: **Semantic integrity verification.** The watermark payload encodes a perceptual hash of the original image. After a possible semantic intervention (here, insertion of an object), the image is flagged as tampered when the Hamming distance between the extracted payload hash and a freshly computed hash of the observed image exceeds the threshold τ.

5 Semantic Integrity

Motivation. Watermarks are traditionally used for ownership verification: given the watermark key, any party can confirm that an image originated from a specific generator instance. However, once an image is released, a malicious actor can manipulate it semantically—adding, removing, or replacing objects—while preserving enough visual similarity to deceive human viewers. Our multi-bit capacity enables a stronger property: *semantic integrity*, whereby the watermark itself encodes a content-bound fingerprint that can detect post-generation tampering.

DINOHash fingerprint. We use DINOHash (Singhi et al., 2025; proteus-photos, 2024), a perceptual hashing scheme based on a DINOv2 ViT-S/14 (with registers) (Oquab et al., 2024) feature extractor, to compute a compact binary fingerprint of an image. DINOHash outputs a 96-bit hash; the 96 bits are ordered by importance (most- to least-important, i.e. most- to least-robust to benign distortions), and we keep the 64 most-robust as $H(x) \in \{0,1\}^{64}$. DINOHash is designed to be stable under minor photometric and geometric distortions (compression, slight crop, brightness change) while being sensitive to semantic content changes such as object insertion or removal.

5.1 Semantic Integrity Protocol

Issuance. Generate the image x *without* a watermark, compute its DINOHash fingerprint $H = H(x) \in \{0,1\}^{64}$, re-generate the same image with the identical prompt and seed while embedding H as the 64-bit payload (Algorithm 1), and release the watermarked image x_w.

Verification. Given a possibly tampered image x', extract the payload $\hat{H}$ (Algorithm 2), compute the current hash $H' = H(x')$, and flag x' as tampered if $d = \mathrm{HD}(H', \hat{H}) \geq \tau$, where $\tau = 8$ bits.

Why it works. The payload lives in the tokenizer's latent bit-planes, so pixel-level edits need not erase the re-encoded votes and $\hat{H}$ keeps matching the issued hash. DINOHash, by contrast, responds to semantic change, so a salient insertion moves H' and pushes d past τ. On an unmodified image only embedding-induced hash drift and extraction error remain, and both stay below τ.

6 Experimental Evaluation

6.1 Experimental Setup

Dataset. We generate 10,000 Infinity images from captions sampled from the COCO 2014 validation set (Lin et al., 2014) and 5,000 class-conditional BiGR images from ImageNet (Russakovsky et al., 2015) class labels. Detection thresholds are the 99th percentile of each model's unwatermarked

	Infinity				BiGR-L-512			
Payload	**Bit Acc.**	**Perfect**	**TPR@1%**	**AUROC**	**Bit Acc.**	**Perfect**	**TPR@1%**	**AUROC**
16-bit	99.91	98.90	99.99	0.999956	97.67	84.86	99.92	0.999978
32-bit	99.77	95.18	99.98	0.999988	96.83	66.16	100.00	0.999997
64-bit	98.71	60.40	99.99	0.999996	94.79	33.44	100.00	1.000000

Table 1: Watermark extraction and detection without attack. Infinity: 10,000 COCO-prompted images at 1024×1024. BiGR-L-512: 5,000 class-conditional images at 512×512 with bias β=0.06. Bit Acc., Perfect, and TPR@1% are in %; "Perfect" is the fraction of images with all K bits correct, and TPR@1% and AUROC are against the unwatermarked-generation null of each model.

generations (10,000 for Infinity, 5,000 for BiGR). A null of real photographs of equal count (COCO; ImageNet) gave near-identical TPR and AUROC, so we report the generation null only.

Model. We use the publicly released Infinity-2B checkpoint with `infinity_vae_d32reg` ($V = 32$ bit-planes) and FLAN-T5-XL text conditioning. All images are generated at 1024×1024 resolution. For BiGR we use the BiGR-L-512 checkpoint at 512×512 resolution, yielding N=1,024 tokens with C=32 bit-planes. Payloads are embedded with an additive Bernoulli-probability bias of β=0.06 and the interleaved carrier assignment $\ell(i) = i \bmod K$, and we evaluate $K \in \{16, 32, 64\}$.

Baseline. We compare against the post-hoc multi-bit watermark TrustMark (Bui et al., 2025), applied to the same Infinity images with a 32-bit payload under the same protocol (Appendix M). Its detection and image quality are on par with ours, but, being trained against common image distortions, it has higher bit accuracy and is more robust, most clearly to cropping.

Hardware. For reference, on a server with 4 $\times$ RTX 3090 GPUs, non-watermarked generation takes ≈3.5 s/image at 1024×1024 resolution, and watermarked generation takes ≈3.9 s/image. Semantic-integrity issuance runs both passes per image (Section 5.1), plus two DINOHash computations.

6.2 Watermark Accuracy and Detection

Table 1 reports bit accuracy and perfect-match rate (all K bits correct) across payload sizes without any post-generation attack. Both models recover the payload with high accuracy, though BiGR-L-512 falls behind Infinity, and the gap widens with payload size—most visibly at 64 bits, where BiGR's perfect-match rate drops noticeably while Infinity's remains high.

The same table reports TPR at 1% FPR and AUROC against the unwatermarked-generation null of each model. Despite BiGR-L-512's lower bit accuracy, detection remains near-saturated for both: TPR@1% is at or above 99.9% and AUROC at or above 0.9999 across all payload sizes.

6.3 Robustness to Common Attacks

We evaluate the 32-bit watermark under seven common image transformations applied after generation (Table 2), attacking 1,000 watermarked images per model and 1,000 matched unwatermarked images (same prompts/labels and seeds), all scored at the deployment threshold θ (99th percentile of the clean unwatermarked null, held fixed across attacks), so TPR@θ and FPR@θ are the measured operating point under each attack. Under this suite, both watermarks are robust to photometric distortions (blur, color jitter, grayscale, JPEG): FPR@θ stays within $[0.1, 1.5]\%$ and TPR@θ is at or near 100%. Spatial cropping is the strongest distortion, as it destroys the token grid; for Infinity it also shifts the null—FPR@θ reaches 33.7% (center) and 19.9% (random crop), with recalibrated TPR@1%* of 98.4% and 61.9% (AUROC 0.998/0.968)—while BiGR's null is stable under every attack (FPR@$\theta \leq 2.6\%$, TPR@1%$^* \geq 99.7\%$, AUROC ≥ 0.9999). Appendix N extends this comparison to BitMark's own, stronger suite via a zero-bit adaptation of our scheme. Blur, JPEG and grayscale are still handled there; the remaining gaps are on transformations that permute the token grid—cropping and flips—to which position-indexed carriers are inherently vulnerable. Both schemes degrade under arbitrary rotation.

	Infinity				BiGR-L-512			
Attack	**Bit Acc.**	**TPR@θ**	**FPR@θ**	**TPR@1%***	**Bit Acc.**	**TPR@θ**	**FPR@θ**	**TPR@1%***
None	99.73	100.00	0.60	100.00	96.82	100.00	0.90	100.00
Gaussian Noise	87.81	94.00	0.40	94.90	92.08	99.00	0.10	99.80
Gaussian Blur	97.52	100.00	1.20	100.00	96.08	100.00	0.70	100.00
Color Jitter	98.96	100.00	1.50	100.00	96.18	100.00	1.00	100.00
Center Crop	83.35	99.80	33.70	98.40	82.58	100.00	0.80	100.00
Random Crop	68.87	96.10	19.90	61.90	87.33	99.90	2.60	99.70
Grayscale	97.82	100.00	1.10	100.00	96.27	100.00	1.10	100.00
JPEG Compress	94.83	99.80	0.80	99.80	95.86	100.00	0.60	100.00

Table 2: Robustness to image transformations (32-bit watermark): 1,000 watermarked images per model and 1,000 matched unwatermarked negatives are attacked and scored at the deployment threshold θ, so FPR@θ is the measured false-positive rate under each attack. *TPR at a per-attack recalibrated 1%-FPR threshold; bootstrap 95% CIs for FPR@θ are within ± 0.7 pp except Infinity Center/Random Crop ($[30.4, 36.8]$/$[17.5, 22.3]$) and BiGR Random Crop ($[1.6, 3.6]$). Attack parameters and per-attack AUROC are listed in Tables 7 and 8.

	Infinity, 1024^2			**BiGR-L-512**, 512^2, β=0.06		
Method	**FID** $\downarrow$	**KID** ($\times 10^2$) $\downarrow$	**CLIP** $\uparrow$	**FID** $\downarrow$	**KID** ($\times 10^2$) $\downarrow$	**CLIP** $\uparrow$
Unwatermarked	30.57	1.33	25.79	23.34	0.80	24.65
16-bit	31.13	1.32	25.75	19.79	0.79	24.46
32-bit	30.95	1.34	25.71	19.33	0.77	24.44
64-bit	30.75	1.32	25.73	19.47	0.78	24.48

Table 3: Image quality, watermarked vs. unwatermarked, within each model. FID/KID references: 40,504 real COCO 2014 validation images at 1024^2 for Infinity, 50,000 real ImageNet validation images at 512^2 for BiGR-L-512. CLIP Score uses the generation text (COCO caption; ImageNet class label).

6.4 Image Quality

Table 3 quantifies the perceptual impact of watermarking using Fréchet Inception Distance (FID) (Heusel et al., 2017), Kernel Inception Distance (KID) (Bińkowski et al., 2021), and CLIP Score (Radford et al., 2021) (ViT-L/14). Within each model, watermarking leaves CLIP Score essentially unchanged relative to the unwatermarked baseline, and FID/KID show no meaningful quality degradation.

6.5 Semantic Integrity Evaluation

We evaluate semantic integrity on Infinity only; Appendix J explains why BiGR's per-step bias would inflate the embedding-induced hash drift.

Tampering is simulated by pasting a visually salient forbidden object (e.g. cigarette pack or vodka bottle) at a random location, scaled so its visible (non-transparent) pixels cover 5%, 10%, or 15% of the image area. Figure 3 sweeps τ, Table 4 reports the $\tau = 8$ operating point (clean-image false-alarm rate 2.34%), and Figure 6 plots the underlying distance distributions. An intervention also damages the payload, since an opaque paste overwrites the latent carriers beneath it, so we additionally report the payload Hamming distance $\mathrm{HD}(\hat{H}, w)$ against the payload embedded at issuance. Appendix H separates the two causes by repeating these conditions with the paste geometry held identical per image and only the pasted content varied: substituting the image's own displaced content for the foreign object lowers the hash displacement by 5.17 bits $[5.01, 5.33]$ at 15% area, so the signal follows the replacement content rather than the occlusion.

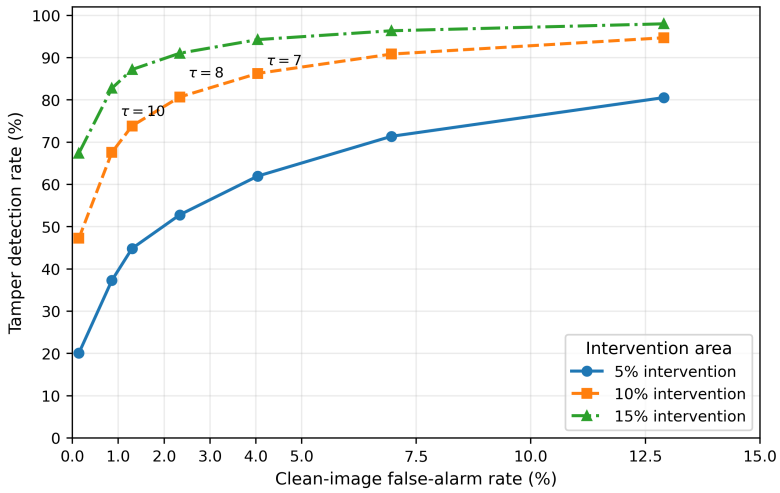

Figure 3: **Semantic tamper detection across intervention sizes.** Detection rate against clean-image false-alarm rate as the Hamming-distance threshold τ varies, for semantic interventions whose visible object pixels cover 5%, 10%, and 15% of the image area. Markers indicate integer values of τ; detection improves consistently with intervention size. The $\tau = 8$ operating point is reported in Table 4.

