BYTEMARK: MULTI-BIT WATERMARK FOR BIT-WISE IMAGE GENERATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Watermarking is a central tool for provenance tracing of AI-generated content, yet existing methods for bit-wise image generators are zero-bit: they can only distinguish between watermarked or clean (non-watermarked) data. Many real-world deployments demand more, such as tracing a generated image back to the specific model, user, or API key that produced it. We introduce BYTEMARK, the first multi-bit watermarking scheme for bit-wise image generators. Since bit-wise generators require an equal number of zeros and ones to achieve high fidelity, BYTEMARK encodes each message bit in the transitions within tokens, by inducing the next generated bit to be either *repeated* or *flipped* based on the value of the message bit. Because the rule is relative to the preceding bit rather than to a fixed value, it embeds the message without disrupting the bit distribution. We further extend this mechanism to payloads with more than a thousand bits by embedding different parts of a message into many partitions of a given image. We show that BYTEMARK preserves the visual fidelity of generated images, is highly robust to a wide range of watermark removal attacks, and that it exhibits radioactivity, i.e., new models trained on BYTEMARK-ed images inherit and produce the watermark. These properties enable BYTEMARK to provide fine-grained provenance tracing in bit-wise image generation.

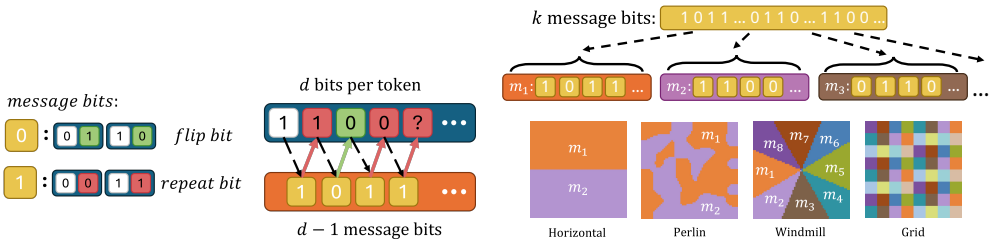
1 INTRODUCTION

Generative models now produce images that are nearly indistinguishable from human-created content (Rombach et al., 2022; Han et al., 2025). The same capability enables large-scale fraud, non-consensual imagery, and deepfakes (Chesney & Citron, 2019), and the reliable identification of synthetic media has become a regulatory priority (European Data Protection Supervisor, 2025; California Legislative Information, 2024). Passive forensics is no longer effective since the manipulations produced by current generators are so convincing that neither human inspectors, nor state-of-the-art forensic detectors, nor even the generating model itself can reliably recognize them (Wu et al., 2026).

The focus has therefore shifted to *active* provenance, where providers mark their outputs at generation time. The EU AI Act obliges providers of generative systems to mark their outputs in a machine-readable format (European Data Protection Supervisor, 2025). Comparable requirements have been enacted or proposed in other jurisdictions (Biden, 2026; Cyberspace Administration of China et al., 2025; California Legislative Information, 2024) and industry has converged on the C2PA content-credentials standard, a cryptographically signed record of an asset's origin and edit history (Coalition for Content Provenance and Authenticity (C2PA), 2024). C2PA, however, carries this record as metadata, which can be stripped intentionally by adversaries and unintentionally by re-encoding, screenshots, and social-media compression. Thus, the provenance signal has to be paired with a mark that persists in the image itself. One type of marks that has been shown particularly robust is in-generation watermarks (Wen et al., 2023; Kerner et al., 2025), which embed a signal during image generation.

Multiple generation-time image watermarks target latent diffusion models and embed the mark in the continuous initial noise (Wen et al., 2023; Kociszewski et al., 2026). Other marks target autoregressive image generators that instead emit discrete token sequences. Watermarks for autoregressive image generators usually rely on red-green list schemes, adapted from the language domain (Kirchenbauer et al., 2023), that divide the vocabulary in red and green tokens and bias generation towards the green

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.



(a) Single token ($\leq d-1$ message bits). **(b)** Image partitioning ($> d-1$ message bits).

Figure 1: **Overview of our BYTEMARK. (a) Single token.** Each message bit determines how the next generated bit relates to its predecessor. A message bit of one nudges the model to *repeat* the preceding bit, while a message bit of zero induces it to *flip*. The rule is applied at every position of every token, thus each token carries the full message. Since the next bit is relative to the preceding bit, BYTEMARK preserves the balance of zeros and ones that bit-wise generators require for high-quality image generation. **(b) Image partitioning.** To go beyond the $d-1$ message bits that fit into a single token of dimension d, we embed different segments of the message into different partitions of the image. Horizontal and Perlin partitions double the length of the message, Windmill enables $8\times$ expansion, and Grid yields a $64\times$ higher capacity.

ones. This approach has been shown successful for image autoregressive models that emit tokens (Jovanović et al., 2025) and those that operate on bit-streams (Kerner et al., 2025). However, all these schemes share the same limitation, namely they remain strictly zero-bit, *i.e.,* they can *detect watermarked content* but do not carry any information.

This is a severe limitation because responding to abuse and enforcing licensing terms requires knowing *which* model and *which* customer or API key produced an image, and thus a payload large enough to identify the source. Unfortunately, existing generation-time zero-bit watermarks can not embed, nor recover such information. We close this gap with BYTEMARK, the first multi-bit watermark for bit-wise image autoregressive models.

Our key observation is that a message can be encoded not in the *values* of the generated bits but in the *transitions* between them. At every position of every token, BYTEMARK influences the generation process based on the current message bit: a message bit of one nudges the next generated bit to *repeat* its predecessor, while a message bit of zero induces it to be *flipped*, see Figure 1 (a). Because the bit transitions are relative to the preceding bit rather than toward a fixed value, this approach does not systematically favor zeros or ones, so the balance of zeros and ones that bit-wise generators rely on is preserved. This is crucial for the quality of the generated content. When the bit-message is shorter than the token dimensionality d, every token carries the full message. Longer payloads are split across spatial partitions of the image, with different regions carrying different message segments, presented in Figure 1 (b), which extends the capacity to more than a thousand bits. With this embedding, BYTEMARK is robust to standard value-metric and geometric augmentations as well as to targeted watermark removal attacks and survives the training of new models on the watermarked outputs, thus exhibiting the property referred to as *radioactivity* (Sablayrolles et al., 2020; Meintz et al., 2025). This enables model owners to maintain provenance even across models trained on their generated data.

Our contributions are summarized as follows:

1. We propose BYTEMARK, the first multi-bit watermarking scheme for bit-wise image generative models, which embeds multi-bit messages in the generation process at the level of bits.
2. We show that BYTEMARK is highly robust against 10 watermark removal attacks, covering value-metric, geometric, and targeted removal attacks.
3. We demonstrate that BYTEMARK is highly *radioactive*, which enables us to trace a derived model back to the exact source model and user whose outputs it was trained on.
4. We perform an extensive empirical evaluation on 4 bit-wise image generative models against 9 baselines, and 10 watermark removal attacks, spanning payloads from zero to more than a thousand bits.

2 Background and Related Work

Image Generation. Image generation has seen two dominant generative model paradigms, namely *diffusion models* (DMs) (Rombach et al., 2022; Labs et al., 2025), where samples are generated through iterative denoising starting from pure random noise until a final image is produced, and *image autoregressive models* (IARs), which treat an image as a sequence of discrete tokens drawn from a learned codebook (Esser et al., 2021; Yu et al., 2025). Visual autoregressive modeling (VAR) (Tian et al., 2024) replaces raster-order token prediction with a coarse-to-fine formulation. An image is represented as a sequence of residual token maps at increasing resolutions, and the model predicts each scale conditioned on all coarser ones. The Infinity model (Han et al., 2025) further improved on the discrete codebook by using bit-wise prediction with an implicit codebook that uses binary spherical quantization (Zhao et al., 2025) to select entries from the codebook. This scales the effective vocabulary to near infinity (e.g., 2^{32} and 2^{56}) yielding extremely high quality images. We focus on designing a novel multi-bit watermark for the state-of-the-art bit-wise image generation.

Image Watermarking. A watermark embeds a signal into generated content such that a party holding a secret key can later detect it. Every scheme negotiates a trade-off between three quantities: *quality*, the perceptual cost of embedding, *robustness*, the fraction of the signal surviving benign or adversarial transformation, and for multi-bit schemes, *capacity*, the number of carried message bits. Watermarks can be embedded at different steps in the generation process. We distinguish between *post-hoc* schemes (Zhang et al., 2019; Bui et al., 2025; Souček et al., 2025; Tancik et al., 2020), which apply the watermark on existing images and that are optimized to be imperceptible, and *in-generation* schemes (Fernandez et al., 2023; Wen et al., 2023; Kerner et al., 2025; Kociszewski et al., 2026; Jovanović et al., 2025), which influence the generation process to embed a signal into the generated content.

Current *post-hoc watermarking schemes* are generally based on a trained encoder (E) and a decoder (D), where the encoder embeds the watermark message m_w and the decoder retrieves it. The core differences in the post-hoc schemes arise in the training process of the encoder-decoder pair. TrustMark (Bui et al., 2025) targets imperceptibility by leveraging a focal frequency loss that guides the watermark into the low-frequency regions. StegaStamp (Tancik et al., 2020) uses a strong perturbation pipeline during training, enabling high robustness, which is traded off with imperceptibility. WAM (Sander et al., 2025) embeds shorter messages and recovers location masks to extract multiple messages per image cleanly. InvisMark (Xu et al., 2025), VideoSeal (Fernandez et al., 2024), PixelSeal (Souček et al., 2025) and ChunkySeal (Petrov et al., 2025) each pushed robustness, capacity and imperceptibility of trained embed-decode architectures to a new level. We provide a more detailed description of each method in Appx. A.

Generation-time watermarking schemes instead alter the sampling process itself. The standard approaches to watermarking DMs influence the initial generation noise. Tree-Ring (Wen et al., 2023) adds a ring pattern in the Fourier domain of the initial latent, whereas SERUM (Kociszewski et al., 2026) adds a fixed noise and detects it with a trained detector model. Alternatively, StableSignature (Fernandez et al., 2023) fine-tunes the decoder so that its outputs carry a mark. As IARs rely on the generation of tokens, their watermark techniques mostly follow common practices in large language models (LLMs) and embed the watermark information in the generated tokens. Specifically, they leverage KGW (Kirchenbauer et al., 2023), which separates the vocabulary into a green list G and a red list R based on a hash of the preceding token(s). During generation, the model is then nudged to produce tokens from G by adding a small bias δ to the logits of tokens in G. For detection, a hypothesis test evaluates if the number of observed green tokens is higher than the expected number. WMAR (Jovanović et al., 2025) observed that a direct application of KGW to IARs is not possible, as the process of encoding an image to tokens introduces an error and only a fraction of the original tokens from the generation-time is reconstructed. Therefore, they fine-tune the encoder and decoder of IARs to better reconstruct the generated token stream. BitMark (Kerner et al., 2025) is a KGW-based scheme that is designed specifically for bit-wise generative models, which produce images out of tokens based on bits. It separates the bit vocabulary into a green and red list with fixed length n. However, these schemes are *zero-bit*: they only provide knowledge of whether a watermark is present or not. Our ByteMark removes this limitation and enables the embedding of a multi-bit message into state-of-the-art IARs based on bit-wise generation.

Watermark Radioactivity. A watermark is *radioactive* if a model trained on watermarked data produces outputs carrying a detectable trace of the original watermark (Sablayrolles et al., 2020). The radioactive watermark persists through a training run of a model and turns into evidence of what data the model was trained on. Sander et al. (2024) evaluate this phenomenon in LLMs and show that a model trained on watermarked text emits detectable watermarked outputs even when such text only makes up a small fraction of its training data. Meintz et al. (2025) extended the analysis of radioactivity to image generative models and found that the existing watermarks lose their signal after fine-tuning. Recently, BITMARK (Kerner et al., 2025) was proposed as the first radioactive watermark for bit-wise image generative models that showed high detectability in downstream models across different architectures. Subsequently, SERUM (Kociszewski et al., 2026) achieved radioactivity for DMs by embedding a fixed noise pattern and training its detector under the same augmentations that appear in typical training pipelines. All of these marks are zero-bit, so a radioactive trace reveals only *that* watermarked data was used to train a given model but does not allow us to trace back to the precise source of the radioactive data. To bridge this gap, we introduce BYTEMARK, the first multi-bit radioactive watermark. The same message lets a provider recognize its own outputs, down to the individual model, when filtering synthetic data from future training corpora to avoid model collapse (Shumailov et al., 2024).