Condition	Detection (%)	Mean HD	Payload HD
Unmodified (clean)	—	2.52	0.84
5% intervention	52.80	8.75	1.36
10% intervention	80.68	12.74	1.97
15% intervention	91.04	15.73	2.71

Table 4: Semantic integrity (Infinity) at $\tau = 8$ bits (clean-image false-alarm rate 2.34%), 5,000 images per condition. Mean HD is the tamper signal d (extracted payload vs. recomputed hash); Payload HD compares the extracted and embedded payloads and measures watermark damage alone (Appendix H).

7 Conclusion

We introduced a training-free multi-bit watermarking framework for binary-token visual autoregressive models: payload bits are spread over redundant latent token-bit carriers, steered through a model-specific sampler hook (uncertainty-guided forcing for Infinity, a soft probability bias for BiGR), and recovered by re-encoding and majority voting. Embedding a DINOHash fingerprint as the payload turns the same channel into a semantic-integrity check: a large Hamming mismatch between the recovered and the recomputed hash flags manipulation.

Limitations. Extraction relies on access to the corresponding frozen tokenizer encoder and on the tokenizer's quantization structure remaining sufficiently stable after image transformations. Strong adversarial purification attacks that regenerate the image through a different model may destroy the watermark. The semantic integrity threshold $\tau = 8$ is empirically calibrated and may require adjustment for different manipulation types, generators, or image domains, and payload damage from benign transformations raises the false-alarm rate, to 48% under JPEG compression (Appendix I).

Future work. Extensions include adaptive carrier assignment, per-user payloads, stronger error-correcting codes, robustness to diffusion-based purification attacks, and a full cross-model evaluation of the Infinity and BiGR instantiations under a shared benchmark, including extending the semantic-integrity protocol to BiGR.

REPRODUCIBILITY STATEMENT

The method is fully specified in the paper: Algorithms 1 and 2 define embedding and extraction, Section 4.4 the detection statistic and its threshold calibration, and Section 5.1 the semantic-integrity issuance and verification protocol. Model-specific instantiations, including architectures, carrier spaces, and all hyperparameters, are given in Appendix A (Infinity) and Appendix B (BiGR); the optional PGD refinement in Appendices D–F; the exact attack parameters in Table 7; and the key-management assumptions in Appendix C. Section 6 states the datasets, the public checkpoints, the sample counts, and the hardware. All randomness—the watermark key, prompt sampling, per-image generation, carrier selection, and the stochastic attacks, whose realizations are drawn from seeds derived deterministically from the image identifier and attack name—is controlled by fixed seeds, and each run records its watermark key(s) and run configuration. Generation and extraction run under mixed precision, whose kernels are not bit-reproducible, so quantities that fall near a decision boundary can change between runs and aggregate rates can move by a few tenths of a percentage point; Appendix H quantifies this for the semantic-integrity threshold. Each run also caches the per-image records, soft votes, and null distributions, so every table, ROC curve, and threshold can be recomputed offline without regenerating images. Appendix J documents the calibration and known limitations of the semantic-integrity threshold, and Appendix H the equal-geometry control that separates semantic change from carrier damage.

AI USE STATEMENT

We used generative AI tools in two ways. First, to write the code for all experiments, under the authors' supervision; the authors reviewed all of this code. Second, in preparing the manuscript: to draft parts of the text, to polish and format it for readability, and to review it for gaps in the methodology and experiments. We mainly used Claude Opus 5, Kimi K3, and GLM 5.3, and other frontier models to a lesser extent. The authors reviewed and revised all AI-assisted text and take full responsibility for the content of this paper.

REFERENCES

Kasra Arabi, R. Teal Witter, Chinmay Hegde, and Niv Cohen. Seal: Semantic aware image watermarking, 2026. URL https://arxiv.org/abs/2503.12172.

Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans, 2021. URL https://arxiv.org/abs/1801.01401.

Tu Bui, Shruti Agarwal, and John Collomosse. Trustmark: Robust watermarking and watermark removal for arbitrary resolution images. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18629–18639. IEEE, 2025.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024. URL http://jmlr.org/papers/v25/23-0870.html.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021.

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

Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 15733–15744, June 2025.

Shaozhe Hao, Xuantong LIU, Xianbiao Qi, Shihao Zhao, Bojia Zi, Rong Xiao, Kai Han, and Kwan-Yee K. Wong. BiGR: Harnessing binary latent codes for image generation and improved visual representation capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=1Z6PSw7OL8.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

Louis Kerner, Michel Meintz, Bihe Zhao, Franziska Boenisch, and Adam Dziedzic. Bitmark: Watermarking bitwise autoregressive image generative models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proceedings of the International Conference on Machine Learning*, 2023.

Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, pp. 740–755, 2014.

Yisroel Mirsky and Wenke Lee. The creation and detection of deepfakes: A survey. *ACM Computing Surveys*, 54(1):1–41, 2021.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

proteus-photos. Dinohash perceptual hash. https://github.com/proteus-photos/dinohash-perceptual-hash, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning*, pp. 8748–8763, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. URL https://arxiv.org/abs/1409.0575.

Shree Singhi, Aayan Yadav, Aayush Gupta, Shariar Ebrahimi, and Parisa Hassanizadeh. Provenance detection for ai-generated images: Combining perceptual hashing, homomorphic encryption, and ai detection models, 2025. URL https://arxiv.org/abs/2503.11195.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems*, 2024.

Yu Tong, Zihao Pan, Shuai Yang, and Kaiyang Zhou. Training-free watermarking for autoregressive image generation, 2025. URL https://arxiv.org/abs/2505.14673.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 2017.

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and difficult to remove. In *Advances in Neural Information Processing Systems*, 2023.

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

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

Xuandong Zhao, Kexun Zhang, Zihao Su, Saastha Vasan, Ilya Grishchenko, Christopher Kruegel, Giovanni Vigna, Yu-Xiang Wang, and Lei Li. Invisible image watermarks are provably removable using generative ai. *Advances in neural information processing systems*, 37:8643–8672, 2024.

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*, pp. 657–672, 2018.

This appendix gives the model-specific instantiation details of the framework introduced in Section 4, together with additional results and analyses. Appendices A and B detail the Infinity (force-based) and BiGR (bias-based) instantiations of Algorithms 1 and 2. Appendix C states the symmetric-key setting. Appendices D–F present the optional PGD refinement and its PGD-refined counterparts of the accuracy, detection, robustness, and quality results reported in Section 6. The remaining appendices give additional figures (Appendix G), an equal-geometry control separating semantic change from watermark damage in the tamper signal (Appendix H), the false-alarm rate and detection under common image transformations (Appendix I), further notes on the semantic-integrity protocol (Appendix J), an Infinity carrier-selection ablation (Appendix K), and paired qualitative examples (Appendix L). Appendix N adapts the scheme to zero-bit detection for a head-to-head comparison with BitMark (Kerner et al., 2025).

A Watermarking the Infinity Model

This appendix gives the Infinity-specific instantiation of Algorithms 1 and 2. Infinity uses a force-based version of the general framework: we first let the original sampler draw the finest-scale binary token matrix, then overwrite only selected high-uncertainty entries with the target payload bit.

A.1 Architecture and Carrier Space

Infinity (Han et al., 2025) is a 2B-parameter visual autoregressive transformer for text-to-image generation. Images are represented through a multi-scale residual binary tokenizer with a sequence of $S = 13$ scales $\{s_1, s_2, \ldots, s_{13}\}$. At the finest scale s_{13}, each image is represented as a $64 \times 64 = 4{,}096$ spatial grid of tokens. Each token is a $C = 32$-bit binary code, yielding a carrier matrix $Z \in \{0,1\}^{4096\times 32}$ and $4{,}096 \times 32 = 131{,}072$ binary carrier entries per image.

Given a text prompt, Infinity encodes the prompt with FLAN-T5-XL (Chung et al., 2024) and generates tokens scale by scale. At scale s_{13}, the transformer predicts a Bernoulli logit for each spatial position b and bit-plane v. We denote the corresponding probability by $p_{b,v} = \sigma(\mathcal{L}_{b,v})$. The key property used by the watermark is that the probability matrix $P \in [0,1]^{4096\times 32}$ is available immediately before sampling. Entries with $p_{b,v} \approx 0.5$ are the least expensive to modify because the model has nearly equal preference for either bit value.

A.2 Carrier Assignment

For a K-bit payload $\mathbf{w} \in \{0,1\}^K$, each watermark bit w_k is assigned to a set of candidate positions $\mathcal{C}_k$ in the finest-scale carrier matrix. The current implementation uses bit-plane assignment:

$$v = k \bmod 32. \tag{3}$$

For $K = 32$, all $4{,}096$ spatial positions in the selected bit-plane are candidates. For $K = 16$, each bit instead spans two bit-planes, $v{=}k$ and $v{=}k{+}16$, over all $4{,}096$ positions, giving $8{,}192$ candidates, and the extraction vote averages the two planes. For $K = 64$, the spatial positions are split into two groups of $2{,}048$ tokens, so that two payload bits can share the same bit-plane without sharing the same spatial carriers. We use a deterministic contiguous spatial split: the first and second halves of the 64×64 token grid form the two groups.

Algorithm 3 Infinity Watermark Embedding (Force-Based)

Require: Logits $\mathcal{L} \in \mathbb{R}^{4096\times 32}$, payload $\mathbf{w} \in \{0,1\}^K$, selection count M
Ensure: Watermarked token sample $Z \in \{0,1\}^{4096\times 32}$
$Z \leftarrow \textsc{OriginalSample}(\mathcal{L})$
$P \leftarrow \sigma(\mathcal{L})$ {Bernoulli probabilities}
for $k = 0$ **to** $K-1$ **do**
  Identify candidate positions $\mathcal{C}_k$ by bit-plane and token group
  Compute $u_{b,v} = |P_{b,v} - 0.5|$ for $(b,v) \in \mathcal{C}_k$
  $\mathcal{S}_k \leftarrow \mathrm{argmin}_M\{u_{b,v} : (b,v) \in \mathcal{C}_k\}$
  $Z_{b,v} \leftarrow w_k$ for all $(b,v) \in \mathcal{S}_k$
end for
return Z

Algorithm 4 Infinity Watermark Extraction & Detection

Require: Image $\hat{x}$, payload $\mathbf{w} \in \{0,1\}^K$, frozen Infinity VAE encoder
Ensure: Extracted bits $\hat{\mathbf{w}}$, detection score S
$\hat{Z} \leftarrow \text{VAE-}\textsc{Encode}(\hat{x})|_{s_{13}}$
for $k = 0$ **to** $K-1$ **do**
  $v_k \leftarrow |\mathcal{C}_k|^{-1} \sum_{(b,v)\in\mathcal{C}_k} \hat{Z}_{b,v}$
  $\hat{w}_k \leftarrow \mathbb{1}[v_k > 0.5]$
end for
$S \leftarrow \sum_{k=0}^{K-1}(2w_k - 1)(v_k - 0.5)$
return $\hat{\mathbf{w}}, S$

A.3 Force-Based Embedding

The Infinity hook replaces the scale-s_{13} sampling function with WatermarkedSample. It operates on the logit tensor $\mathcal{L} \in \mathbb{R}^{4096\times 32}$ immediately before the original sampler is called. For each payload bit k, the hook computes $u_{b,v} = |\sigma(\mathcal{L}_{b,v}) - 0.5|$ over $(b,v) \in \mathcal{C}_k$, selects the M lowest-cost (most uncertain) positions, and then sets those sampled bits to the target value w_k. We use $M = 3{,}200$ for $K = 16$, $M = 1{,}600$ for $K = 32$, and $M = 800$ for $K = 64$, corresponding to an approximately constant selection ratio of 39% of candidate positions per payload bit.

The selected sets $\mathcal{S}_k$ are not stored after embedding; extraction votes over the full carrier set $\mathcal{C}_k$, matching the experiments of Section 6.

A.4 Extraction and Detection

Given a possibly transformed image $\hat{x}$, we re-encode it through the frozen Infinity VAE encoder and extract the reconstructed binary token matrix $\hat{Z} \in \{0,1\}^{4096\times 32}$ at scale s_{13}. For each watermark bit k, we compute a soft vote over the carrier set

$$v_k = |\mathcal{C}_k|^{-1} \sum_{(b,v)\in\mathcal{C}_k} \hat{Z}_{b,v}, \tag{4}$$

and recover the hard bit by thresholding:

$$\hat{w}_k = \mathbb{1}[v_k > 0.5]. \tag{5}$$

When the reference payload is known, the signed detection score is the same as Eq. 2.

B Watermarking the BiGR Model

This appendix gives the BiGR-specific instantiation of Algorithms 1 and 2 for **BiGR** (Hao et al., 2025), a second visual autoregressive model. BiGR shares the binary-token representation that our method exploits, but differs from Infinity in two ways that change the embedding strategy: (i) BiGR is a *bidirectional masked-prediction* transformer rather than a left-to-right causal model, and (ii) our

embedding is *bias-based* rather than *force-based*: instead of overwriting sampled bits at a hand-picked subset of positions, we nudge the model's pre-sampling distribution toward the watermark and let the generative process commit the watermark itself. The extraction stage carries over from the main method unchanged; the BiGR-specific PGD refinement is described in Sec. F.

Notation. To avoid a clash with the notation of Section 4 (where K is the payload length), we write C for the per-token codebook bit-depth (the number of binary bits per token; $C \in \{24, 32\}$), N for the number of tokens, and keep K for the watermark length. BiGR represents a $H \times H$ image as $N = H^2/16^2$ tokens, each a C-bit binary code (N=256 at 256 px; N=1024 at 512 px). We denote the watermark payload $\mathbf{w} \in \{0,1\}^K$ and the bias magnitude β.

B.1 Architecture

BiGR comprises two components, both frozen at inference: a binary tokenizer—the Binary Autoencoder (B-AE)—and a class-conditional bidirectional transformer that generates within the B-AE's binary latent space.

Binary tokenizer (B-AE). The B-AE maps an image $x \in \mathbb{R}^{3\times H\times W}$ to a *single-scale* binary latent at stride 16: a grid of $N = (H/16)^2$ positions, each a C-bit binary code. A convolutional encoder produces continuous features; a 1×1 projection followed by a sigmoid yields, at each position, C independent Bernoulli probabilities $p \in [0,1]^C$ over the C bit-planes. Binarization converts these to the code $z \in \{0,1\}^C$—stochastically ($z_c \sim \mathrm{Bernoulli}(p_c)$) during training and deterministically ($z_c = \mathbb{1}[p_c > 0.5]$) at inference—with a straight-through estimator routing gradients through the rounding step. Each code weights a learned C-entry embedding table and the selected embeddings are summed to a continuous vector that the decoder renders back to pixels. Two properties are central to our scheme: (i) a token is a C-bit *vector* of independently-modelled bits, not an index into a categorical codebook; and (ii) the latent is a single $N \times C$ grid, not a multi-scale pyramid.

Generator. BiGR's generator is a class-conditional bidirectional transformer over the binary canvas. Given a partially masked canvas, it predicts—for every masked position—a C-vector of independent Bernoulli probabilities; the model emits per-bit marginals, never a joint distribution over the 2^C possible codes. Generation proceeds by iterative masked prediction (a discrete form of diffusion) with classifier-free guidance combining conditional and unconditional predictions. The output of each step is the guidance-combined per-bit probability, which the loop then ranks by confidence and commits by sampling (Sec. B.2).

Contrast with Infinity. Infinity's tokenizer is a 13-scale residual binary pyramid and its generator decodes causally, scale by scale; our Infinity watermark therefore forces bits at the finest scale only. BiGR's tokenizer is single-scale, so every token of the unique grid is an equally natural carrier, and—because its bits are independent Bernoulli marginals—the most direct way to steer them is to nudge those marginals rather than overwrite a sampled code. This is the architectural reason BiGR admits a bias-based embedding where Infinity uses bit-forcing.

B.2 BiGR Generation

Unlike Infinity's scale-by-scale causal decoding, BiGR generates all N tokens *simultaneously* via iterative masked prediction. Starting from a fully masked token canvas, the transformer runs T refinement steps (we use T=10). At step t it predicts a Bernoulli probability $p_i \in [0,1]^C$ for every still-masked token i, ranks tokens by confidence $c_i = \|p_i - 0.5\mathbf{1}\|$ (perturbed by Gumbel noise for diversity), and *commits* the top-ranked tokens by drawing $z_i \sim \mathrm{Bernoulli}(p_i)$. A cosine mask schedule controls how many tokens are committed per step, so that confident tokens are fixed early and serve as context for later, less confident ones. The final binary canvas $Z \in \{0,1\}^{N\times C}$ is decoded to pixels by the frozen Binary Autoencoder (B-AE).

B.3 Bias-Based Embedding

We group the N tokens into K groups of $G = N/K$ tokens each, one group per watermark bit. By default we use *interleaved* assignment $\ell(i) = i \bmod K$, so that each watermark bit is spread

Algorithm 5 BiGR Bias-Based Watermark Embedding

Require: Watermark $\mathbf{w} \in \{0,1\}^K$, bias β, token count N, bit-depth C, iterations T
Ensure: Watermarked token canvas $Z \in \{0,1\}^{N \times C}$
Initialise masked canvas; $\ell(i) \leftarrow i \bmod K$ {interleaved groups}
for $t = 1$ **to** T **do**
  $p_i \leftarrow$ BiGR-Predict(canvas, $i \in$ masked) $\{p_i \in [0,1]^C\}$
  $\tilde{p}_i \leftarrow \mathrm{clip}\big(p_i + \beta(2w_{\ell(i)}-1)\mathbf{1}_C,\ 0,\ 1\big)$
  $c_i \leftarrow \|\tilde{p}_i - 0.5\mathbf{1}\| +$ Gumbel noise
  $\mathcal{S}_t \leftarrow$ top-confidence masked tokens (cosine schedule)
  $z_i \sim \mathrm{Bernoulli}(\tilde{p}_i)$ for $i \in \mathcal{S}_t$; commit to canvas
end for
$Z \leftarrow$ canvas
return Z

across the entire image; this makes recovery robust to spatially correlated B-AE re-encoding errors. (A *consecutive* assignment $\ell(i) = \lfloor i/G \rfloor$ is also supported.)

Embedding does not touch the sampler's output directly. Recall (Sec. B.1) that at each refinement step the transformer emits, *after* classifier-free guidance has combined the conditional and unconditional predictions, a C-vector of independent Bernoulli probabilities $p_i \in [0,1]^C$ for every masked token i. We inject the watermark bias into these per-bit marginals—between guidance and sampling, leaving the transformer and guidance otherwise untouched—so that the bias acts on the model's own probability space rather than overriding a drawn sample. For token i assigned to watermark bit $\ell(i)$, the bias is applied uniformly across all C bit-planes:

$$\tilde{p}_i \;=\; \mathrm{clip}\big(p_i + \beta\,(2w_{\ell(i)}-1)\,\mathbf{1}_C,\ 0,\ 1\big)\,. \tag{6}$$

A watermark bit $w_{\ell(i)}{=}1$ pushes all C bits of every token in the group toward 1; $w_{\ell(i)}{=}0$ pushes them toward 0. The biased $\tilde{p}_i$ then drives the standard BiGR step: confidence is computed from $\tilde{p}_i$ and the Bernoulli sample is drawn from $\tilde{p}_i$. Algorithm 5 summarizes the procedure, which drops into BiGR's generation loop as a per-step probability hook.

Bias-before-selection feedback loop. Because the bias enters *before* confidence ranking, watermark-aligned tokens have their confidence inflated and are committed *earlier* in the refinement process. Committed tokens then become visible context for subsequent steps, so the model is steered toward further watermark-aligned predictions—a self-reinforcing loop in which the bias does not merely tilt a coin toss but actively restructures the generation trajectory. This is a property unique to bidirectional iterative decoding; Infinity's one-pass causal sampler has no analogue, which is why BiGR uses a soft bias where Infinity uses a hard bit-forcing rule.

B.4 Extraction

Given an image $\hat{x}$, we re-encode it through the frozen B-AE encoder and take the deterministic binary codes $\hat{Z} \in \{0,1\}^{N \times C}$ (the pre-quantization continuous projection is used only for the optional PGD refinement, Sec. D; model-specific details are given in Sec. F). For each watermark bit k we aggregate all $G \cdot C$ binary codes in its group and threshold:

$$s_k = \sum_{i:\,\ell(i)=k} \sum_{c=1}^{C} \hat{Z}_{i,c}, \qquad \hat{w}_k = \mathbb{1}\big[s_k > \tfrac{GC}{2}\big]\,, \tag{7}$$

i.e. a majority vote over the $G \cdot C$ bits of the group. Bit accuracy is $\mathrm{BA} = \frac{1}{K}\sum_k \mathbb{1}[\hat{w}_k = w_k]$, and the signed detection score (Eq. 2) applies with $v_k = s_k/(GC)$ in place of the mean soft vote.

B.5 Representative Results

We evaluate the BiGR instantiation on **BiGR-L-512**, a class-conditional bidirectional model at 512×512 resolution, yielding $N{=}1{,}024$ tokens with $C{=}32$ bit-planes. We embed a $K{=}32$-bit

payload with interleaved carrier assignment $\ell(i) = i \bmod K$ and bias β=0.06, over 5,000 class-conditional generations. Generation uses classifier-free guidance scale 2.5 and Gumbel temperature 0.01 with a 'down' schedule. With $G{=}N/K{=}32$ tokens per group, each payload bit is carried by $G\,C{=}1{,}024$ binary votes. Detection uses two null distributions of 5,000 images each: unwatermarked BiGR generations (`bigr_nonwm`, matching the generative prior of the positives) and real ImageNet photographs (`real_imagenet`, a random subset of the ImageNet validation set, drawn with the same count). We report the 16-, 32-, and 64-bit configurations under the same protocol.

The no-PGD BiGR results are reported alongside Infinity in Tables 1, 2, and 3; we give the BiGR-specific setup here and defer the PGD-refined counterparts to Sec. F.

Discussion. The BiGR port validates that our framework is not specific to Infinity's causal multi-scale pipeline: it transfers to any binary-token generator whose sampler can be intercepted, and the bias-based variant demonstrates that a soft, distortion-minimising injection can substitute for hard bit-forcing when the host model is iterative. The bias-before-selection feedback loop is an incidental but useful consequence of BiGR's bidirectional refinement—one that has no counterpart in the causal setting and that effectively amplifies a small bias into a strongly recoverable watermark.

B.6 Interpreting the BiGR FID Shift

We observe that BiGR watermarking shifts FID from 23.34 to approximately 19.3–19.8, while KID remains similar and CLIP Score decreases slightly. Since FID compares marginal feature distributions rather than class- and seed-matched image pairs, this suggests that watermarking changes the distribution of BiGR outputs while retaining comparable distribution-level quality.