3 OUR BYTEMARK METHOD

BYTEMARK embeds a multi-bit message into bit-wise generated images. We start with the embedding of the watermark within a single token, followed by its detection and message decoding, and then present how to expand the payload to more than a thousand bits. Each message bit influences a bit value at a fixed bit position within every token. The message bit of one (1) biases the next bit to repeat the previous one, while the message bit of zero (0) induces a flip of the next bit value in comparison to the previous one. We provide an overview of the BYTEMARK embedding in Figure 1.

3.1 MESSAGE EMBEDDING

Overview. A bit-wise generator operates on tokens which are a sequence of d bits $b_0, \ldots, b_{d-1}$. BYTEMARK embeds one message bit into each of the $d-1$ transitions (b_{j-1}, b_j). At each position j we check the message bit m_j. When $m_j = 1$ we nudge the model to *repeat* the previous bit: $b_j = b_{j-1}$. If $m_j = 0$ we nudge it to *flip* the bit: $b_j = \neg b_{j-1}$. These operations correspond to the green and red lists from the BITMARK watermark, namely $G^{(0)} := \{01, 10\}$ and $G^{(1)} := \{00, 11\}$, read as pairs $(b_{j-1}\, b_j)$. The first bit b_0 is sampled unchanged and only serves as context. The same message is embedded in *every* token of the image, which yields redundancy that is exploited during detection for higher robustness of the watermark against removal attacks. The transition rules leave the ratio of ones to zeros balanced, which is necessary for preserving image quality as we explore in Appx. C.2. To embed the message m, we nudge the prediction of the model by adding a small bias δ to the logit ℓ^{b_j} of the bit value that completes a pair in $G^{(m_j)}$. For example, if $m_j = 1$ and the previous bit value $b_{j-1} = 0$, then we add δ to the logit for bit 0, namely ℓ^0. We detail the transition rules in Appx. C.1. The rules hold for each position $j \in [1, d-1]$ in *every* token of the generated output. We obtain the following next bit prediction probabilities p_j for our BYTEMARK watermark:

$$p_j(b_j \mid b_{j-1}, m_j) = \begin{cases} \dfrac{\exp(\ell_j^{b_j} + \delta)}{\exp(\ell_j^{b_j} + \delta) + \exp(\ell_j^{\neg b_j})} & \text{if } (b_{j-1}\, b_j) \in G^{(m_j)}, \\ \dfrac{\exp(\ell_j^{b_j})}{\exp(\ell_j^{b_j}) + \exp(\ell_j^{\neg b_j} + \delta)} & \text{if } (b_{j-1}\, b_j) \in G^{(\neg m_j)}. \end{cases} \tag{1}$$

Why not embed the message directly? The simplest alternative is a *naive* embedding that biases each bit position j directly toward the value m_j, without looking at the previous bit. It fails for two reasons: quality and robustness. Bit-wise generators and their tokenizers rely on a balanced number of zeros and ones (Kerner et al., 2025). A message with unequal counts shifts this ratio and visibly degrades the image quality (see Figure 3), and even a balanced message pulls every token toward one codeword, which degrades the quality in the same way as an unbalanced ratio of zero and one bits.

Additionally, the tokenizers also restore the balanced bit distribution when an image is re-encoded. Thus, even a bias that survives generation is removed at detection time (Appx. C.2). The above design of BYTEMARK fixes both problems. By nudging the bit *transitions*, we keep the ones-to-zeros ratio fixed and favor no single message, since each message bit is represented by a bit pair. This results in BYTEMARK surviving the re-encoding far better than the naive baseline (see more in Appx. C).

3.2 DETECTION AND DECODING

Detection and decoding are separate stages. The detector decides whether a watermark is present and controls the false positive rate. The decoder recovers the message m only for images the detector has flagged. Both stages operate on the same bit-stream, obtained by encoding the image. Following Kerner et al. (2025), we assume that the **model owner** keeps the generative model behind a secure API and is the only one with access to the encoder.

Detection. BYTEMARK embeds a message bit m_j by nudging towards repeating or flipping the preceding bit at position j. Detecting the watermark therefore tests whether the transitions at each position are biased towards one direction. Let $b_{i,j}$ denote bit j in token i and N' the number of scored tokens. For each position $j \in \{1, \dots, d-1\}$ within a token, and across all tokens $i \in \{0, \dots, N'-1\}$, we count how often the bit repeats its predecessor:

$$a_j = \sum_{i=0}^{N'-1} \mathbb{1}[b_{i,j} = b_{i,j-1}], \qquad z_j = \frac{2a_j - N'}{\sqrt{N'}}. \tag{2}$$

A repetition indicates a pair in $G^{(1)}$ and an alternation a pair in $G^{(0)}$, sso a_j counts the bits voting for $m_j = 1$ and $\mathbb{E}[z_j]$ carries the sign of $2m_j - 1$. Under the idealised model, in which the transitions of an un-watermarked image are independent fair coins, each z_j is drawn from a standard normal distribution. If the image is watermarked, then the bias δ increases the magnitude of the z-score. We distinguish between two detection strategies. The *blind* detector has no information about m and aggregates the magnitudes:

$$z_{\text{blind}} = \frac{1}{\sqrt{d-1}} \sum_{j=1}^{d-1} |z_j|, \tag{3}$$

which, under the null of independent fair-coin transitions, has mean $\sqrt{2(d-1)/\pi}$. The *oracle* detector scores against a known target message m:

$$z_{\text{oracle}} = \frac{1}{\sqrt{d-1}} \sum_{j=1}^{d-1} (2m_j - 1)\, z_j, \tag{4}$$

which is standard normal under the same null. The idealized model does not hold exactly, since the transitions of real images are neither independent nor unbiased. We therefore calibrate the decision threshold on held-out un-watermarked generated images and analyze the null distribution in detail in Appx. D.

Decoding. When an image passes the detection threshold, we can extract the embedded watermark message. The sign of each z_j is exactly the estimate of the corresponding message bit:

$$\hat{m}_j = \mathbb{1}[z_j > 0], \qquad j = 1, \dots, d-1. \tag{5}$$

The magnitude $|z_j|$, which is computed across all tokens, is the confidence of the estimate for the bit value at that position. Decoding therefore does not require any additional pass over the image, and z_j can be used directly to obtain the message.

3.3 PAYLOAD EXPANSION BEYOND A SINGLE TOKEN

In the default construction of BYTEMARK, we represent a message in one token, which enables the message length of $d-1$ bits for the token of d dimensions (e.g., 31-bit message for Infinity-2B model with 32-bit token size, 55-bit message for Infinity-8B model with 56-bit token size (Han et al., 2025), or 63-bit message for Instella-T2I (Wang et al., 2025) with 64-bit token size). To go beyond this limit, we split the image into R regions and let each region carry a different part of the message. A

message of $k = R\,(d-1)$ bits is then spread across the image, while every token still carries only $d-1$ bits and the balanced bit statistics of Sec. 3.1 are untouched.

Payload Layout. Let $\mathcal{R}_1, \ldots, \mathcal{R}_R$ denote the partition of the token grid. Region $\mathcal{R}_i$ carries the i-th part $m_{\mathcal{R}_i}$ of the message, embedded into each of its tokens exactly as in Eq. (1). Detection and decoding run per region with the statistics of Sec. 3.2, and the R decoded chunks are concatenated. The partition of the token grid is fixed at generation time and message independent. The detector applies the same partition to the token grid, decodes each region independently and places the resulting part $m_{\mathcal{R}_i}$ at the position chosen at embedding time. We explore a multitude of different partition strategies in Figure 10. For generative models that produce images across multiple scales, we define the partition once on the finest token grid. At coarser scales, each token is assigned to the region that overlaps most of its area. We further explore this embedding scheme in Appx. E.2.

Choosing the Partition. Separating the image into different partitions has limited effect on robustness under value-metric or regeneration attacks. However, geometric transformations drastically move tokens across partition boundaries and votes for one part of the message can be counted toward another. Our evaluation of many candidate partitions in Appx. E.2 (Figure 10, Table 12) yields two simple rules that are designed to keep the mismatch between original and attacked layout small. *(i) Regions should be spatially contiguous.* Interleaving chunks across tokens, in raster order or at random, is the most natural extension, but a geometric shift then scrambles nearly every token, whereas for contiguous regions mostly tokens along the boundary are affected. *(ii) Regions should be preserved by the expected transforms.* A partition is more robust to precisely those transforms that map it onto itself: a horizontal partition survives a horizontal flip.

Capacity. The number of partitions R directly constructs a trade-off between capacity and robustness. Each partition carries $1/R$ of the tokens, so the number of positions voting for each message bit shrinks monotonically with R and their reliability drops by a factor of $\sqrt{R}$. We quantify this trade-off, and the gain from shorter messages, in Appx. E.2 and Figure 11.

4 Empirical Evaluation

Experimental Setup. Unless stated otherwise, we evaluate ByteMark primarily on Infinity-2B (Han et al., 2025) with $\delta = 2$ and 31 bits per message. Messages with 62 bits on Infinity-2B use the *horizontal* partition of the generated image, messages with 124 bits leverage the *diagonal* partition, messages with 248 bits use the *windmill* pattern, and all messages with more bits use the *grid* partition (according to the description in Sec. 3.3, also see the visualizations in Figure 1 (b) and Figure 10). For the detection of ByteMark, we leverage the *oracle* setting for the TPR@1%FPR and the *blind* setting for the message extraction and bit accuracy. We additionally evaluate on other bit-wise generative models, such as Instella-T2I (Wang et al., 2025), BiGR-L-512 (Hao et al., 2025), and Infinity-8B (Han et al., 2025). We evaluate the robustness of our method with 500 prompts taken from the MS-COCO 2014 validation dataset and compute FID, KID, and CLIP scores against 5000 MS-COCO validation images. We provide additional information about the setting in Appx. B.

Baselines. To the best of our knowledge, we are the first to develop an in-generation multi-bit watermark for bit-wise autoregressive models. Therefore we compare ByteMark with BitMark (Kerner et al., 2025) and multiple post-processing watermarks. Namely, TrustMark (Bui et al., 2025), StegaStamp (Tancik et al., 2020), WAM[1] (Sander et al., 2025), InvisMark[2] (Xu et al., 2025), VideoSeal (Fernandez et al., 2024), PixelSeal (Souček et al., 2025), and ChunkySeal (Petrov et al., 2025). All post-hoc baselines are explained more thoroughly in Appx. A.2. We additionally implement a naive baseline, which embeds the message directly in the generated bits. We describe the baseline in more detail in Appx. C.2 and denote it as *Naive Base*.

4.1 ByteMark is Robust Against Watermark Removal Attacks

We evaluate our ByteMark against three attack families, *value-metric* attacks that blur, compress, noise, or recolor the images, *geometric attacks* that influence the orientation of the image, and

[1]When run on clean images WAM's detector always returns a message of only zeros.

[2]As also found by Souček et al. (2025), InvisMark effectively only encodes 94 bits, due to 6 bits of the message having fixed values. We therefore evaluate InvisMark on only the functional bits.

Table 1: **BYTEMARK is highly robust, especially against targeted removal attacks.** We report the mean bit accuracy of the recovered payload under each attack. Avg. is the mean over the 10 attack columns. Per column, **best** and second best results are highlighted.