One possible explanation is BiGR's iterative generation mechanism. Applying the watermark bias before token-confidence ranking can change which tokens are committed early and thereby influence later predictions. We chose this intervention point because it produced the best visual quality among those examined in our preliminary experiments, although it can still alter the sampling trajectory. We also qualitatively observed that the evaluated BiGR checkpoint generally produced less visually faithful generations than Infinity, despite using the generation settings reported in the original BiGR paper. Because the models differ in conditioning, resolution, and evaluation data, this is not a controlled quantitative comparison; however, BiGR's lower baseline fidelity and greater variability may make its outputs more sensitive to small sampling interventions. We present this as a possible explanation rather than an established causal conclusion.

Watermarked and unwatermarked BiGR generations use identical settings, so the observed shift is not caused by differing generation configurations. It is specific to the interaction between bias-based embedding and BiGR's iterative dynamics and does not affect the semantic-integrity principle. We validate that protocol on Infinity, whose localized final-scale intervention is better suited to the current two-pass issuance procedure.

C Symmetric-Key Setting

We assume that the watermark embedder and authorized verifier share a secret key κ. The key seeds a pseudorandom assignment of latent token-bit positions to watermark carrier groups. The generator, tokenizer, embedding algorithm, extraction algorithm, and detection threshold may be public, while the carrier assignment remains unknown to unauthorized parties. An attacker may apply common image transformations or semantic edits and may have access to the public model weights, but does not know κ. The secret assignment is intended to make targeted watermark removal or payload replacement more difficult. We do not claim public verifiability, non-repudiation, or robustness after key compromise; adaptive key-recovery and detector-query attacks are outside the scope of this work.

D PGD Refinement

When one or more payload bits remain on the wrong side of the voting threshold—particularly at larger payload sizes—we optionally apply a small Projected Gradient Descent (PGD) refinement

Payload	Bit Acc. (%)	Perfect Match (%)	Mean Correct
16-bit	99.91	98.91	15.98/16
32-bit	99.79	95.44	31.93/32
64-bit	98.74	61.21	63.19/64

Table 5: Watermark extraction accuracy with PGD refinement (10,000 COCO-prompted images; $\varepsilon = 0.004$, 10 steps, $\alpha = 0.1$).

Payload	TPR@1% (%)	AUROC
16-bit	99.99	0.9999
32-bit	100.00	1.0000
64-bit	99.99	1.0000

Table 6: Watermark detection at 1%-FPR with PGD refinement ($\varepsilon = 0.004$, 10 steps, $\alpha = 0.1$).

step in pixel space. The refinement uses a differentiable surrogate of the tokenizer encoder to increase the vote margin of incorrectly decoded bits, while constraining the perturbation by $\|\delta\|_\infty \leq \varepsilon$. In our experiments we use $\varepsilon = 0.004$ and 10 gradient steps. The exact surrogate loss is model-specific: for Infinity we use a loss on incorrect decoded bit-planes, while for BiGR we use a group-count loss over the B-AE pre-quantization projection. The model-specific PGD equations and the PGD-refined results are given below (Sec. E and Sec. F); Section 6 reports no-PGD results throughout.

E PGD Refinement for the Infinity Model

For high payload sizes, especially $K = 64$, a small number of bits may remain on the wrong side of the extraction threshold after the initial force-based embedding. We repair these with the PGD refinement of Sec. D: starting from the watermarked image x_w, we optimize

$$\delta = \arg\min_{\|\delta\|_\infty \leq \varepsilon} \mathcal{L}_{\text{wrong}}(x_w + \delta), \tag{8}$$

where $\mathcal{L}_{\text{wrong}}$ is a signed-margin loss that pushes each group's vote mean toward the target bit, weighted toward incorrectly decoded bits (weight 1) with a small preservation term on already-correct bits (weight α=0.1).

PGD-refined results. The tables below report the PGD-refined counterparts of the no-PGD results of Tables 1, 2, and 3, all with an L_∞ budget $\varepsilon = 0.004$, 10 steps (step size $\varepsilon/10$), and soft-preservation weight $\alpha = 0.1$. For reference, the per-attack AUROC of the no-PGD 32-bit watermark, supplementing Table 2, is given in Table 8.

F PGD Refinement for the BiGR Model

The bias β trades watermark strength against perceptual distortion; a small β preserves image quality but may leave several bits on the wrong side of the $GC/2$ threshold after the stochastic B-AE round-trip. We repair these with the PGD refinement of Sec. D: using the differentiable pre-quantization B-AE projection $z(\hat{x}{+}\delta)$, we minimise the squared error between the expected per-group 1-count and the target:

$$\min_{\|\delta\|_\infty \leq \varepsilon} \sum_{k=0}^{K-1} \big(s_k(\hat{x}{+}\delta) - w_k\, GC\big)^2, \tag{9}$$

with $\varepsilon = 0.004$ and 10 signed-gradient steps, then re-extract from $\hat{x}{+}\delta$. The perturbation is imperceptible ($\approx 1/255$ per channel) yet, because it acts directly on the B-AE input, it increases group vote margins and improves the probability that borderline groups cross the threshold.

Attack	Parameters
None	—
Gaussian Noise	$\sigma = 0.05$
Gaussian Blur	kernel size $= 5$
Color Jitter	hue $= 0.05$, saturation $= 0.3$, contrast $= 0.3$
Center Crop	crop keeps 80% of each side (centered), white-padded to original size
Random Crop	as Center Crop, with a uniform random offset
Grayscale	—
JPEG Compress	quality $= 75$

Table 7: Parameters of the image transformations used in all robustness evaluations of this work (Tables 2, 8, 9, 13, 19, and 16). The identical transformation is applied to the watermarked images and to the matched unwatermarked negatives; in Table 16, to the untampered and tampered version of each image. The zero-bit BitMark comparison (Appendix N) uses a separate, stronger, BitMark-aligned suite (Table 20).

Attack	Infinity	BiGR-L-512
None	1.0000	1.0000
Gaussian Noise	0.9896	0.9999
Gaussian Blur	1.0000	1.0000
Color Jitter	1.0000	1.0000
Center Crop	0.9979	1.0000
Random Crop	0.9681	0.9999
Grayscale	1.0000	1.0000
JPEG Compress	0.9999	1.0000

Table 8: Per-attack AUROC for the no-PGD 32-bit watermark, supplementing Table 2 (which reports Bit Acc., TPR@θ, FPR@θ, and TPR@1%* for the same 1,000 watermarked images and 1,000 matched unwatermarked negatives per model). Attack parameters are listed in Table 7.

PGD-refined results. PGD raises bit accuracy from 97.67% to 97.69% at 16-bit, from 96.83% to 97.17% at 32-bit, and from 94.79% to 96.26% at 64-bit, lifting the perfect-payload rate from 84.86% to 85.02% (16-bit), 66.16% to 68.60% (32-bit), and 33.44% to 44.88% (64-bit) (Table 11; no-PGD counterparts in Table 1). PGD triggers on $757/5{,}000$ images at 16-bit, $1{,}692/5{,}000$ at 32-bit, and $3{,}328/5{,}000$ at 64-bit—those with at least one borderline bit—at a mean perturbation $\|\delta\|_\infty$=6.5×10^{-4} ($\approx 0.16/255$ per channel), 8.7×10^{-4} ($\approx 0.22/255$), and 1.2×10^{-3} ($\approx 0.30/255$) respectively, well below the ε budget. Detection remains saturated (Table 12); PGD widens the watermarked score margin from $+1.022$ to $+1.023$ at 16-bit, from $+2.064$ to $+2.128$ at 32-bit, and from $+3.695$ to $+4.410$ at 64-bit. Image quality is essentially unchanged by the perturbation: relative to the watermarked images before refinement, PGD moves FID by at most 0.06 and CLIP Score by at most 0.013 across all payload sizes. The gaps from the unwatermarked baseline that appear in Table 14—FID falling from 23.34 to 19.79, 19.30, and 19.41 at 16, 32, and 64-bit, with CLIP Score dropping by about 0.19, 0.21, and 0.18—are the watermark's quality bias, present before refinement, not an effect of the perturbation. Table 13 reports the PGD-refined counterparts of the no-PGD robustness table (Table 2): bit accuracy improves slightly on every attack (≤ 0.25 pp) and all detection columns (TPR@θ, FPR@θ, TPR@1%*) are unchanged, so the PGD perturbation does not trade robustness for the accuracy gain. The no-PGD BiGR per-attack AUROC is given in Table 8.

G Additional Figures

H An Equal-Geometry Control for the Tamper Signal

Verification compares two hashes, $d = \mathrm{HD}(H', \hat{H})$, and an edit moves both of them. It moves H' because the content changed, which is what the protocol means to measure. It also moves $\hat{H}$, because an opaque paste overwrites the latent carriers beneath it. A rise in d is therefore consistent

Attack	Bit Acc. (%)	TPR@θ	FPR@θ	TPR@1%*	AUROC
None	99.73	100.00	0.60	100.00	1.0000
Gaussian Noise	87.81	94.00	0.40	94.80	0.9895
Gaussian Blur	97.51	100.00	1.20	100.00	1.0000
Color Jitter	98.97	100.00	1.50	100.00	1.0000
Center Crop	83.30	99.90	33.70	98.50	0.9983
Random Crop	68.85	96.30	19.90	62.00	0.9682
Grayscale	97.84	100.00	1.10	100.00	1.0000
JPEG Compress	94.83	99.70	0.80	99.70	0.9998

Table 9: Robustness to image transformations (32-bit watermark, 1,000 images) with PGD refinement ($\varepsilon = 0.004$, 10 steps, $\alpha = 0.1$), measured at the deployment threshold θ (99th percentile of the clean unwatermarked null, held fixed); 1,000 matched unwatermarked negatives are attacked and scored alongside the watermarked images, so FPR@θ is the measured false-positive rate under each attack. *TPR at a per-attack recalibrated 1%-FPR threshold. PGD does not modify unwatermarked images, so the attacked negatives—and hence FPR@θ with its bootstrap 95% CIs (± 0.7 pp except Center/Random Crop: $[30.4, 36.8]$/$[17.5, 22.3]$)—coincide with the no-PGD values of Table 2. Attack parameters are listed in Table 7.

Method	FID $\downarrow$	KID ($\times 10^2$) $\downarrow$	CLIP Score $\uparrow$
Unwatermarked (baseline)	30.57	1.33	25.79
16-bit watermark	31.13	1.29	25.75
32-bit watermark	30.95	1.35	25.71
64-bit watermark	30.74	1.35	25.73

Table 10: Image quality metrics with PGD refinement ($\varepsilon = 0.004$, 10 steps, $\alpha = 0.1$).

with either cause. We separate the two by holding the intervention's *geometry* fixed and varying only its *content*. For every image and every area we paste the same object at the same location through the same alpha mask, and replace only the RGB values composited through that mask. Three arms form an ordered semantic ladder: **object**, the foreign object itself; **self-copy**, a horizontally displaced copy of the image's own content; and **blur**, the image's own content low-pass filtered with a Gaussian of $\sigma = 6$ px, fixed in advance. The control is a separate run over the same released images as Table 4: no image is regenerated, and only the tampering and verification are repeated, giving $45{,}000$ per-image rows across the nine cells.

Two per-image quantities separate the two causes. The hash displacement $d_{\text{hash}} = \text{HD}(H(x_w), H(x'))$ involves no watermark at all. The payload displacement $d_{\text{pay}} = \text{HD}(\hat{H}(x_w), \hat{H}(x'))$ involves no perceptual hash. Both are zero when $x' = x_w$, and Table 15 reports both for all nine cells.

These two do not add up to d. An edit moves both hashes, and some of that movement never reaches d. A bit that already disagreed can be flipped back into agreement, which lowers d while raising d_{hash}; and a bit flipped in *both* hashes at once is counted by d_{hash} and by d_{pay} while leaving d unchanged. This is systematic rather than occasional: in every row of Table 15 the mean d_{hash} exceeds the rise in d above the 2.52-bit unmodified baseline. Because Hamming distance is a metric, what holds in place of addition is the triangle bound $|d - d_{\text{unmod}}| \leq d_{\text{hash}} + d_{\text{pay}}$ against the same image's unmodified value d_{unmod}, which we confirm in all $45{,}000$ rows. The two displacements cap how far d can move; they do not decompose it.

Even an untouched image has $d_{\text{unmod}} = 2.52$ bits, for two measured reasons. Payload extraction at $K{=}64$ is imperfect (Table 1), and embedding the fingerprint of x inside x_w already moves the hash by $\text{HD}(H(x_w), H(x)) = 1.73$ bits. Together these account for the baseline, though as distances they do not sum to it.

Finally, d_{pay} is not the payload column of Table 4. That column, $\text{HD}(\hat{H}, w)$, measures the distance from the payload embedded at issuance, so it carries the clean extraction error as an offset. d_{pay} starts at zero, which is what a comparison between arms needs.

Payload	Bit Acc. (%)	Perfect Match (%)	Mean Correct
16-bit	97.69	85.02	15.63/16
32-bit	97.17	68.60	31.10/32
64-bit	96.26	44.88	61.61/64

Table 11: BiGR-L-512 watermark extraction accuracy with PGD refinement (5,000 class-conditional images; $\varepsilon = 0.004$, 10 steps).

Payload	TPR@1% (%)	AUROC
16-bit	99.92	0.9999781
32-bit	100.00	0.9999987
64-bit	100.00	0.9999997

Table 12: BiGR-L-512 watermark detection at 1%-FPR with PGD refinement ($\varepsilon = 0.004$, 10 steps).

Paired differences at matched geometry. Every arm runs on the same images with the same paste geometry, so we can compare the arms image by image rather than in aggregate. That removes between-image variation from the comparison. The lower panel of Table 15 reports these paired differences with 95% intervals. At 15% area the object arm exceeds self-copy by 5.17 $[5.01, 5.33]$ bits of d_{hash}, and self-copy exceeds blur by 5.25 $[5.13, 5.37]$. The ladder is monotone at every area, and both gaps widen with it.[1] The occlusion in these comparisons is identical by construction, so the differences cannot be attributed to damaged carriers. Were the detector responding to occlusion rather than to content, the three arms would score alike; instead they separate by more than five bits at the largest area.

Carrier damage tracks area, not content. d_{pay} behaves quite differently from d_{hash}. It grows with the pasted area, as expected when more carriers are overwritten, but at a fixed area it barely distinguishes the arms (Table 15). At 15% the blur arm damages carriers as much as the self-copy arm, to within 0.02 bits, while displacing the hash by less than half as much. One caveat: $\hat{H}(x_w)$ is read from the issuance record, produced by a separate run, so d_{pay} carries an unmeasured floor. We therefore compare it across arms at matched geometry only, and make no claim about its absolute magnitude.

Neither carrier damage nor pixel magnitude explains the gap. Two low-level quantities could in principle be driving the separation instead of meaning: how much of the watermark the paste destroyed, and how hard it changed the pixels. Neither does. Sorting the images by their measured d_{pay} and comparing arms within each group leaves the ladder intact, and the worst-damaged images separate furthest: at 15% area object leads self-copy by 5.47 bits there, against 5.17 overall. A regression controlling for pixel magnitude prices the whole object-to-self-copy difference in magnitude at 0.52 of those 5.17 bits.[2] We do not claim this rules out every low-level cue.

Scope of the control. The control has three limits. First, blur is not silent: at 15% area it flags 27.76% of images, against a clean false-alarm rate of 2.34%. Much of that falls on images that

[1] The object arm is re-run alongside the controls so that all three share identical paste geometry. It reproduces Table 4 to within 0.4 percentage points of detection ($52.98/80.56/91.42$ against $52.80/80.68/91.04$), and to within 0.01 bits on every continuous quantity. The residual is images at the $\tau = 8$ boundary under nondeterministic mixed-precision extraction. Table 4 retains the original figures.

[2] Carrier damage: at 15% area the group with $d_{\mathrm{pay}} \geq 5$ bits holds $1{,}229$ object, 834 self-copy and 844 blur images, with mean d_{hash} of 14.55, 9.08 and 3.64. Within an arm d_{hash} varies by about one bit across d_{pay} groups, and the two are essentially uncorrelated, $|r|$ never exceeding 0.05 at any arm or area. Pixel magnitude: the mean absolute per-channel difference inside the changed region at 15% area is 86.1 (object), 61.2 (self-copy) and 12.0 (blur) of 255. Regressing d_{hash} on the arm, the magnitude and the changed-pixel fraction gives arm coefficients of $+8.39$ (SE 0.18) for object and $+3.80$ (SE 0.15) for self-copy, both relative to blur, with a magnitude slope of 0.0208 bits per grey level; the object-to-self-copy gap of $86.1 - 61.2 = 24.9$ levels is therefore worth $24.9 \times 0.0208 = 0.52$ bits.

Attack	Bit Acc. (%)	TPR@θ	FPR@θ	TPR@1%*	AUROC
None	97.03	100.00	0.90	100.00	1.0000
Gaussian Noise	92.27	99.00	0.10	99.80	0.9999
Gaussian Blur	96.27	100.00	0.70	100.00	1.0000
Color Jitter	96.38	100.00	1.00	100.00	1.0000
Center Crop	82.71	100.00	0.80	100.00	1.0000
Random Crop	87.48	99.90	2.60	99.70	0.9999
Grayscale	96.45	100.00	1.10	100.00	1.0000
JPEG Compress	96.05	100.00	0.60	100.00	1.0000

Table 13: BiGR-L-512 robustness to image transformations (32-bit watermark, 1,000 images) with PGD refinement ($\varepsilon = 0.004$, 10 steps), measured at the deployment threshold θ (99th percentile of the clean unwatermarked null, held fixed); 1,000 matched unwatermarked negatives are attacked and scored alongside the watermarked images, so FPR@θ is the measured false-positive rate under each attack. *TPR at a per-attack recalibrated 1%-FPR threshold. PGD does not modify unwatermarked images, so the attacked negatives—and hence FPR@θ with its bootstrap 95% CIs (± 0.6 pp except Random Crop: $[1.6, 3.6]$)—coincide with the no-PGD values of Table 2. Attack parameters are listed in Table 7.

Method	FID $\downarrow$	KID ($\times 10^2$) $\downarrow$	CLIP Score $\uparrow$
Unwatermarked (baseline)	23.34	0.80	24.65
16-bit watermark	19.79	0.79	24.46
32-bit watermark	19.30	0.75	24.44
64-bit watermark	19.41	0.78	24.47

Table 14: BiGR-L-512 image quality metrics with PGD refinement ($\varepsilon = 0.004$, 10 steps).

Area	Arm	d_{hash}	d_{pay}	Mean d	Detection (%)
—	Unmodified	0.00	0.00	2.52	2.34†
5%	Object	7.47	1.61	8.74	52.98
	Self-copy	4.45	1.47	5.95	26.88
	Blur	1.79	1.25	3.64	6.74
10%	Object	11.37	2.25	12.75	80.56
	Self-copy	7.02	1.93	8.54	53.56
	Blur	2.91	1.74	4.81	16.00
15%	Object	14.12	2.99	15.73	91.42
	Self-copy	8.95	2.33	10.64	71.60
	Blur	3.70	2.35	5.94	27.76

Paired difference in d_{hash}	5%	10%	15%
Object − Self-copy	$+3.01$ $[2.90, 3.13]$	$+4.35$ $[4.20, 4.49]$	$+5.17$ $[5.01, 5.33]$
Self-copy − Blur	$+2.67$ $[2.58, 2.75]$	$+4.11$ $[4.01, 4.21]$	$+5.25$ $[5.13, 5.37]$

Table 15: Equal-geometry control (Infinity), 5,000 images per cell, $\tau = 8$. All three arms at a given area share the same object, size, position and alpha mask per image; only the RGB composited through that mask differs. Blur uses $\sigma = 6$ px, fixed in advance. d_{hash} is hash displacement, d_{pay} payload displacement, both floored at zero and neither an additive component of d. †For the unmodified row the percentage is the false-alarm rate. Lower panel: per-image paired differences with 95% intervals (± 1.96 standard errors of the paired difference).

already sat close to the threshold before any edit.[3] Not all of it does, and a mild, largely meaning-preserving edit that crosses τ a quarter of the time is a real cost of this operating point.

[3]Flagged blur images have mean unmodified d of 3.76, against 2.05 for unflagged ones. At 5% area, 58.5% of the flagged images come from the 12.9% of images whose unmodified d is at least 5.

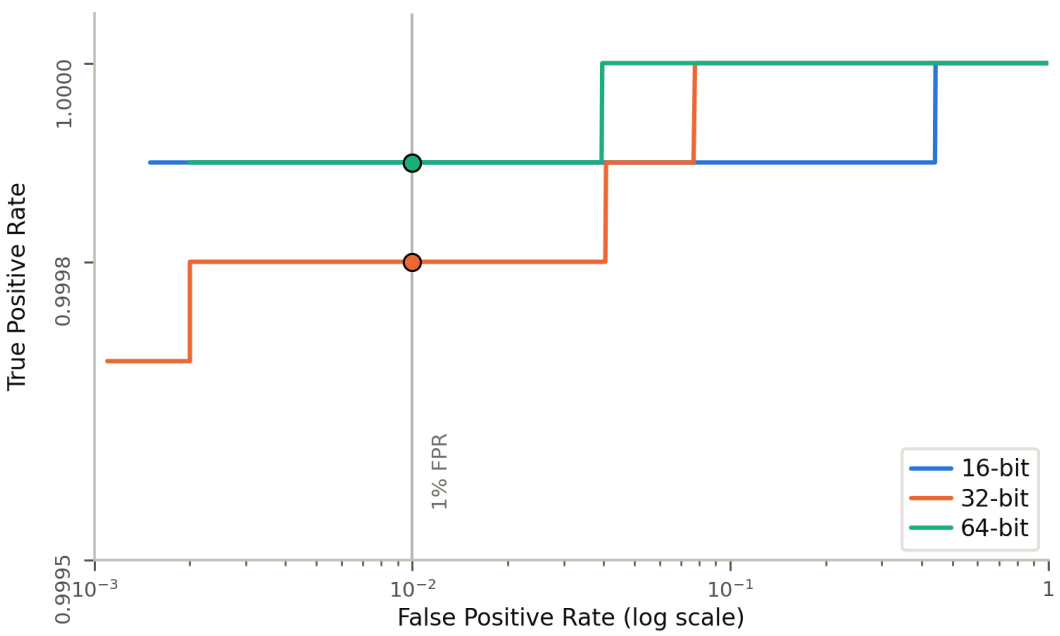

Figure 4: ROC curves for watermark detection (Infinity; 16, 32, 64-bit payloads), measured on 8-bit PNG outputs. The FPR axis is logarithmic and the TPR axis is cropped to the operating region near 1; the vertical gray line marks the 1% FPR threshold. Best viewed in color.