Method	None	Value-metric				Geometric				Regeneration		Avg.
		Blur	JPEG	Noise	Color	Rot	Crop	H-flip	V-flip	VAE	CtrlReg+	
Naive Base - 32 bit	0.87	0.73	0.74	0.62	0.57	0.49	0.50	0.50	0.52	0.63	0.48	0.58
BYTEMARK- 31 bit	**1.00**	**1.00**	**1.00**	0.99	**0.80**	0.64	0.84	0.87	0.73	**1.00**	**0.79**	**0.87**
BYTEMARK- 62 bit	**1.00**	**1.00**	**1.00**	0.95	0.79	0.59	0.77	0.80	0.55	**1.00**	0.71	0.82
BYTEMARK- 124 bit	**1.00**	**1.00**	**1.00**	0.90	0.73	0.54	0.66	0.61	0.55	**1.00**	0.63	0.76
BYTEMARK- 248 bit	**1.00**	**1.00**	0.99	0.85	0.70	0.54	0.60	0.55	0.54	0.99	0.60	0.74
BYTEMARK- 496 bit	0.99	0.98	0.96	0.79	0.67	0.52	0.55	0.52	0.50	0.98	0.57	0.70
BYTEMARK- 992 bit	0.98	0.95	0.92	0.73	0.63	0.51	0.52	0.50	0.50	0.93	0.55	0.67
BYTEMARK- 1984 bit	0.93	0.89	0.85	0.67	0.60	0.51	0.51	0.50	0.50	0.86	0.53	0.64
InvisMark - 94 bit[2]	**1.00**	**1.00**	**1.00**	0.95	0.50	0.53	**1.00**	**1.00**	**1.00**	0.89	0.48	0.83
StegaStamp - 100 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.76	0.52	0.50	0.47	0.49	**1.00**	0.61	0.73
TrustMark - 100 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.59	0.53	0.51	**1.00**	0.47	**1.00**	0.52	0.76
VideoSeal - 256 bit	**1.00**	**1.00**	0.99	0.82	0.64	0.73	0.89	**1.00**	0.49	0.99	0.51	0.81
PixelSeal - 256 bit	**1.00**	**1.00**	**1.00**	0.79	0.68	**0.89**	0.98	**1.00**	0.50	0.99	0.52	0.83
ChunkySeal - 1024 bit	**1.00**	**1.00**	0.98	0.76	0.51	0.69	0.97	**1.00**	0.51	0.86	0.52	0.78
WAM (global) - 32 bit[1]	**1.00**	**1.00**	**1.00**	**1.00**	0.70	0.64	**1.00**	**1.00**	0.65	**1.00**	0.74	**0.87**
WAM (region) - 32 bit[1]	**1.00**	**1.00**	**1.00**	**1.00**	0.53	0.55	**1.00**	**1.00**	0.59	**1.00**	0.50	0.82

regeneration attacks that pass the images through generative models, such as VAEs or regenerate the image with attacks such as CtrlRegen+ (Liu et al., 2025). We detail the settings precisely in Appx. B.3. We report the mean bit accuracy of our watermark under *blind* detection and the baselines in Table 1 and the TPR@1%FPR under *oracle* detection in Table 13. The results show that our approach achieves high bit accuracy and TPR@1%FPR for most conventional attacks.

Message Recovery. Table 1 shows that BYTEMARK recovers the 31-bit message perfectly under blur, JPEG compression, and VAE regeneration, and nearly perfectly under noise. The strongest advantage over the post-hoc baselines is under the targeted regeneration. Averaged over all attacks, BYTEMARK matches the best post-hoc baseline at 31 bits while embedding the message in-generation. Increasing the payload size from 31 to 1984 bits leaves robustness against blur, JPEG, and VAE regeneration near unchanged, but becomes more vulnerable against geometric attacks.

Watermark Detection. Table 13 reports the TPR@1%FPR under oracle detection. BYTEMARK is detected at 100% TPR@1%FPR under blur, JPEG compression, noise, and VAE regeneration, and loses little under color. Even under CtrlRegen BYTEMARK retains a TPR of at least 77% across all payloads. The strongest attack against BYTEMARK is arbitrary rotation, similarly to BITMARK. However, rotation can be estimated with openly accessible orientation-estimation tools (Maji & Bose, 2020); re-rotating the image restores near-perfect TPR@1%FPR, as we evaluate in Appx. F.

4.2 BYTEMARK IS HIGHLY RADIOACTIVE

A watermark is called *radioactive* (Sablayrolles et al., 2020; Sander et al., 2024) when a model trained on watermarked content produces outputs that carry a detectable trace of the mark. In this setting, the original model $\mathcal{M}_1$ generates outputs marked with, *e.g.,* our BYTEMARK. As the watermarked content is published online, it might be used in the training or fine-tuning set of a new model. *Radioactivity* then enables the detection of the use of the watermarked content, even when it was used to train subsequent models.

Setup. Following Kerner et al. (2025), we simulate the highly costly training by fully fine-tuning a pre-trained Infinity-2B. We generate $1{,}000$ images with Infinity-2B from MS-COCO 2014 validation prompts and watermark the data with $\delta = 2$ and a fixed 31-bit message. The watermarked outputs of $\mathcal{M}_1$ are then used to fine-tune the second model $\mathcal{M}_2$. We further vary the ratio of watermarked to clean generated data. We provide the detailed hyperparameters in Appx. B.1.

Multi-bit Radioactivity. Table 2 reports the TPR@1%FPR for the detection of BYTEMARK in the fine-tuned $\mathcal{M}_2$. The results highlight that our watermark is highly radioactive and can be detected at

Table 2: **BYTEMARK is radioactive and decays gracefully.** Model $\mathcal{M}_2$ is fine-tuned on 1,000 images of which the stated (in the first header row) percentage (%) fraction is watermarked. We report TPR@1%FPR (in %) under blind and oracle detection strategies, and bit accuracy of the recovered 31-bit message. The p-value is computed from a Mann-Whitney U test of $\mathcal{M}_2$-generated vs. $\mathcal{M}_1$-generated images.

Fraction of Watermarked Data	0%	1%	5%	10%	25%	50%	100%
BITMARK	0.3	1.0	2.9	22.7	98.3	100.0	100.0
Blind	1.0	0.8	0.6	1.6	9.6	98.4	100.0
Mann-Whitney p	0.21	0.08	0.02	10^{-9}	10^{-97}	$<10^{-160}$	$<10^{-160}$
Oracle	0.8	1.6	6.4	23.8	98.4	100.0	100.0
Mann-Whitney p	0.19	10^{-3}	10^{-40}	10^{-105}	$<10^{-160}$	$<10^{-160}$	$<10^{-160}$
Bit accuracy	50.9±8.7	51.6±8.8	56.5±9.0	62.6±8.7	79.9±7.9	95.4±4.3	99.9±0.6

100% TPR@1%FPR when making up 50% or more of the training data, following the radioactivity of BITMARK very closely. Similar to BITMARK, BYTEMARK influences nearly all bits with a repetitive signal that is reinforced across all scales and all tokens. This strong repetition and the high robustness that we explore in Table 13 are likely the reasons why BYTEMARK exhibits such high radioactivity. We additionally report the p-value of a Mann-Whitney U test computed on 500 samples generated without active watermarking by $\mathcal{M}_1$ and $\mathcal{M}_2$, which results in a significant shift under the oracle detection when 1% of the training data is watermarked. The distinctive benefit of BYTEMARK is that not only the presence of the watermark is radioactive but also its message. The last row of Table 2 reports the bit accuracy of the message decoded from the outputs of $\mathcal{M}_2$: when BYTEMARK-ed images make up at least 50% of the training data, the message is recovered with more than 95% accuracy.

4.3 BYTEMARK EXTENDS TO OTHER MODELS

While our main evaluation focuses on Infinity-2B model, our BYTEMARK is also applicable to other bit-wise architectures, such as Infinity-8B (Han et al., 2025), BiGR (Hao et al., 2025), and Instella (Wang et al., 2025). The embedding procedure is the same for all models. Only the size of each token used to embed the message changes depending on the architecture. The partition strategies still apply for models with a spatial grid, enabling embedding longer messages. We provide more details on these models, their configuration, and the adaptions in Appx. B.4. We report the bit accuracy under each attack for the different models in Table 3. The robustness against blur, JPEG and VAE regeneration stays high across architectures. Geometric attacks remain the most challenging for all models. Overall, BYTEMARK transfers to different bit-wise models without changes to the method and embeds the message reliably.

Table 3: **BYTEMARK extends across many architectures.** We report bit accuracy against each attack. **Avg.** is the mean over the 10 attack columns (excluding None). Per column, **best** and second best results are highlighted.

Model	Bits	δ	None	Value-metric				Geometric				Regeneration		Avg.
				Blur	JPEG	Noise	Color	Rot	Crop	H-flip	V-flip	VAE	CtrlReg+	
Infinity-2B	31	2.0	**1.00**	**1.00**	**1.00**	**0.99**	**0.80**	**0.64**	**0.84**	**0.87**	**0.73**	**1.00**	**0.79**	**0.87**
Infinity-8B	55	2.0	0.99	0.97	0.96	0.87	0.73	0.57	0.75	0.67	0.66	0.91	0.59	0.77
Instella	63	1.5	0.97	0.96	0.97	0.83	0.66	0.49	0.53	0.54	0.52	0.97	0.73	0.72
BiGR	31	3.0	0.82	0.76	0.79	0.62	0.51	0.49	0.59	0.57	0.57	0.78	0.60	0.63

4.4 ABLATIONS

Image Quality. We analyze the impact of BYTEMARK on the generation quality in Table 4. Therefore, we analyze FID, KID, and CLIP scores against 5000 MS-COCO validation images and compute the GenEval2 (GE2) scores (Kamath et al., 2025) for prompt following. In GE2 each prompt is decomposed into atoms and a visual question answering model answers one templated question per

Table 4: **BYTEMARK does not impact visual quality.** We report FID ↓ and KID ↓ for visual diversity and CLIP ↑ and GE2 ↑ for prompt following capability.

Model	Method	FID ↓	KID $\times 10^3$ ↓	CLIP ↑	GE2 (AM) ↑	GE2 (GM) ↑
Infinity-2B	Clean	37.3	13.15	16.48	61.6	15.9
	Ours	35.3	11.39	16.52	61.0	15.7
Infinity-8B	Clean	28.3	8.48	16.48	74.1	29.0
	Ours	27.2	6.96	16.47	73.9	29.0
BiGR	Clean	28.7	7.30	17.68	*n/a*	*n/a*
	Ours	24.6	4.97	17.72	*n/a*	*n/a*
Instella	Clean	25.7	5.24	16.43	37.9	5.7
	Ours	26.2	5.41	16.42	35.6	6.4

atom. GE2 (AM) aggregates the per atom accuracy as arithmetic mean and GE2 (GM) as geometric mean. Since BiGR is a class conditioned model for ImageNet, we compute the quality against 5000 ImageNet validation images and cannot compute the GE2 scores for this model. Our results show that applying our method preserves the visual fidelity of the generated content, with FID, KID, and CLIP scores remaining almost unaffected. We also provide visual examples in Figure 13. In addition to its limited impact on visual quality, BYTEMARK also adds computational overhead comparable to BITMARK to the generation and detection speed, as we explore in Appx. F.3.

Watermark Strength. We additionally analyze the impact of the watermark strength, represented by the bias δ on the generation quality of the model and the detection of the watermark. We report the quality metrics at different δ in Table 18. Our results show that for $\delta \leq 3$, BYTEMARK has negligible influence on the generation quality. Our main results select $\delta = 2$, due to the trade-off in imperceptibility and robustness. We provide visual examples for the different strengths in Figure 12. We also find that with higher δ, the watermark becomes more robust, with the highest gain under geometric attacks and regeneration attacks.

Higher Payload Capacity. In Sec. 3.3 we propose a method to extend the message length beyond single token boundaries. We analyze the impact of this design on the quality in Table 19. We find that different message lengths have nearly no impact on the image quality, meaning that the capacity extension is free in image quality. However the impact is larger on the detection, as doubling the number of bits also halves the tokens that vote for the message. This results in a decline in overall robustness with longer messages. We observe in Table 1 that we can expand the payload length at the cost of a graceful gradual degradation in the robustness. We further extend our ablations to the opposite direction, reducing the overall payload size from 31-bits to 15, 7, 3 and 2. Our results in Figure 11 show that smaller messages result in higher robustness.