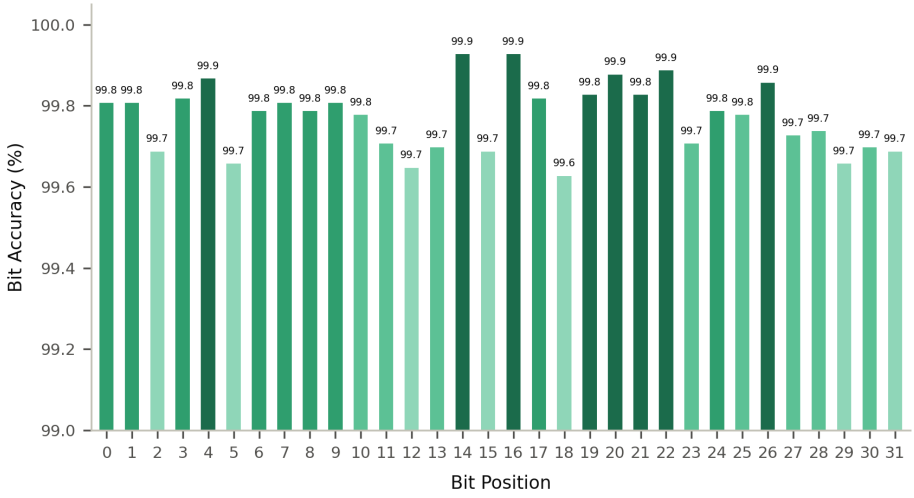

Figure 5: Per-bit accuracy across 32 watermark bit positions (Infinity; 10,000 images).

Second, d_{hash} uses the same perceptual hash as the detector. The control therefore shows that detection follows the hash rather than the damaged carriers. Whether the hash in turn tracks *meaning* is a separate claim, and it rests on the ordering of the ladder and on DINOv2's documented invariances, not on an independent measurement of semantics.

Third, the control varies only the pasted content, so it does not cover the transformations a deployed verifier would meet. Appendix I measures false alarms and detection under the common image transformations of Table 2; rescaling and brightness changes remain unevaluated.

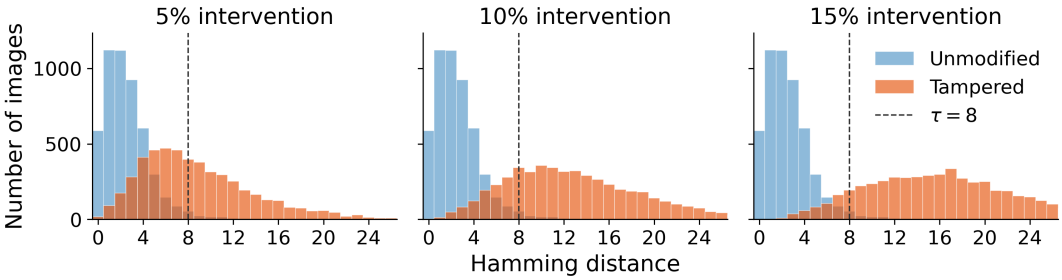

Figure 6: **Hamming-distance distributions for semantic integrity (Infinity).** Unmodified images (blue) and tampered images (orange) separate increasingly as the pasted forbidden object's visible pixels cover 5%, 10%, or 15% of the area. The dashed line marks the operating threshold $\tau = 8$. 5,000 images per condition.

	Untampered				Detection (%) by area		
Attack	**False alarm (%)**	**Mean d**	d_{hash}	**Payload HD**	**5%**	**10%**	**15%**
None	2.20	2.51	0.00	0.84	52.50	81.40	91.80
Gaussian Noise	72.80	12.63	1.04	11.36	92.70	98.10	99.20
Gaussian Blur	18.10	4.96	0.39	3.38	73.30	91.90	96.50
Color Jitter	12.60	4.09	1.10	2.13	63.70	86.80	93.90
Center Crop	99.80	18.90	5.45	16.37	100.00	100.00	100.00
Random Crop	100.00	22.41	5.74	20.44	100.00	100.00	100.00
Grayscale	61.30	9.29	5.14	4.78	91.20	96.00	99.00
JPEG Compress	48.10	8.07	0.58	6.69	84.80	94.30	97.90

Table 16: Semantic integrity (Infinity) under the transformations of Table 2 at $\tau = 8$, on the first 1,000 of the 5,000 released images (sorted by image id). Untampered: the transformation is applied to the released image with no object pasted, so every flag is a false alarm; d_{hash} is the hash displacement of Appendix H and Payload HD is $\text{HD}(\hat{H}, w)$ as in Table 4. Detection: the forbidden object is pasted at the same position and size as in Table 4, and the image is then transformed with the same realization as its untampered counterpart. Bootstrap 95% CIs are within ± 3.1 pp.

I Semantic Integrity under Common Image Transformations

Table 4 verifies the released images exactly as issued. A deployed verifier may instead receive a copy that has been re-encoded or otherwise processed without any change of meaning, and such processing can move both the recovered payload and the recomputed hash. We therefore repeat the verification under the seven transformations of the robustness evaluation (Table 2, parameters in Table 7) on the first 1,000 of the 5,000 released images, the same number as in Table 2. Each transformation is applied to two versions of every image. The first is the untampered image, so any flag is a false alarm. The second is the image after the forbidden-object paste of Table 4, with the same object, size and position, so any flag is a detection. Both versions receive the same realization of each stochastic transformation. Unlike in Table 2, the transformed image is rounded to 8 bits before verification, and $\hat{H}$ and H' are both computed from the rounded image, which is what a verifier would receive. The suite contains no rescaling or brightness change.

Without a transformation, the evaluation reproduces the per-image tamper signal d of the equal-geometry control's object arm on all 1,000 images at every area, and its rates match Table 4 to within 0.8 percentage points (2.20% false alarms on this subset, against 2.34% on all 5,000 images).

Under noise, blur, color jitter and JPEG the perceptual hash is largely stable, with d_{hash} at or below 1.10 bits, and the false alarms come mainly from the payload. The 64-bit payload uses half as many carriers per bit as the 32-bit payload of Table 2 ($M = 800$ rather than 1,600, Appendix A), and its bit accuracy is lower under every transformation, on a different image set: 82.3% under noise and 89.6% under JPEG, against 87.8% and 94.8% in Table 2. The resulting payload error of 11.36 and 6.69 bits pushes d past τ on 72.8% and 48.1% of untampered images. Blur and color jitter are milder, at 18.1% and 12.6%. Grayscale moves the hash by 5.14 bits, raises the payload error to 4.78

bits, and flags 61.3%. The two crops replace 36% of the image with white padding, which is itself a substantial content change (d_{hash} of 5.45 and 5.74 bits), and flag nearly every image.

Detection rises under every transformation, but not because the check becomes more sensitive: the transformation raises d on the untampered image as well. Under the four photometric transformations, the hash displacement caused by the pasted object is essentially unchanged (7.65–7.82 bits at 5% area, against 7.66 without a transformation). Under noise, grayscale and cropping most untampered images are flagged, and under JPEG about half, so at this operating point a flag no longer reliably indicates tampering. Under blur and color jitter the gap to detection remains, e.g. 12.6% false alarms against 63.7–93.9% detection under color jitter. Raising τ does not remove the problem: at $\tau = 12$, JPEG still flags 23.8% of untampered images, while detection at 5% area falls to 59.5%. For the photometric transformations, the limiting factor is therefore the robustness of the 64-bit payload rather than of the perceptual hash, which stronger error-correcting codes (Section 7) could address.

J Semantic-Integrity Protocol Notes

Why Infinity only. We evaluate semantic integrity on the Infinity instantiation only. The protocol is model-agnostic in principle (Section 5.1) but relies on the watermarked regeneration staying perceptually faithful to the unwatermarked original so that the embedding-induced hash drift remains small; Infinity's force-based embedding satisfies this by overwriting a subset of high-uncertainty sampled bits in a single pass at the final scale, whereas BiGR's bias-based embedding is applied per step inside the iterative masked-prediction loop and compounds across refinement steps, which would inflate that drift. We therefore leave a cross-model integrity evaluation under BiGR to future work (Section 7).

Dynamic range of the tamper signal. Two unrelated images should disagree on about half their hash bits, which for 64 bits means a distance near 32. The retained DINOHash bits are slightly biased, so the real ceiling sits a little lower: comparing the payload extracted from one issuance image against the hash of another gives 29.45 bits on average (SD 4.39).[4] At $\tau = 8$ that flags 99.94% of unrelated pairs, so entirely different content is separated almost perfectly.

The difficulty is at the other end. A clean image already sits at 2.52 bits, leaving under six bits of headroom before τ, and even the 15% intervention reaches only 15.73—about half way from the clean baseline to that unrelated-image ceiling. Everything short of wholesale replacement has to be resolved inside that band. That is why 5% tampering is caught only 52.80% of the time, and why the choice of τ matters as much as it does.

A single-pass issuance variant. Issuance as specified generates each image twice, once to obtain $H(x)$ and once to carry it. Infinity's coarse-to-fine sampling admits a cheaper variant. At the scale where the watermark is embedded, every coarser scale is already committed. One could therefore decode a preview image $\tilde{x}$ from those scales, compute $H(\tilde{x})$, and embed that fingerprint in the remaining refinement: one generation instead of two, with the drift bounded by what the remaining refinement changes rather than by an independent re-sample.

We have not evaluated this variant. The quantity that would decide it is $\mathrm{HD}(H(\tilde{x}), H(x_w))$, because the verifier recomputes the hash of the released image and not of the preview. That distance would have to fit inside what is left of the budget below $\tau = 8$ once the 1.73-bit embedding drift and the $K{=}64$ extraction error have taken their share. We note it as the natural route to halving issuance cost.

K Carrier Selection Ablations (Infinity)

The force-based Infinity embedding of Sec. A.3 makes two design choices: *which* candidate positions to overwrite (the selection criterion) and *how many* (the selection ratio $M/|\mathcal{C}_k|$). The main-line

[4] Five of the 64 retained bits have a marginal probability outside $[0.25, 0.75]$, the two most extreme being 0.003 and 0.818. The measured mean matches the $\sum_k 2p_k(1-p_k) = 29.44$ predicted from the per-bit marginals alone, over random unrelated pairs from the $5{,}000$ issuance images.

Mode	Ratio	M	Bit Acc. (%) ↑	Perfect (%) ↑	TPR@1% (%) ↑	AUROC ↑	FID ↓	CLIP ↑
uncertain*	0.39	1,600	99.83 [99.76, 99.88]	95.8 [94.6, 97.0]	100.0	1.0000	63.91	25.58
random	0.39	1,600	97.17 [96.90, 97.44]	52.1 [49.2, 55.1]	99.9 [99.7, 100.0]	0.9999	63.78	25.62
confident	0.39	1,600	84.20 [83.53, 84.84]	4.9 [3.6, 6.3]	97.9 [96.7, 98.8]	0.9962	63.59	25.69
uncertain	0.10	410	86.10 [85.59, 86.60]	3.2 [2.1, 4.4]	99.3 [98.6, 99.8]	0.9994	63.69	25.63
uncertain	0.25	1,024	98.90 [98.72, 99.07]	77.9 [75.4, 80.4]	100.0	1.0000	63.89	25.60
uncertain	0.60	2,458	99.82 [99.73, 99.89]	96.4 [95.2, 97.5]	100.0	1.0000	64.09	25.58
uncertain	0.80	3,277	96.93 [96.66, 97.20]	47.9 [44.9, 50.8]	99.9 [99.7, 100.0]	0.9999	64.26	25.63

Table 17: Infinity carrier-selection ablation (32-bit payload, 1,000 COCO-prompted images per arm, no PGD; detection vs. the same 1,000 unwatermarked negatives in every arm). Brackets are 95% bootstrap CIs (2,000 resamples); degenerate CIs are omitted. The ratio column abbreviates the baseline 0.390625 as 0.39. *The starred row is the shared baseline configuration: the top three rows ablate the *selection criterion* at the baseline ratio (Fig. 7), the bottom four the *selection ratio* under uncertainty-guided selection (Fig. 8). KID (not shown) is 1.38–1.44×10^{-2} across all arms, with within-eval subset standard deviations 0.11–0.14×10^{-2}; FID is a point estimate (no CI).

configuration overwrites the M *most uncertain* entries per payload bit—those with $p_{b,v}$ closest to 0.5—at a ratio of 0.390625 (M=1,600 of the $4{,}096$ bit-plane candidates). We ablate both choices on the 32-bit payload, with 1,000 COCO-prompted images per arm, all arms sharing the same prompts and random seed, and no PGD refinement. Detection is measured against the same 1,000 unwatermarked negatives in every arm, and we report 95% bootstrap CIs (2,000 resamples). Table 17 gives the full sweep; Figures 7 and 8 visualize the two ablations.

Selection criterion. Fixing the ratio at the baseline 0.390625, we vary the criterion: the M most uncertain positions (main line), M uniformly random positions, or the M most confident positions (Fig. 7). Uncertainty-guided selection nearly saturates recovery (99.83% bit accuracy; 95.8% of images with all 32 payload bits correct). Random selection is noticeably weaker (97.17%; 52.1% perfect), and confident selection largely fails (84.20%; 4.9% perfect). Detection degrades far less than decoding: even confident selection retains AUROC 0.9962 and 97.9% TPR@1%, so a bad criterion weakens *payload recovery* long before it endangers *detectability*. Image quality is statistically indistinguishable across criteria (FID 63.6–63.9, with KID and CLIP CIs overlapping; Table 17): uncertainty-guided selection strictly dominates, buying substantially better recovery at no measurable quality cost. This is consistent with the low-semantic-cost argument of Eq. 1: extraction votes over all $4{,}096$ candidates, and forcing bits the model considers near-ties leaves an image whose re-encoding reliably returns the forced values, whereas overwriting bits the model is confident about creates local token combinations inconsistent with the rest of the sample, which the tokenizer's decode–encode round trip does not preserve.

Selection ratio. Fixing the criterion to uncertainty-guided, we vary the selection ratio over five settings: 0.10, 0.25, 0.39, 0.60, and 0.80, corresponding to $M = 410$, $1{,}024$, $1{,}600$, $2{,}458$, and $3{,}277$ (Fig. 8). Recovery does not improve monotonically with more forcing; it saturates on an interior plateau (ratios 0.25–0.60: 98.9–99.8% bit accuracy, 77.9–96.4% perfect payloads) and degrades at both ends. At 0.10 too few carriers are forced per bit, so the extraction vote over the full candidate set is dominated by re-encoding noise (86.1%; 3.2% perfect). At 0.80 the selection extends well beyond the near-uncertain band into positions the model is increasingly confident about, reproducing the confident-selection failure (96.9%; 47.9% perfect). Image quality stays flat across the whole sweep (FID 63.69–64.26, KID 1.38–1.44×10^{-2}, CLIP 25.58–25.63): the ratio does not trace a quality–recoverability trade-off. The baseline ratio 0.390625 sits in the center of the plateau, so the operating point is robust to the exact choice of M.

Takeaway. Both choices behave as the low-semantic-cost account predicts: the watermark must be written into positions the model regards as nearly free, in numbers large enough to dominate the post-re-encoding vote—and no more.

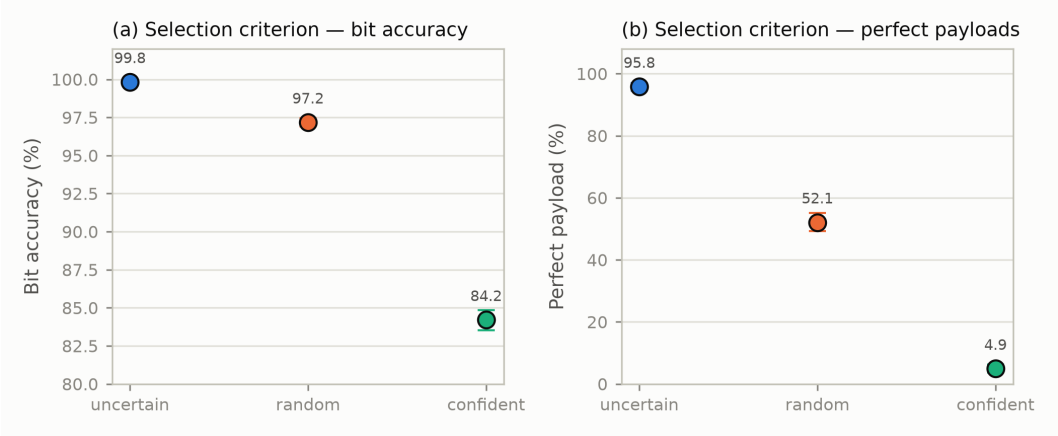

Figure 7: **Carrier-selection criterion** (Infinity, 32-bit payload, ratio 0.390625, 1,000 images per arm, no PGD). (a) Bit accuracy; (b) fraction of images with all 32 payload bits recovered. Error bars are 95% bootstrap CIs. Overwriting the least uncertain entries nearly saturates recovery; random selection loses about 3 points of bit accuracy and half of the perfect payloads; forcing the most confident entries collapses recovery—while image quality is unchanged across all three arms (Table 17).

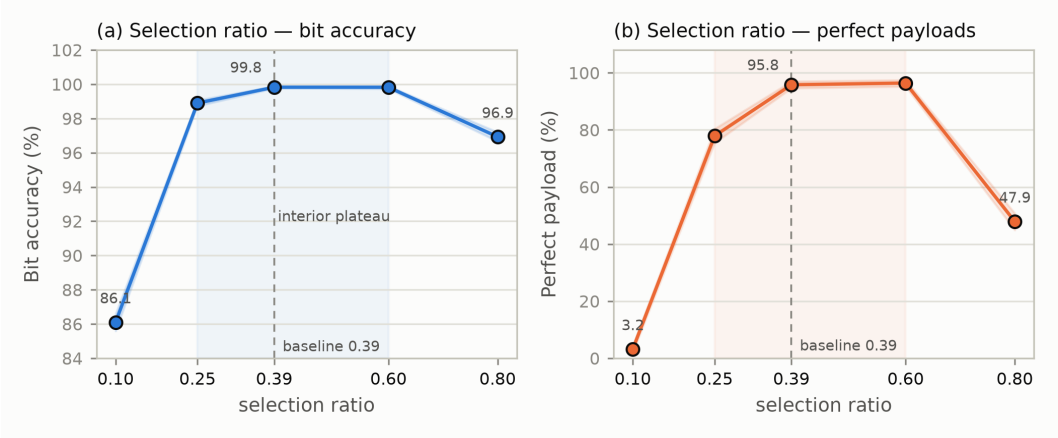

Figure 8: **Selection-ratio sweep** (Infinity, 32-bit payload, uncertainty-guided selection, 1,000 images per arm, no PGD). Recovery saturates on an interior plateau (shaded band, ratios 0.25–0.60) and degrades at both ends: at 0.10 too few forced carriers leave the extraction vote dominated by re-encoding noise, while at 0.80 forcing reaches past the near-tie band into confidently determined positions whose overwrites do not survive re-encoding. The baseline ratio 0.390625 (dashed line) sits at the center of the plateau; image quality is flat across the sweep (Table 17).

L Additional Qualitative Examples

Figure 9 and Figure 10 show additional side-by-side examples of unwatermarked (top row) and watermarked (bottom row) images for Infinity and BiGR, respectively. Each column is a matched pair generated from the same prompt and seed. The Infinity examples use a 32-bit watermark embedded at scale s_{13} with PGD refinement; the BiGR examples use a 32-bit bias-based watermark on BiGR-L-512. Visual differences remain imperceptible for Infinity, while BiGR watermarked outputs exhibit minor stylistic shifts relative to the unwatermarked baseline.



Figure 9: Infinity paired examples. **Top**: unwatermarked. **Bottom**: 32-bit watermarked with PGD refinement. Six matched pairs generated from identical prompts and seeds.



Figure 10: BiGR paired examples. **Top**: unwatermarked. **Bottom**: 32-bit watermarked. Six matched pairs generated from identical class labels and seeds (BiGR-L-512).

M Comparison with a Post-Hoc Multi-Bit Watermark (TrustMark)

Post-hoc watermarks embed a message into a finished image, so they need no access to the generator and apply to Infinity's outputs unchanged. We compare against TrustMark (Bui et al., 2025), an open-source, trained encoder–decoder watermark that embeds 100 bits per image. We watermark the 10,000 unwatermarked Infinity generations of Section 6, which are the detection null and quality baseline of Tables 1 and 3, with our 32-bit key. We score them under the same protocol: the same null, the same 1,000 attacked pairs with the same per-image attack realizations (Table 7), the same metric code, and the same 8-bit PNG artifact.

Configuration. We use the default TrustMark-Q checkpoint (release 0.9.2) at its default strength, with TrustMark's resolution scaling: the watermark residual is computed at 256×256 and upsampled to 1024×1024. The 100 embedded bits hold the payload, BCH parity bits, and a 4-bit schema tag. We use the schema with the strongest error correction, which holds up to 40 payload bits and corrects up to 8 bit errors, rather than the default one (61 payload bits, 5 errors). Our 32-bit key is zero-padded to 40 bits.

Accuracy and detection. Bit accuracy is measured on the 32 key bits of TrustMark's error-corrected payload, or on the uncorrected decoder bits when error correction fails. TrustMark's own decision is that error correction succeeds and returns the key. That decision has no threshold, so to report TPR at a fixed FPR we score the decoder output with the known-payload statistic of Eq. 2,

$$S_{\mathrm{TM}} = \sum_{j=1}^{100} (2c_j - 1)\big(\sigma(\ell_j) - 0.5\big), \tag{10}$$

	Accuracy and detection				Quality		
Method	**Bit Acc.**	**Perfect**	**TPR@1%**	**AUROC**	**FID** ↓	**KID** ($\times 10^2$) ↓	**CLIP** ↑
Unwatermarked	—	—	—	—	30.57	1.33	25.79
Ours (32-bit)	99.77	95.18	99.98	0.999988	30.95	1.34	25.71
TrustMark-Q	100.00	99.93	100.00	1.000000	30.29	1.32	25.76

Table 18: Ours vs. TrustMark-Q without attack (32-bit payload, the same 10,000 Infinity images at 1024×1024). Bit Acc., Perfect, and TPR@1% are in %; detection is against the same unwatermarked-generation null. TrustMark's own decision reaches 99.70% TPR at 0.00% FPR. Recomputing the unwatermarked FID and CLIP Score in the TrustMark pipeline reproduces the values above.

	Ours (Infinity)				TrustMark-Q			
Attack	**Bit Acc.**	**TPR@θ**	**FPR@θ**	**TPR@1%***	**Bit Acc.**	**TPR@θ**	**FPR@θ**	**TPR@1%***
None	99.73	100.00	0.60	100.00	99.99	100.00	1.00	100.00
Gaussian Noise	87.81	94.00	0.40	94.90	99.99	100.00	1.00	100.00
Gaussian Blur	97.52	100.00	1.20	100.00	99.99	100.00	1.00	100.00
Color Jitter	98.96	100.00	1.50	100.00	99.74	100.00	1.20	100.00
Center Crop	83.35	99.80	33.70	98.40	99.99	100.00	0.70	100.00
Random Crop	68.87	96.10	19.90	61.90	99.99	100.00	0.70	100.00
Grayscale	97.82	100.00	1.10	100.00	99.82	100.00	0.90	100.00
JPEG Compress	94.83	99.80	0.80	99.80	99.99	100.00	1.10	100.00

Table 19: Robustness of ours vs. TrustMark-Q (32-bit payload): the same 1,000 watermarked images and the same 1,000 unwatermarked negatives, attacked with identical per-image realizations (Table 7). Columns as in Table 2, whose Infinity columns are repeated here. For TrustMark-Q, AUROC is 1.0000 under every attack, the bootstrap 95% CIs for FPR@θ are within ±0.8 pp, and its own decision keeps 96.8–99.9% TPR at 0.0% FPR.

where c is the 100-bit codeword TrustMark embeds for our key and ℓ_j are the decoder logits. We threshold S_{TM} at the 99th percentile of the same unwatermarked null (Section 4.4).