Composition with Post-Hoc Methods. In deployment settings, a generation-time mark may be paired with post-hoc watermarks. We therefore evaluate if BYTEMARK and post-hoc watermarks interfere with each other in Table 15 and Table 16. We find that the marks coexist almost perfectly, with both BYTEMARK and the post-hoc watermarks being mostly unimpacted across all attacks.

5 CONCLUSIONS

We presented BYTEMARK, a multi-bit watermarking scheme for bit-wise image generators that encodes each message bit through the choice of repeating or flipping the preceding bit. This design maintains the delicate balance between zeros and ones during bit-wise generation, preserving the quality of generated content. BYTEMARK is robust to value-metric, geometric, and regeneration attacks and, through spatial partitioning, supports payloads of up to 1984 bits. Furthermore, BYTEMARK exhibits radioactivity such that both the watermark's presence and its embedded message survive downstream model fine-tuning. Applying the watermark adds only very small overhead onto the generation speed and both detection and decoding can be done in a single forward pass, performed in less than a second. Together, these capabilities establish BYTEMARK as a high-capacity and persistent watermarking approach that extends precise provenance tracing from generated images to downstream models.

AI DISCLOSURE

We used generative AI (GenAI) tools for coding assistance; this includes assistance implementing our BYTEMARK and other evaluated methods, experiments as well as visualizations. Furthermore, GenAI provided feedback and initial interpretations for experiments and hypotheses. Our method was evaluated on synthetically generated data. We used AI to improve the quality of the text in the paper as well as to polish the final version including the automated LLM Feedback offered by the conference. This includes sections containing mathematical claims, where AI was used only for formatting.

We have not used generative AI tools for assistance with translation, cleaning, and reformatting datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims or assisting in the writing of proofs. We have reviewed all AI-assisted work by checking the LLM-generated code and images. The manuscript was fully reworked and proof-read by all authors. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of GenAI.

ETHICS STATEMENT

This work addresses critical societal challenges posed by increasingly realistic AI-generated imagery, including misinformation, fraud, and harmful content dissemination. BYTEMARK contributes towards a mitigation of these harms by enabling reliable provenance of generated images even through multiple training cycles via radioactivity. However, we also recognize that watermarking is only a component of provenance rather than a final solution. It requires active application and can be removed by adversaries. The absence of a watermark should thus not be interpreted as evidence of natural content.

REPRODUCIBILITY STATEMENT

We provide comprehensive implementation details together with open-sourced code to ensure reproducibility of our results. All experimental configurations, including hyperparameters for fine-tuning and image generation are detailed in Appx. B.1. Our evaluation is performed on the open-source MS-COCO validation set. All experiments were conducted on standard hardware specified in Appx. B.2.

REFERENCES

President Joseph R Biden. *Executive Order 14110 on Safe, Secure, and Trustworthy AI*. Kdp Print Rescue. W. Frederick Zimmerman, 2026. ISBN 9781934840757. URL https://books.google.de/books?id=P9pA0gEACAAJ.

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

California Legislative Information. SB 942, Becker. California AI Transparency Act., 2024. URL https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=202320240SB942.

Bobby Chesney and Danielle Citron. Deep fakes: A looming challenge for privacy, democracy, and national security. *Calif. L. Rev.*, 107:1753, 2019.

Coalition for Content Provenance and Authenticity (C2PA). Content credentials: C2PA technical specification, version 2.4. https://spec.c2pa.org/specifications/specifications/2.4/specs/C2PA_Specification.html, 2024. Accessed: 2026-08-31.

Cyberspace Administration of China, Ministry of Industry and Information Technology, Ministry of Public Security, and National Radio and Television Administration. Measures for the labeling

of artificial intelligence-generated and synthetic content. Joint Ministerial Decree No. [2025] / Official Notice available via the Cyberspace Administration of China, 2025. URL https://www.cac.gov.cn/2025-03/14/c_1743654684782215.htm. Effective September 1, 2025.

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

European Data Protection Supervisor. *AI Act Regulation (EU) 2024/1689 – Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) (Text with EEA relevance)*. Publications Office of the European Union, 2025. doi: doi/10.2804/4225375.

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

Pierre Fernandez, Hady Elsahar, I. Zeki Yalniz, and Alexandre Mourachko. Video seal: Open and efficient video watermarking, 2024. URL https://arxiv.org/abs/2412.09492.

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.

Nikola Jovanović, Ismail Labiad, Tomas Soucek, Martin Vechev, and Pierre Fernandez. Watermarking autoregressive image generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=hVdD72iom4.

Amita Kamath, Kai-Wei Chang, Ranjay Krishna, Luke Zettlemoyer, Yushi Hu, and Marjan Ghazvininejad. Geneval 2: Addressing benchmark drift in text-to-image evaluation, 2025. URL https://arxiv.org/abs/2512.16853.

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. URL https://openreview.net/forum?id=VSir0FzFnP.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17061–17084. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

L. Kish. *Survey Sampling*. Wiley, 1965. ISBN 9780471489009. URL https://books.google.de/books?id=yoppAAAAMAAJ.

Jan Kociszewski, Hubert Jastrzębski, Tymoteusz Stępkowski, Filip Manijak, Krzysztof Rojek, Franziska Boenisch, and Adam Dziedzic. SERUM: Simple, efficient, robust, and unifying marking for diffusion-based image generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=AiBUm6iKBf.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. URL https://arxiv.org/abs/2506.15742.

Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2004. ISSN 0047-259X. doi: https://doi.org/10.1016/S0047-259X(03)00096-4. URL https://www.sciencedirect.com/science/article/pii/S0047259X03000964.

Yepeng Liu, Yiren Song, Hai Ci, Yu Zhang, Haofan Wang, Mike Zheng Shou, and Yuheng Bu. Image watermarks are removable using controllable regeneration from clean noise. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mDKxlfraAn.

Subhadip Maji and Smarajit Bose. Deep image orientation angle detection, 2020. URL https://arxiv.org/abs/2007.06709.

Michel Meintz, Jan Dubiński, Franziska Boenisch, and Adam Dziedzic. Radioactive watermarks in diffusion and autoregressive image generative models, 2025. URL https://arxiv.org/abs/2506.23731.

Aleksandar Petrov, Pierre Fernandez, Tomáš Souček, and Hady Elsahar. We can hide more bits: The unused watermarking capacity in theory and in practice, 2025. URL https://arxiv.org/abs/2510.12812.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models . In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, Los Alamitos, CA, USA, June 2022. IEEE Computer Society. doi: 10.1109/CVPR52688.2022.01042. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01042.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Radioactive data: tracing through training. In *International Conference on Machine Learning*, pp. 8326–8335. PMLR, 2020.

Tom Sander, Pierre Fernandez, Alain Oliviero Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=qGiZQb1Khm.

Tom Sander, Pierre Fernandez, Alain Oliviero Durmus, Teddy Furon, and Matthijs Douze. Watermark anything with localized messages. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IkZVDzdC8M.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 631:755–759, 07 2024. doi: 10.1038/s41586-024-07566-y.

Tomáš Souček, Pierre Fernandez, Hady Elsahar, Sylvestre-Alvise Rebuffi, Valeriu Lacatusu, Tuan Tran, Tom Sander, and Alexandre Mourachko. Pixel seal: Adversarial-only training for invisible image and video watermarking, 2025. URL https://arxiv.org/abs/2512.16874.

Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2117–2126, 2020.

Keyu Tian, Yi Jiang, Zehuan Yuan, BINGYUE PENG, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=gojL67CfS8.

Ze Wang, Hao Chen, Benran Hu, Jiang Liu, Ximeng Sun, Jialian Wu, Yusheng Su, Xiaodong Yu, Emad Barsoum, and Zicheng Liu. Instella-t2i: Pushing the limits of 1d discrete latent space image generation, 2025. URL https://arxiv.org/abs/2506.21022.

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-rings watermarks: Invisible fingerprints for diffusion images. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Z57JrmubNl.

Jiaqi Wu, Yuchen Zhou, Dennis Tsang Ng, Xingyu Shen, Kidus Zewde, Ankit Raj, Tommy Duong, and Simiao Ren. When the forger is the judge: Gpt-image-2 cannot recognize its own faked documents, 2026. URL https://arxiv.org/abs/2604.25213.

Rui Xu, Mengya Hu, Deren Lei, Yaxi Li, David Lowe, Alex Gorevski, Mingyu Wang, Emily Ching, and Alex Deng. Invismark: Invisible and robust watermarking for ai-generated image provenance. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pp. 909–918, February 2025.

Yigit Yilmaz, Elena Petrova, Mehmet Kaya, Lucia Rossi, and Amir Rahman. Unimark: Unified adaptive multi-bit watermarking for autoregressive image generators, 2026. URL https://arxiv.org/abs/2604.11843.

Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18431–18441, October 2025.

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

Yue Zhao, Yuanjun Xiong, and Philipp Kraehenbuehl. Image and video tokenization with binary spherical quantization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yGnsH3gQ6U.

A Extended Background

A.1 In-Generation Watermarks

BitMark (Kerner et al., 2025) watermarks bit-wise autoregressive image models (e.g. Infinity) at the level of the individual bits that make up each token. The method fixes a pattern length n and partitions $\{0,1\}^n$ into a green list G and a red list $R = \{0,1\}^n \setminus G$, with $\gamma = |G|/2^n$ (the paper uses $|G| = |R|$, so $\gamma = 1/2$). For each token of d bits, the context of bit $j \geq n$ is the $n-1$ preceding bits of the same token. Note that the first $n-1$ bits of every token are left unbiased, so $k = d - (n-1)$ bits per token carry the watermark. If appending b_j to the context yields a pattern in G, a bias $\delta > 0$ is added to the logit of b_j before sampling. This increases the likelihood of a green bit to be sampled. The lists are chosen such that the two continuations of any context never fall in the same list, such that every eligible position is biased. By default, BitMark uses $n = 2$ with $G = \{01, 10\}$, which also avoids skewing the marginal frequency of 0s and 1s, which the tokenizer would otherwise wash out on re-encoding. Detection re-encodes the candidate image with the model's encoder, recomputes the n-bit patterns, counts the number C of green bits among the $T = \sum_i r_i k$ eligible positions, where r_i is the number of tokens at scale i, and tests the null hypothesis H_0: the bits were generated without knowledge of G via $z = (C - \gamma T)/\sqrt{T\gamma(1-\gamma)}$. An image is then defined as watermarked, when $z \geq 4$, which corresponds to a probability of false positive of 3×10^{-5}.

UniMark (Yilmaz et al., 2026) proposes a multi-bit watermark for IARs. It embeds messages by exchanging sampled tokens before decoding into the image, rather than biasing the sampler directly. Similar to WMAR (Jovanović et al., 2025), every token is assigned to a green or red list. However, the list assignment differs, since it depends on a secret key, and the position index, not the prior token. A B-bit message is then BCH-encoded to n bits and the token positions are cut into n blocks, with each block carrying one bit of the message. A block that is assigned the bit zero (0) embeds tokens from the red list and a block that is assigned the bit one (1) embeds tokens from the green list. Thus depending on the message bit, every token in a block is swapped with the most similar embedding in the assigned list. For message extraction, the image is encoded and each block's green fraction measured. The green fraction then directly decodes into the message and can be decoded with BCH. The design of the watermark relies on a key factor that does not extend to bit-wise models. Namely, it requires the separation of the tokens into red and green lists and the selection of a similar token from the correct list. In bit-wise models, similarity is defined via Hamming distance, which means changes in the tokens would be minor and likely not robust against encoding with the bit-wise encoder, which changes each token drastically, as shown by Kerner et al. (2025).

A.2 Post-Hoc Watermarks

Current post-hoc watermarks share a common structure. They consist of a trained encoder E, which adds a learned residual to an image and a trained decoder D, which recovers the message. The methods differ mostly in the construction of the networks, the transformations used during training, the objective of the watermark and the size of the embedded payload.

StegaStamp (Tancik et al., 2020) targets specifically robustness against real-world processing and is trained against a print-to-photograph pipeline incorporating a lot of value-metric attacks. These augmentations result in high robustness at the cost of imperceptibility.

TrustMark (Bui et al., 2025) targets imperceptibility at arbitrary resolutions. The watermark itself is sampled at 256×256 resolution and then up-sampled to the full resolution image. The incorporation of a focal frequency loss results in the watermark being present mostly in low frequencies. While the paper evaluates payloads from 32 to 200 bits, the released checkpoint encodes 100 bits, optionally protected by a BCH error-correcting code. We use this checkpoint with the full 100-bit payload and no error correction, so that reported bit accuracies reflect the raw watermark.

InvisMark (Xu et al., 2025) follows TrustMark and embeds 100^2 bits. It also uses the same up-scaling approach, but leverages a worst-case training strategy to improve robustness, sampling transformations with the highest recovery loss. The quality loss combined of pixel, perceptual, frequency and adversarial loss results in a high image quality.

VideoSeal (Fernandez et al., 2024) is primarily designed for video watermarking, but can also be used as a general image watermark. The core impact for improved performance is a two-stage training process, that results in higher robustness and faster and more stable convergence.

ChunkySeal (Petrov et al., 2025) asks the question whether the capacity of current post-hoc watermarks can be improved. They derive bounds on how many bits an image can carry under a PSNR constraint and under linear transformations, and show that current models sit orders of magnitude below them. As evidence for this gap, the authors scale VideoSeal to 1024 bits, showing that the capacity of post-hoc watermarks is not yet reached.

PixelSeal (Souček et al., 2025) improves on existing watermarks by observing multiple failure modes. Training does not use standard perceptual losses such as MSE or LPIPS which require careful hyperparameter tuning. Instead the embedder is trained against a discriminator network and the message loss. The training is performed in three stages, beginning with message extraction, then adversarial loss for imperceptibility and finally task-specific fine-tuning.

Watermark Anything with Localized Messages (WAM) (Sander et al., 2025)'s core objective is localization of the watermark within an image. The watermark extraction is based on a segmentation network that returns a score and message for different segments of the image. While the payload is restricted to 32 bits per message, multiple messages can be embedded into a single image.

B Experimental Setting

We provide access to an anonymized version of the code here.

B.1 Hyperparameters

Our setting to evaluate radioactivity follows Kerner et al. (2025). We fine-tune $\mathcal{M}_2$ on 1,000 prompts from the MS-COCO 2014 validation dataset, with images generated by Infinity-2B. The model is fine-tuned for 1 epoch with a learning rate of 10^{-4} and a batch-size of 1. For evaluation, the model $\mathcal{M}_2$ generates 500 images with MS-COCO validation prompts that were not used for training.

B.2 Hardware configuration and computational cost

We performed experiments on compute nodes with Intel(R) Xeon(R) Gold 5420+ with 4 NVIDIA L40 and Intel(R) Xeon(R) Gold 6330 CPUs with 4 NVIDIA A40 GPUs. Radioactivity and large-scale experiments were performed on 4 NVIDIA A100-SXM4-40GB GPUs.

B.3 Attack settings

We compare each watermarking method across identical attack settings on the resolution produced by the generative models. Attacks with a random component are drawn independently per image. The same seed is used for every method.

- **None** — the un-attacked image.
- **Blur** — Gaussian blur with kernel size 7.
- **Noise** — Gaussian noise with $\sigma = 0.1$.
- **JPEG** — JPEG re-encoding at quality 75%.
- **Color** — Color jitter with fixed hue $+0.3$, and saturation, contrast, and brightness scaled by 3.0.
- **Crop** — Random crop to 0.7 of each side, resized back to the original size.
- **Rot** — Random rotation sampled uniformly on $[-180°, +180°]$.
- **Rot+crp** — Same rotation, then cropped to the largest axis-aligned rectangle that avoids the border and resized back.
- **Rot(dsk)** — Same rotation, then *deskewed*, by using an orientation-estimation network which predicts the angle and rotates back.
- **H-flip**, **V-flip** — Deterministic horizontal and vertical reflection.

- **VAE** — Encode and decode through the Stable Diffusion 2.1 autoencoder.
- **CtrlReg+** — CtrlRegen (Liu et al., 2025) attack, which regenerates the image through a ControlNet-guided Stable Diffusion image-to-image pipeline at strength 0.5, conditioned on its own edge map.

B.4 Model Details

We provide information about the bits per token and the full number of bits in Table 5. Note that the bits per token guide the default length of the message embedded by ByteMark, which we denote by "Content positions". The spatial token grid relates to how the tokens of the bit stream are ordered. The images in Instella are represented as one long bit-stream without spatial information.

Table 5: **Geometries of the evaluated models.** We report the number of tokens, bits per token, content positions, bits in total, and whether the model uses a spatial token grid.

	Infinity 2B	**Infinity 8B**	**BiGR**	**Instella**
Tokens	10,521 (13 scales)	10,521 (13 scales)	1,024 (32×32)	128 (1D)
Bits per token	32	56 (14×4)	32	64
Content positions	31	55	31	63
Bits in total	336,672	589,176	32,768	8,192
Spatial token grid	Yes	Yes	Yes	No

C Bit-Level Design Space

Sec. 3 presents how we embed each message bit in the transitions between consecutive bits rather than in the bits themselves. This section provides details on the transition rules and then the main motivation for this choice.

C.1 Detailed Transition Rules

ByteMark is embedded in the transitions between consecutive bits rather than in the bits themselves. We leverage the most robust green and red lists from the BitMark watermark, namely $G^{(0)} := \{01, 10\}$ and $G^{(1)} := \{00, 11\}$, read as pairs $(b_{j-1}\, b_j)$. To embed the message m, we nudge the prediction of the model by adding a small bias δ to the logit ℓ^{b_j} of the bit value that completes a pair in $G^{(m_j)}$. We have the following 4 cases:

(1) If $m_j = 1$ and the previous bit value $b_{j-1} = 0$, then we add δ to the logit for bit 0, namely ℓ^0.

$$
\begin{aligned}
p_j(b_j{=}0 \mid b_{j-1}{=}0, m_j{=}1) &= \frac{\exp(\ell_j^0 + \delta)}{\exp(\ell_j^0 + \delta) + \exp(\ell_j^1)}, (00) \in G^{(1)}, \\
p_j(b_j{=}1 \mid b_{j-1}{=}0, m_j{=}1) &= \frac{\exp(\ell_j^1)}{\exp(\ell_j^0 + \delta) + \exp(\ell_j^1)}, (01) \in G^{(0)}.
\end{aligned} \tag{6}
$$

(2) If $m_j = 1$ and the previous bit value $b_{j-1} = 1$, then we add δ to the logit for bit 1, namely ℓ^1.

$$
\begin{aligned}
p_j(b_j{=}0 \mid b_{j-1}{=}1, m_j{=}1) &= \frac{\exp(\ell_j^0)}{\exp(\ell_j^0) + \exp(\ell_j^1 + \delta)}, (10) \in G^{(0)}, \\
p_j(b_j{=}1 \mid b_{j-1}{=}1, m_j{=}1) &= \frac{\exp(\ell_j^1 + \delta)}{\exp(\ell_j^0) + \exp(\ell_j^1 + \delta)}, (11) \in G^{(1)}.
\end{aligned} \tag{7}
$$

(3) If $m_j = 0$ and the previous bit value $b_{j-1} = 1$, then we add δ to the logit for bit 0, namely ℓ^0.

$$p_j(b_j{=}0 \mid b_{j-1}{=}1, m_j{=}0) = \frac{\exp(\ell_j^0 + \delta)}{\exp(\ell_j^0 + \delta) + \exp(\ell_j^1)}, (10) \in G^{(0)},$$
$$p_j(b_j{=}1 \mid b_{j-1}{=}1, m_j{=}0) = \frac{\exp(\ell_j^1)}{\exp(\ell_j^0 + \delta) + \exp(\ell_j^1)}, (11) \in G^{(1)}. \tag{8}$$

(4) If $m_j = 0$ and the previous bit value $b_{j-1} = 0$, then we add δ to the logit for bit 1, namely ℓ^1.

$$p_j(b_j{=}0 \mid b_{j-1}{=}0, m_j{=}0) = \frac{\exp(\ell_j^0)}{\exp(\ell_j^0) + \exp(\ell_j^1 + \delta)}, (00) \in G^{(1)},$$
$$p_j(b_j{=}1 \mid b_{j-1}{=}0, m_j{=}0) = \frac{\exp(\ell_j^1 + \delta)}{\exp(\ell_j^0) + \exp(\ell_j^1 + \delta)}, (01) \in G^{(0)}. \tag{9}$$

C.2 Naive Baseline

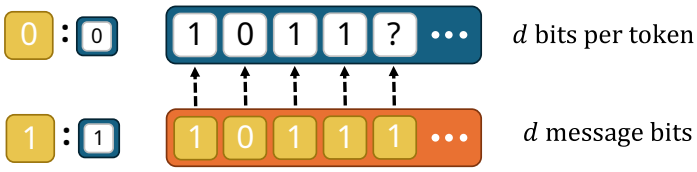


Figure 2: **Visualization of the naive baseline.** The naive baseline biases every bit within a token directly to the message bit, which disrupts the generation process of the model.

To construct the naive baseline, we bias each bit position j directly toward the message bit m_j (Figure 2). This method fails on both robustness against attacks and imperceptibility. When bits are biased directly, a message with an unbalanced amount of zeros and ones triggers the quality degradation visualized in Figure 3. We evaluate multiple versions of the naive baseline to mitigate the quality issues. The *mirrored* version embeds a balanced message in the first 16 bits and then mirrors it directly. The *interleaved* version repeats 01 for 32 bits. The construction requirement that messages have to be balanced cuts the total number of messages that can be embedded by 86%. Other options are lowering the bias δ or only applying the bias per chance or on later scales. However none are effective at fully improving visual quality, as shown by Figure 4.

Robustness. During generation, the message is embedded perfectly, with 72% of biased positions taking the message value and a majority vote over the tokens recovering the message perfectly. However, the signal does not fully survive re-encoding. After the image is decoded to pixels and encoded again, the per-position agreement with the message falls to 50.3%. Majority voting over the ∼10,000 tokens of an image still recovers 87% of the bits, but most attacks remove it. We call this behavior *tokenizer balancing*: the bit-wise encoder maps every image to a token stream with a balanced number of zeros and ones, as also observed by Kerner et al. (2025). A watermark that shifts this balance is therefore undone by the very encoder that has to read it back.

Quality. Two factors degrade image quality for the naive baseline. The first is the *tokenizer balancing*. A message with unequal amounts of ones and zeros disrupts the balance and destroys quality, as we show in Figure 3. An excess of zeros results in a red-shift with horizontal ridges across the image. An excess of ones results in a cyan-shift with blurry vertical ridges. The second is *codeword clustering*, which occurs even for a perfectly balanced message. Since every token is pushed towards the same value, all tokens of the image are pulled toward one point in the codebook. The token stream loses the diversity needed to represent textures and fine detail, and the image degrades in the same way as under an unbalanced message. We visualize this in Figure 4.



Figure 3: ***Tokenizer balancing* degrades quality.** The images were generated with the same prompt, but nudged to different lists. An un-watermarked image has $\sim$50% ones. An imbalance towards zeros causes a shift towards red and horizontal ridges, while an excess of ones visualizes as a cyan tint and vertical ridges.



Figure 4: ***Codeword clustering* degrades quality.** We analyze different versions of the naive baseline and find that balancing the number of zeros and ones is not enough.

D Detection Properties

D.1 Observed Bias

Under the fair-coin model the blind statistic of Eq. (3) has mean of $\sqrt{2(d-1)/\pi} \approx 4.4$ at $d-1 = 31$ for un-watermarked images. However, our experiments in Table 6 show that the images carry an excess above this fair-coin model, which we denote as *inflation*.