Results. TrustMark-Q matches our detection and image quality (Table 18) and has higher clean bit accuracy: 99.93% vs. 95.18% perfect match, and 98.06% even before its error correction. Under attack (Table 19) it keeps at least 99.7% bit accuracy and 100% TPR@θ. Its null also stays calibrated under cropping (FPR@$\theta \leq 1.2\%$), where ours shifts to 33.7% and 19.9%. The gap reflects the design. TrustMark is trained against a noise model that covers five of our seven attacks (noise, blur, color jitter, grayscale and JPEG) as well as crop-and-resize, whereas our scheme is training-free and its position-indexed carriers do not survive a shifted token grid. The two methods also rest on different deployment assumptions. TrustMark is a post-processing step whose decoding is public by design ("not conditional on any secret key" (Bui et al., 2025)), and it ships with a watermark-removal network. Our watermark is embedded by the generator's own sampler, needs no training, and is keyed by a secret carrier assignment (Appendix C).

N Zero-Bit Adaptation and Comparison with BitMark

Our method is multi-bit: it embeds and decodes a K-bit payload (Sec. A). To the best of our knowledge, no multi-bit watermarking scheme that operates on Infinity's binary tokens exists, so no multi-bit baseline is available on this generator. The closest comparable prior work on a bitwise autoregressive image generator is BitMark (Kerner et al., 2025), which is *zero-bit*: it carries no payload and tests only whether an image was produced by a watermarked model. We therefore restrict the comparison to the zero-bit setting that both methods support: we minimally adapt our scheme to zero-bit and compare detection and robustness head-to-head. The adaptation changes only the payload pattern and the detection statistic; the Infinity embedding hook, the carrier space, the VAE extraction, the PGD refinement, and the evaluation protocol are unchanged from Sec. A and Sec. E.

N.1 THE BITMARK REFERENCE

BitMark watermarks the same public Infinity-2B checkpoint with the 32-bit binary tokenizer (V=32) that we use, generates at 1024×1024 from COCO 2014 prompts with classifier-free guidance scale 4, and reports detection on 10,000 images and robustness on 1,000. Its watermark is a green/red list over n=2-bit contexts (green list $\{01, 10\}$, red list $\{00, 11\}$) with bias strength δ=2, applied across all 13 scales; detection uses a z-score statistic on the count of green-list contexts, with robustness reported at 1%-FPR. BitMark carries *no payload*; our $K \in \{16, 32, 64\}$ is payload capacity, an orthogonal axis. The comparison below therefore isolates detection and robustness, holding the generator fixed.

N.2 ZERO-BIT ADAPTATION

The adaptation turns two knobs and leaves everything else intact.

Payload. We replace the random K-bit payload $\mathbf{w} \sim \{0,1\}^K$ with the fixed all-ones vector $\mathbf{w} = \mathbf{1}_K$. The force-based hook of Algorithm 3 is unchanged in mechanism—it still overwrites the M most-uncertain carriers per group—but now sets every selected carrier to 1 instead of to w_k:

$$Z_{b,v} \leftarrow 1 \quad \text{for all } (b, v) \in \mathcal{S}_k. \tag{11}$$

Every forced carrier pushes the latent in the same direction, so the watermark key reduces to the carrier-selection rule and the group geometry (Sec. A.2); there is no secret bit string to communicate.

Detection score. With w_k=1 for all k, the signed detection score (Eq. 2) collapses to a one-sided surplus of ones,

$$S_{0\text{bit}} = \sum_{k=0}^{K-1} (v_k - 0.5), \tag{12}$$

where v_k is the mean soft vote over carrier group k (Sec. A.4). Under the unwatermarked null each $(v_k - 0.5)$ is approximately zero-mean; under the watermark it is positive because carriers were forced to 1. We threshold at the 99th percentile of the unwatermarked-generation null (1%-FPR), exactly as in Section 4.4. We use this *soft* sum rather than a hard majority over groups: the continuous statistic preserves a confidence gradient, enabling TPR@FPR at any operating point, and it reuses the multi-bit calibration and bootstrap machinery unchanged.

Accuracy, reinterpreted. We retain the bit-accuracy and perfect-match measurements of Table 1 for protocol continuity, but in the zero-bit setting they no longer measure payload decoding (there is no payload). "Bit accuracy" is the fraction of the K carrier groups whose mean vote exceeds 0.5; "perfect match" is the event that all K groups exceed 0.5. They are group-recovery proxies that track how strongly the forced bias survives re-encoding.

Group geometry. We use the `--watermark-bits` group structure with K=32 carrier groups. In zero-bit mode K carries no message—it is only a partition of the carrier space. The detection score $S_{0\text{bit}}$ (Eq. 12) is, up to a positive scale, the total fraction of 1s across all carriers, which is independent of the grouping; consequently *detection* (TPR@1%-FPR, AUROC, and the per-attack robustness of Table 22) is the same for $K \in \{16, 32, 64\}$. We report K=32 for concreteness and do not run 16/64. The bit-accuracy proxy in Table 21 is K-dependent by construction (the fraction of K groups voting >0.5) and is a recovery proxy, not the zero-bit detection metric. The group partition matters only for the embedding hook, which forces the M most-uncertain carriers in each group to 1.

Generation setting. For this comparison we adopt BitMark's reported generation configuration: classifier-free guidance scale 4, 1024×1024, COCO 2014 prompts, and the same Infinity-2B / 32-bit-tokenizer checkpoint used in Section 6. This holds the generator fixed so that detection and robustness differences are attributable to the watermarking scheme rather than to generation parameters.

Attack	Parameters
None	—
Gaussian Noise	$\sigma = 0.1$ on $[0, 1]$ image scale ($\equiv 0.2$ on our $[-1, 1]$ tensors)
Gaussian Blur	kernel size $= 7$, $\sigma = 3.0$ (fixed)†
Color Jitter	hue $= 0.3$, saturation $= 3.0$, contrast $= 3.0$, brightness $= 3.0$
Center Crop	crop keeps 70% of each side (centered), resized back to original size
Random Crop	crop keeps 70% of each side (random offset), resized back to original size
Rotation	random angle in $[-180°, 180°]$, black-filled corners
Grayscale	—
JPEG Compress	quality $= 75$
Horizontal Flip	—
Vertical Flip	—

Table 20: Parameters of the image transformations used in the zero-bit robustness evaluation (Table 22). We match BitMark's *released implementation* (Kerner et al., 2025) rather than its text alone, because the code fixes several values the paper leaves unstated (color-jitter brightness 3.0, resize-back crop, black rotation fill) and applies a $\pm 180°$ rotation range where the text reports $[0°, 180°]$; we follow the code in each case. The one deliberate deviation is the blur standard deviation.† The suite is stronger than our standard suite (Table 7)—larger blur kernel, color jitter roughly an order of magnitude stronger, a more aggressive crop (70% vs 80%), plus rotation and flips. The identical transformation is applied to the watermarked images and to the matched unwatermarked negatives. †BitMark's code leaves `torchvision`'s default in place, redrawing $\sigma \sim \mathcal{U}(0.1, 2.0)$ independently per image; we instead fix σ=3.0. Because a 7-tap kernel truncates the Gaussian at ± 3 px, both settings are clipped toward the same box-filter limit (σ_{eff}=2.00 px): ours yields σ_{eff}=1.84 px, against 1.64 px for BitMark's strongest draw and 0.95 px for its mean draw. Our blur attack is therefore deterministic and modestly stronger than theirs ($1.12\times$ their maximum, $1.93\times$ their mean), so the blur row of Table 22 is conservative with respect to our method.

N.3 Attack Protocol

For a fair robustness comparison we use the attack parameters reported in BitMark's Table 2 rather than our milder standard suite (Table 7); the full list is given in Table 20. We additionally report a grayscale attack and a deterministic center-crop attack. The regeneration attacks that BitMark reports—SD2.1-VAE re-encoding and CtrlRegen+ reconstruction—require auxiliary diffusion models and are deferred; placeholder rows are reserved in the robustness table.

The evaluation protocol follows BitMark's: the $1{,}000$ watermarked images are each attacked and re-encoded, and detection is scored at the unwatermarked-generation null's 99th-percentile threshold (the 1%-FPR operating point), giving TPR@1%-FPR on the attacked positives—the same definition BitMark reports. We do not attack the null images here (the attacked-negative diagnostics of Table 2 are omitted in this comparison), so per-attack FPR@θ, recalibrated TPR@$1\%^\star$, AUROC under attack, and bootstrap CIs are not reported.

N.4 Results

Table 21 reports zero-bit recovery accuracy and detection, and Table 22 reports robustness under the BitMark-aligned attack suite, with BitMark's reported TPR@1%-FPR alongside our own. Image quality (FID, KID, CLIP) is essentially unchanged from the multi-bit setting because the zero-bit adaptation leaves the watermarking mechanism unchanged (only the payload pattern and the detection statistic differ; Sec. N.2), so the watermark carries the same perceptual footprint. These metrics are reported in Table 3, so we omit a separate table here.

Method	Bit Acc. (%)	Perfect (%)	Mean Correct	TPR@1% (%)	AUROC
Ours	99.71	93.8	31.91/32	100.00	1.0000
BitMark	—	—	—	99.90	—

Table 21: Zero-bit group-recovery accuracy and detection (1,000 COCO-prompted images, 1024×1024, guidance scale 4). "Bit Acc." and "Perfect" are the K-dependent recovery proxies of Sec. N.2 (fraction of the K carrier groups, and all K groups, with mean vote >0.5); "Mean Correct" is the mean number of groups recovered over K. Detection (TPR@1%, AUROC) is the K-invariant metric of Sec. N.2, reported against the unwatermarked-generation null at the 1%-FPR operating point, as in Section 4.4; the real-photograph null gives the same saturated values and is omitted. BitMark's detection value is transcribed from Table 2 of Kerner et al. (Kerner et al., 2025) (Infinity-2B, $\delta=2$), and BitMark reports detection on 10,000 images, so the detection sample sizes differ. PGD refinement is a no-op in this configuration—the perturbation ($\|\delta\|_\infty \approx 0.15/255$) is below the $1/255$ 8-bit quantization step and recovery is already saturated—so pre- and post-PGD coincide.

Attack	Bit Acc. (%)	TPR@1% (Ours)	TPR@1% (BitMark)
None	99.71	100.0	100.0
Gaussian Noise	61.18	41.0	99.6
Gaussian Blur	83.53	98.9	99.9
Color Jitter	50.82	25.7	99.8
Random Crop	48.12	2.6	98.8
Rotation	49.46	8.5	20.1
JPEG	94.85	100.0	100.0
Horizontal Flip	47.51	1.5	100.0
Vertical Flip	46.22	0.6	78.8
Center Crop†	48.66	3.6	—
Grayscale†	97.43	100.0	—
SD2.1-VAE+	—	—	100.0
CtrlRegen+	—	—	91.6

Table 22: Zero-bit robustness vs. BitMark (1,000 watermarked images, attack parameters in Table 20). The detection statistic is K-invariant (Sec. N.2), so these numbers are those of the zero-bit scheme, not specific to any particular K. **Ours**: bit accuracy (group-recovery proxy) and TPR at the 1%-FPR threshold of the clean unwatermarked-generation null, i.e. TPR@1%-FPR on the attacked positives—the same definition BitMark uses. **BitMark**: TPR@1%-FPR transcribed from Table 2 of Kerner et al. (Kerner et al., 2025) (Infinity-2B, $\delta=2$). PGD is a no-op in this configuration (Table 21), so pre- and post-PGD robustness coincide and a single table is reported. †Center Crop and Grayscale are additional attacks we report; BitMark does not list them. SD2.1-VAE and CtrlRegen+ are deferred for our method; BitMark values are shown for reference.