Per-Position Bias. The inflation is not the same across all bit-positions, with some being more likely to repeat or switch the bit than expected by chance. The per-position bias makes the decoder more likely to output a specific message and over- or under-reports the bit accuracy with specific payloads. We visualize the bias in Figure 5 and find that the message bit at position 2 without correction is much more likely to be decoded to 0 than 1.

Bias Decomposition. To see what the inflation of the blind null consists of, we model $z_j \sim \mathcal{N}(\mu_j, \sigma_j^2)$ for unwatermarked images, estimate μ_j and σ_j per position, and write the mean of z_{blind}

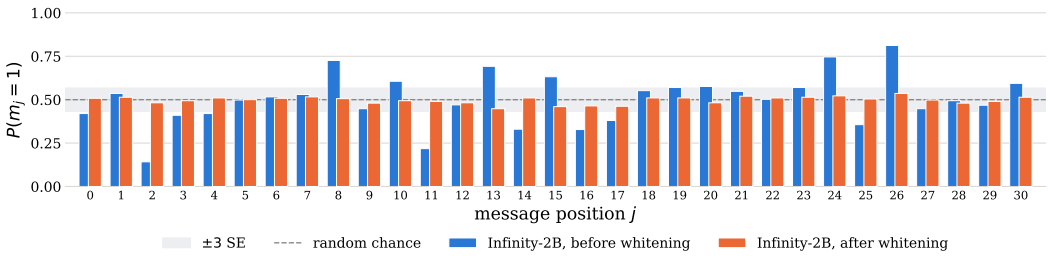


Figure 5: **Per-Position Bias.** Probability of decoding a one at each message position m_j on un-watermarked Infinity-2B generations, with the default decoder and with the corrected decoder of Appx. D.2 fitted on other clean generations.

as

$$\underbrace{\frac{1}{\sqrt{k}}\sum_{j=1}^{k}\sqrt{2/\pi}}_{\text{ideal mean}} + \underbrace{\frac{1}{\sqrt{k}}\sum_{j=1}^{k}(\sigma_j - 1)\sqrt{2/\pi}}_{\text{over-dispersion}} + \underbrace{\frac{1}{\sqrt{k}}\sum_{j=1}^{k}\left(\mathbb{E}|z_j| - \sigma_j\sqrt{2/\pi}\right)}_{\text{structural}}, \tag{10}$$

where $k = R(d-1)$ is the payload length. The first term is the normal mean, under the idealized model of Sec. 3.2. This term is intrinsic to the blind statistic and not removable. The over-dispersion is the excess from $\sigma_j > 1$, which arises because the votes within an image are not independent. The structural bias is the excess from $\mu_j \neq 0$, *i.e.,* the per-position bias of Figure 5.

On a heavy-tailed source the Gaussian model overshoots and the structural term turns negative. We report the decomposition in Table 6. The structural part shrinks with the payload because the bias is a per-token effect and spread over more positions as the payload grows. The structural term is a fixed pattern that can be estimated once on unwatermarked images and subtracted (Appx. D.2).

Table 6: **Decomposition of the Bias.** Mean of the *blind* statistic on un-watermarked images and its decomposition from Eq. (10). Detection is done with Infinity-2B with message length k. We detect on 500 images per row for the generated datasets and 2,500 for natural datasets.

Image Source	k	Observed	$\sqrt{2k/\pi}$	Over-disp.	Structural
Infinity-2B, 1 region	31	7.94	4.44	2.77	0.73
Infinity-2B, 2 regions	62	9.91	6.28	2.99	0.64
Infinity-2B, 4 regions	124	13.19	8.89	3.75	0.55
Infinity-2B, 8 regions	248	17.44	12.57	4.39	0.48
Infinity-2B, 16 regions	496	23.97	17.77	5.74	0.46
BiGR	31	11.04	4.44	8.14	−1.54
Instella-T2I	31	11.23	4.44	4.76	2.03
Infinity-8B	31	7.72	4.44	2.87	0.41
MS-COCO	31	7.74	4.44	2.97	0.33
ImageNet	31	8.30	4.44	4.16	−0.30
LAION	31	10.09	4.44	8.92	−3.27

D.2 Bias Correction via Whitening

The detector of Sec. 3.2 treats the per-position statistics z_j as independent standard-normal variables under the null, which empirical measurement show to be inaccurate (Appx. D.1). We correct this by whitening the decoder(Fernandez et al., 2023), fitted on un-watermarked images. *Centering* removes the structural term of Eq. (10) by subtracting the per-position offset, and *covariance alignment* removes the average over-dispersion and the correlation between positions.

Centering. The detector sums the votes for message position j across all scales before computing z_j, so every scale counts in proportion to its number of tokens and any per-scale offset is absorbed into the summed count. Centering keeps one statistic $z_{s,j}$ per (scale, position), subtracts the unwatermarked

mean $\mu_{s,j}$ of that cell, and combines the scales only afterwards, each with the same weight:

$$u_{s,j} = \frac{z_{s,j} - \mu_{s,j}}{\sigma_s}, \qquad Z_j = \frac{1}{\sqrt{S}} \sum_s u_{s,j}, \tag{11}$$

where $\mu_{s,j}$ and σ_s are estimated on unwatermarked images and S is the number of scales. The extracted message is the sign of Z_j, exactly as in Eq. (5). We weight the scales equally instead of simply accumulating votes across scales. This lowers the voting share of the last scales, which holds 39% of all tokens, and shifts power to intermediate scales, which are more robust under attacks.

Covariance Alignment. While centering fixes the bias at a specific position, it does not impact the correlation between positions. We therefore estimate the covariance matrix Σ of the centered statistics $Z = (Z_1, \ldots, Z_{d-1})^\top$ on un-watermarked images and score the presence of a target message m with

$$z^{\,\mathrm{wht}} = \frac{\boldsymbol{s}^\top \Sigma^{-1} Z}{\sqrt{\boldsymbol{s}^\top \Sigma^{-1} \boldsymbol{s}}}, \qquad \boldsymbol{s} = 2m - 1, \tag{12}$$

the matched filter for correlated noise, where $\boldsymbol{s} = (2m_1 - 1, \ldots, 2m_{d-1} - 1)^\top$. For blind detection, Eq. (12) is evaluated at the decoded message $\hat{m} = \mathrm{sign}(Z)$; its null is not standard normal, and its threshold is calibrated like every other threshold in this paper. Since inverting a 31×31 sample covariance from a few thousand images amplifies estimation noise, we use Ledoit–Wolf shrinkage toward the diagonal (Ledoit & Wolf, 2004) with the data-determined intensity of $\lambda = 0.1$. Covariance alignment cannot change the sign of any Z_j, so the decoded message is that of the centered statistic. We evaluate the whitened decoder against the standard default decoder only in Table 7.

Table 7: Decoder of Sec. 3.2 against the whitened decoder reported with bit accuracy, and blind and oracle TPR@1%FPR on 250 watermarked Infinity-2B images per attack. Mean over the thirteen columns.

		None	Value-metric				Geometric						Regeneration		Mean
			Blur	JPEG	Noise	Color	Rot	Rot(dsk)	Rot+crp	Crop	H-flip	V-flip	VAE	CtrlReg+	
bit accuracy	default	1.00	1.00	1.00	0.98	0.81	0.65	0.94	0.64	0.84	0.87	0.73	1.00	0.80	0.866
	corrected	1.00	1.00	1.00	0.98	0.77	0.68	0.93	0.68	0.88	0.90	0.78	1.00	0.85	0.881
blind TPR@1%FPR	default	1.00	1.00	1.00	0.98	0.99	0.50	0.97	0.26	0.73	0.97	0.52	1.00	0.37	0.791
	corrected	1.00	1.00	1.00	0.98	0.98	0.42	0.97	0.34	0.93	0.98	0.79	1.00	0.66	0.851
oracle TPR@1%FPR	default	1.00	1.00	1.00	1.00	0.92	0.40	0.96	0.32	0.99	1.00	0.83	1.00	0.92	0.874
	corrected	1.00	1.00	1.00	1.00	0.94	0.62	0.98	0.60	1.00	1.00	0.96	1.00	0.98	0.929

D.3 Token Dependence and the Design Effect

The detection excess in Table 6 come from images with large, highly similar or repetitive regions, which quantize to near-identical or complementary tokens that cast the same votes and so look like a watermark to the detector. The correction of Appx. D.2 cannot remove this, because it is fitted once and the excess varies from image to image. What is needed is a per-image estimate of how redundant its votes are. This estimate will by necessity have to be an approximation, as the true value would have to include the covariance of the real world. Ideally the estimate is blind to the watermark, such that it does not penalize the dependence ByteMark creates itself, and gives the same null on every source, so that a threshold set on generated images holds on natural images.

D.3.1 The Textbook Design Effect

Survey statistics (Kish, 1965) corrects a sum of correlated observations by its *design effect*, the factor by which the correlation inflates the variance of the sum relative to independence. For a sum of N' equally weighted votes it is $1 + \bar{\rho}\,(N' - 1)$, where $\bar{\rho}$ is the mean correlation between the votes of two different tokens. We estimate these correlations from the codes. For two tokens p and q that differ in a fraction d_{pq} of their bits, $\rho_{pq} = \frac{d}{d-1}(1 - 2d_{pq})^2 - \frac{1}{d-1}$ is the unbiased estimate of their squared code correlation. It is 1 for identical and for complementary tokens and 0 in expectation for independent random codes. Complementary tokens count as identical because a vote is a transition between adjacent bits, and flipping every bit of a token leaves every transition unchanged. The design effect is

$$\mathrm{DEFF} = 1 + \frac{1}{N'} \sum_{p \neq q} \rho_{pq}, \tag{13}$$

so that N'/DEFF is the effective number of independent votes. The detector divides every z_j by $\sqrt{\mathrm{DEFF}}$ before computing the statistics of Sec. 3.2. Table 8 reports the null before and after the division. Division by the design effect causes the blind detection means to clump around the idealized $\sqrt{2k/\pi} = 4.44$ mean.

The design effect turns out to estimate each image's own mean-square statistic, and the blind statistic depends on little else. We can reformulate $v_{p,j} = 2\,\mathbb{1}[b_{p,j} = b_{p,j-1}] - 1$ for the vote of token p at position j, such that $z_j = N'^{-1/2} \sum_p v_{p,j}$ by Eq. (2). Squaring z_j and averaging over the $d-1$ positions gives, for every image exactly,

$$\frac{1}{d-1}\sum_j z_j^2 \;=\; 1 + \frac{1}{N'}\sum_{p\neq q} c_{pq}, \qquad c_{pq} = \frac{1}{d-1}\sum_j v_{p,j}\, v_{q,j}. \tag{14}$$

Here c_{pq} is the agreement of the votes of tokens p and q. It is 1 if they vote alike at every position and 0 on average if they are unrelated. The right-hand side is the design effect with c_{pq} in place of ρ_{pq}, and the two estimate the same quantity. A vote compares two neighboring bits, so two tokens that differ in a fraction d_{pq} of their bits, spread at random, disagree on a vote with probability $2d_{pq}(1-d_{pq})$. Their expected agreement is then $(1-2d_{pq})^2$, which is the squared code correlation that ρ_{pq} estimates. The design effect is therefore a prediction, made from the codes, of the image's mean-square statistic, and dividing by its square root brings that mean square to about one. For a normal variable the mean absolute value is $\sqrt{2/\pi}$ times the root mean square, so a mean square of one gives a blind statistic of $\sqrt{2k/\pi}$. This about holds, as long as the locations in which p and q differ are spread randomly. Where this is not the case, such as in large repetitive regions, the votes of transitions can agree more or less often than what would be expected if they were.

The division sets the mean square to one, and the mean square is the squared standard deviation plus the squared per-position offset of Appx. D.1. A divisor cannot move a mean, so the offset keeps its share, about a tenth of the mean square, and the standard deviation settles at 0.92 to 0.95 instead of one.

Table 8: **The null of every source before and after DEFF correction.** Mean standard deviation of the per-position statistic z_j over un-watermarked images, and mean blind statistic of Eq. (3), whose value under independent fair-coin votes is $\sqrt{2k/\pi} = 4.44$; raw, and divided by the square root of the design effect over all pairs and of the two restricted estimates of Sec. D.3.3.

		standard deviation of z_j				mean blind statistic			
source	images	raw	all pairs	color region	look-alike	raw	all pairs	color region	look-alike
Infinity-2B generated	2,000	1.64	0.92	1.16	1.12	7.80	4.43	5.53	5.35
MS-COCO	2,500	1.67	0.93	1.12	1.10	7.74	4.47	5.34	5.26
ImageNet	2,500	1.94	0.93	1.13	1.11	8.30	4.44	5.30	5.24
LAION	1,464	3.01	0.95	1.13	1.11	10.09	4.42	5.20	5.11
BiGR-L-512 generated	500	2.83	0.91	–	–	11.04	4.32	–	–
Instella-T2I generated	500	2.07	0.77	–	–	11.23	4.40	–	–
Infinity-8B generated	500	1.65	0.93	–	–	7.72	4.39	–	–

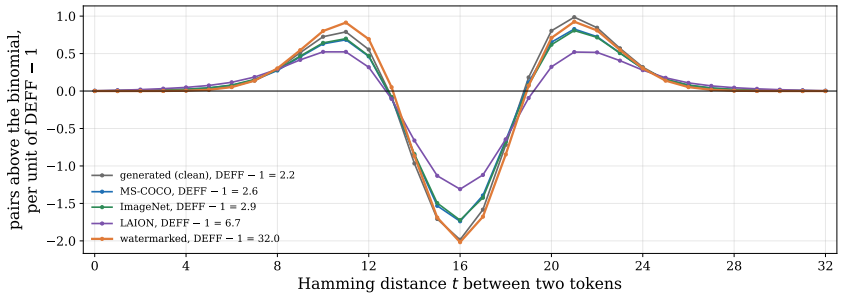
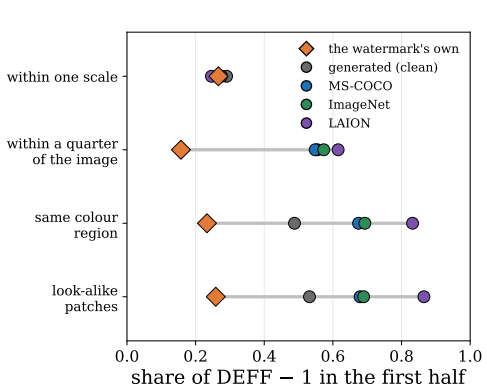
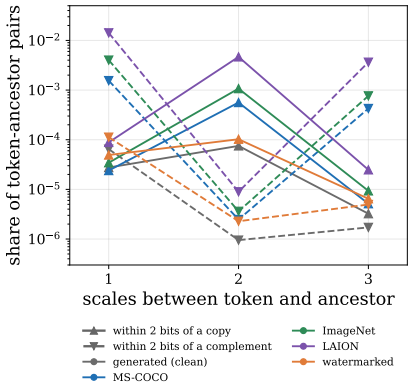


Figure 6: **Where the two kinds of token dependence live.** Excess of token pairs over the binomial of independent codes at every Hamming distance t, scaled by each source's $\mathrm{DEFF} - 1$. Mean over 500 images of each source.



(a) Share of DEFF -1 held by the first half of four ways of splitting the pairs, per clean source and for the watermark's own dependence.



(b) Pairs of a token and its ancestor one to three scales coarser within two bits of a copy (solid) or a complement (dashed).

Figure 7: **How** DEFF $-$ 1 **divides across pair types.** Mean over 500 images of each source.

D.3.2 The Watermark's Own Dependence

The textbook design effect however has one flaw - it isn't blind to the watermark. The same images generated with and without ByteMark enabled have DEFF $= 33.6$ against 3.15 for their clean twins. Since ByteMark biases every token toward the same transition pattern it pushes the hamming distance between random tokens towards the sides. We measure this watermark multiplier across attacks in Table 9a. We additional measure blind detection accuracy's using the textbook estimator, an ideal design effect estimator fully blind to the watermark approximated by using the textbook DEFF measured on the unwatermarked twins to divide the watermarked image scores in table 9b. We find a 8.2% blind detection gap between the textbook and idealized estimator, more than the 4.5% redaction in detection drop including design effect estimation and correction opens. While the watermark multiplier is largest on unattacked images, it is also entirely harmless there with enough margin that even dividing scores by $\sqrt{33.6}$ stays well clear of the detection threshold. The largest drop happens on color which clamps large amount pixels to pure black or white. As this creates the exact repetitive structure design space correction aims to reduce Figure 8.

Table 9: **What the watermark's own dependence costs, on real pairs.** 500 watermarked Infinity-2B generations and their clean twins, the same attack applied to both.

(a) Mean design effect over all pairs, and the watermark multiplier, watermarked over clean, for the Textbook and two

	DEFF, Textbook		watermark multiplier		
attack	clean	watermarked	Textbook	colour region	Look-alike
None	3.15	33.6	10.65	4.39	4.73
Blur	3.25	25.6	7.86	3.43	3.71
JPEG	3.14	21.0	6.69	2.67	3.10
Noise	3.01	7.0	2.32	1.35	1.55
Color	17.29	23.8	1.37	1.36	1.35
Rot	5.01	5.5	1.10	1.02	1.04
Rot(dsk)	3.95	7.8	1.99	1.26	1.33
Rot+crp	3.66	4.2	1.14	1.02	1.05
Crop	3.35	3.7	1.11	1.02	1.04
H-flip	3.14	4.9	1.55	1.16	1.22
V-flip	3.15	4.8	1.52	1.17	1.22
VAE	3.19	18.4	5.75	2.59	2.83
CtrlReg+	3.17	3.6	1.14	1.02	1.05

(b) Blind detection of the default decoder, 3 estimators an an approximation of an ideal estiamtor

	blind TPR@1%FPR, divided by			
attack	none	Textbook	clean twin's	Look-alike
None	1.00	1.00	1.00	1.00
Blur	1.00	1.00	1.00	1.00
JPEG	1.00	1.00	1.00	1.00
Noise	0.98	0.99	0.99	0.98
Color	0.97	0.56	0.71	0.49
Rot	0.41	0.14	0.23	0.17
Rot(dsk)	0.98	0.97	0.98	0.97
Rot+crp	0.24	0.19	0.38	0.28
Crop	0.62	0.76	0.82	0.71
H-flip	0.94	0.92	0.97	0.93
V-flip	0.46	0.31	0.74	0.46
VAE	1.00	1.00	1.00	1.00
CtrlReg+	0.30	0.47	0.59	0.39
mean	0.762	0.716	0.802	0.722
worst nat. FPR	11.7%	1.9%	–	0.8%

D.3.3 Estimates Blind to the Watermark

Aiming to achieve the performance gains of the idealized case we attempt to construct an estimator blind to the watermark. We test the following constructions:

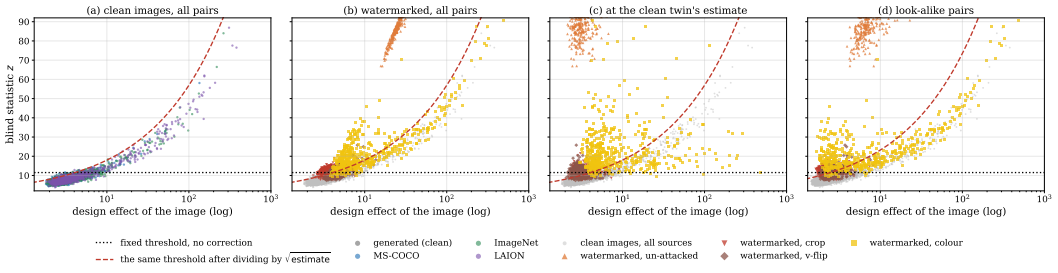


Figure 8: **The watermark's own dependence, image by image.** Design effect of each image (log) against its blind statistic, default decoder. Dotted: the fixed threshold of the main text; dashed: the same threshold after dividing by the square root of the estimate. **(a)** Clean images of the four sources. **(b)** The 500 watermarked twins, all-pairs estimate. **(c)** The same statistics at the design effect of each image's clean twin under the same attack. **(d)** The look-alike estimate; clean images in grey.

(1) The *Odd kernel estimator*: Any weighting of the Hamming distance treats both dependencies alike, because both have the same profile along it. An odd kernel, $f(t) = 1 - 2t/d$, is the natural candidate, since it cancels every lift that is symmetric in t and so ignores the watermark (0.37 on watermarked images against 0.35 on clean ones). However, natural images are symmetric too. The quantiser encodes each scale as the residual of the coarser ones and the sign of that residual alternates, so a token is often the complement of its parent and a copy of its grandparent (Figure 7b). The odd kernel therefore discards the natural dependence along with the watermark's.

(2) The *Distance estimator*: Modeling image dependence as a local effect, while the watermark acts on every token equally, we attempt to exclude most of the watermarks effect by only counting tokens pairs with distances between them of less than a quarter of the image side length.

(3) The *Blind trained estimator*: Forgoing the token space entirely, as it contains the watermarks design space multiplier, we train a regressor to mirror the textbook design space estimate on clean images from 50 pixel statistics.

(4) The *Color region estimator*: Refining the idea behind (3) as repetitive regions can span more than just the immediate vicinity, we follow such regions by creating a 32×32 grid over the encoder input and count two neighboring cells as separated by a color transition when their mean colors differ by more than $\theta = 0.12$. The estimate only counts token pairs which can be connected crossing $\Delta = 0$ transitions.

(5) The *Look-alike estimator*: A simplification of (4) we directly compare the mean red, green and blue values and the pixel standard deviation of the corresponding image patches (the region the token corresponds to in the token grid - for a 64x64 token grid on an 1024 pixel image the 16x16 region) and only count token pairs when all 4 differ by no more than $\theta' = 20\%$.

For (4) and (5) we also consider a smooth weights version, which instead of a binary in- or exclusion of token pairs from design effect assign a weight $[0, 1]$ to each token pair, based on the distance in their metrics. These, as well as the values of θ and θ' are calibrated to a target FPR on a test dataset.

D.3.4 Results

We compare the estimators on the held-out half with the default decoder of the main text, in two ways (Table 10, Figure 9). At the threshold set at 1% on clean generations, which is how the detector would be deployed, we report the false positives on natural images and the detection rate. At equal false positives, with the threshold chosen so that no natural source exceeds 1%, we compare how well each estimator separates watermarked from natural images, independent of where the threshold was set.

Dividing by any design effect is far better than raising the threshold. With no natural source above 1% false positives, the *Textbook* estimator keeps a mean blind detection of 0.69, while the uncorrected detector drops to 0.33. After the division the threshold set on clean generations also holds on natural images, with 1.2% false positives on the worst source against 11.7% without it.

The *Look-alike* and *Color region* estimators do better still. The Look-alike estimator detects more watermarked images than the Textbook one at half its false positives, and both lie above the Textbook curve at equal false positives (Figure 9). Their smooth-weight versions perform within a few points of them.

The other three estimators do not improve on the Textbook one. The *Distance* estimator detects more watermarked images at the generated threshold only because it also lets through more natural images, and at equal false positives it matches the Textbook estimator. The *Odd kernel* and *Blind trained* estimators both underestimate the most redundant natural images, for the reasons given in Sec. D.3.3, and let through several times the nominal rate of false positives. The Blind trained estimator does reach its training goal and assigns watermarked and clean generations the same correction.

For the oracle statistic the estimators differ little, since it suffers far less from natural false positives in the first place.

The textbook estimate costs about a millisecond per image on one CPU thread; the color-region estimates about half a second and the look-alike estimates, which compare every pair of tokens, several seconds.

Figure 9 and Table 10 compare the estimates on the held-out half with the default decoder of the main text, where nothing is fitted but the threshold. Each curve traces one estimate as its threshold moves, against the worst false-positive rate over the three photograph sources; the marker is its operating point at the threshold set at 1% on clean generations.

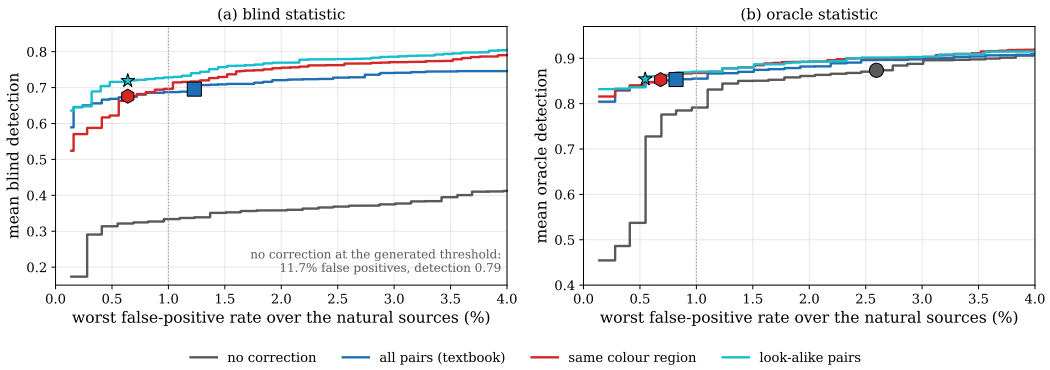


Figure 9: **Restricted ROC plot for design effect** Mean detection over all attacks against the worst false-positive rate over MS-COCO, ImageNet and LAION, held-out half, default decoder, for the blind statistic (a) and the oracle statistic (b). Marked points indicate 1% FPR on clean generated images. Plot restricted to the low FPR regime relevant for watermarking.

Table 10: **Every estimate on the held-out half.** Decoder of Sec. 3.2. At the threshold set at 1% on $1{,}000$ clean generations: the worst false-positive rate over $1{,}250$ MS-COCO, $1{,}250$ ImageNet and 732 LAION images, and the mean blind and oracle TPR@1%FPR over all attacks, 250 watermarked images each.

	threshold on generated				natural FPR $\leq 1\%$	
divisor	worst FPR (blind)	worst FPR (oracle)	blind TPR@1%FPR	oracle TPR@1%FPR	blind	oracle
none	11.7%	2.6%	0.791	0.874	0.334	0.791
all pairs (textbook)	1.2%	0.8%	0.696	0.853	0.687	0.855
Distance	2.9%	1.0%	0.754	0.858	0.694	0.865
Odd kernel	6.6%	1.1%	0.776	0.867	0.586	0.863
Blind trained	4.5%	0.8%	0.754	0.859	0.517	0.864
Color region $\theta = 0.12, \Delta = 0$	0.6%	0.7%	0.676	0.853	0.696	0.868
Color region $\theta = 0.06, \Delta = 2$	1.0%	0.7%	0.697	0.852	0.701	0.868
Color region, smooth weight	0.7%	0.5%	0.681	0.850	0.704	0.868
Look-alike $\theta' = 20\%$	0.6%	0.5%	0.719	0.854	0.728	0.870
Look-alike pairs, smooth weight	0.8%	0.5%	0.691	0.850	0.702	0.871

Decoding is unaffected. The design effect is one scalar per image that rescales all $d - 1$ position statistics alike, so it cannot change the sign of any z_j: the decoded message and every bit-accuracy result in this paper are identical with and without it.

D.3.5 ALL CORRECTIONS TOGETHER

Table 11 applies the steps of Appx. D.2 to the default decoder one at a time, first weighting the scales equally, then centering, then covariance alignment, all fitted on the natural images of the select half. At every stage we report the detector without a design effect and divided by the Textbook and by the Look-alike estimator.

Only equal scale weights and centering change the decoded bits, and together they raise the bit accuracy from 0.866 to 0.881. Covariance alignment and the design effect rescale the statistics without changing their signs. Centering removes the per-position bias of the source it is fitted on. Fitted on natural images, it reduces the skewed positions there from 27 to 9, while those of clean generations stay high.

The corrections overlap. For the oracle statistic, equal scale weights and the design effect do largely the same job. Each alone raises the detection at 1% natural false positives from 0.79 to about 0.86, and together they reach only 0.90. For the blind statistic the design effect remains necessary at every stage. Once all three steps are applied, the threshold set on clean generations also holds on natural images, and the Look-alike estimator gives the highest blind detection. For the oracle statistic the best combination is centering with the Look-alike estimator, and covariance alignment lowers it slightly.

What each variant needs. The textbook design effect needs nothing beyond what the main text already uses: the image under test and the threshold set on clean generations. The restricted estimates add the image's pixels, which the detector has, and one threshold. The corrected decoder needs clean images of the source it is calibrated for: fitted on generations alone, it leaves 7.2% false positives on LAION after the all-pairs division, fitted on natural images 0.8%.

Table 11: **Effect of each correction step.** Starting from the default decoder, we add the steps of Appx. D.2 one at a time and combine each stage with no design effect, the Textbook estimator and the Look-alike estimator. All steps are fitted on natural images from the select half and evaluated on the held-out half. Skewed positions counts the message positions whose decoded bit on clean images differs from 0.5 by more than three standard errors, on $1{,}000$ generated images and on the worst natural source. Bit accuracy and TPR are averaged over all attacks, with 250 watermarked images per attack. The FPR and TPR columns use the threshold set at 1% FPR on $1{,}000$ clean generated images, and the FPR is that of the worst natural source. The last two columns give the best TPR possible while no natural source exceeds 1% FPR.

Decoder	Divisor	Skewed positions		Bit acc.	Worst natural FPR		TPR		TPR at natural FPR $\leq 1\%$	
		generated	natural		blind	oracle	blind	oracle	blind	oracle
Default	None	18	26	0.866	11.7%	2.6%	0.791	0.874	0.334	0.791
	Textbook				1.2%	0.8%	0.696	0.853	0.687	0.855
	Look-alike				0.6%	0.5%	0.719	0.854	0.728	0.870
+ Equal scale weights	None	30	27	0.874	12.0%	2.0%	0.870	0.932	0.340	0.869
	Textbook				1.4%	0.8%	0.712	0.893	0.651	0.898
	Look-alike				1.5%	0.4%	0.782	0.886	0.760	0.911
+ Centering	None	25	9	0.881	11.9%	2.0%	0.846	0.934	0.342	0.877
	Textbook				2.2%	0.8%	0.750	0.899	0.703	0.900
	Look-alike				1.9%	0.4%	0.772	0.894	0.757	0.914
+ Covariance	None	25	9	0.881	10.5%	2.5%	0.851	0.929	0.350	0.892
	Textbook				0.8%	1.3%	0.740	0.898	0.748	0.894
	Look-alike				0.8%	0.6%	0.770	0.894	0.779	0.900

E PAYLOAD EXTENSION

Sec. 3.3 extends the payload beyond one token by partitioning the token grid into R regions that each carry a separate part of the message. This section analyses the design choices behind that extension.

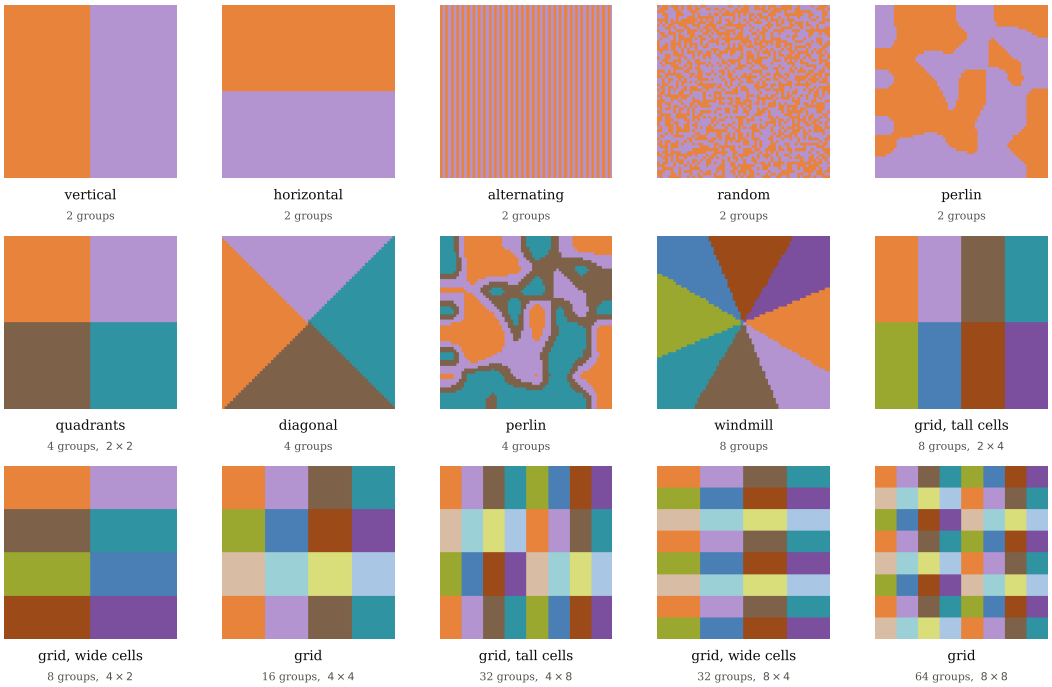


Figure 10: **Different Partitions Used for Expanded Payload.** We use different types of image partitioning depending on the size of the desired payload in our BYTEMARK.

E.1 PARTITION STRATEGIES

A partition in our paper is a fixed, message-independent separation of the token grid into R regions with each region obtaining a label. A message of length k is then split into the R partitions which each carry $d-1$ bits of the message. The information about the partition is shared between watermark embedding and watermark detection. At scales whose token grid is coarser than the partition, each token is assigned to the partition overlapping most of the token. The slight imbalance between partitions this causes at larger number of splits has no measurable impact on the watermark.

We evaluate the partitions shown in Figure 10. Two of them do not follow a direct spatial structure: *alternating* assigns tokens to regions in raster order, and *random* draws each token's label from a secret key. The remaining partitions follow a spatial formation: *vertical* and *horizontal* split the grid along one axis ($R = 2$), *quadrants* and *diagonal* along both axes or both diagonals ($R = 4$), *windmill* into eight sectors ($R = 8$), *perlin* draws a smooth random split by thresholding up-sampled low-resolution noise at its median ($R = 2, 4$). Finally, *grid* tiles the image into $a \times b$ rectangles ($R = 8$ to 64).

E.2 PARTITION CHOICE

We make three observations in Table 12 that motivate the choice of the partitions used in the main paper.

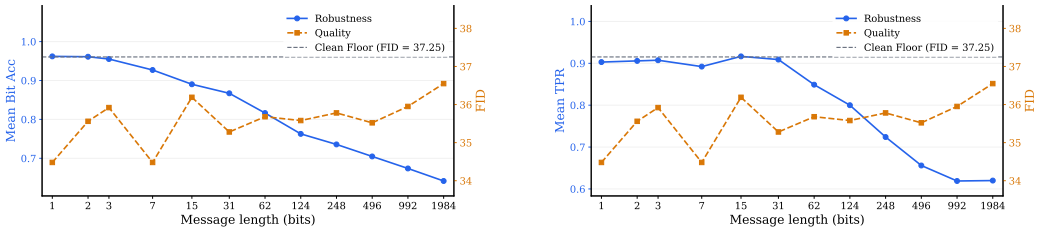
Spatial regions are more robust. At $R = 2$, the two axis-aligned splits reach an average bit accuracy of 0.80, while alternating and random reach 0.77. The strongest difference is due to the geometric attacks, as the geometric transform shifts the token grid and spatial partitions survive this because mostly the tokens along its boundary are impacted.

A partition is stronger against the transforms that map it onto itself. We observe in Table 12 that the vertical partition is more robust against the vertical flip and the horizontal partition more robust against the horizontal flip (highlighted in green). Under these attacks, the partition strategies are mapped onto itself, which keeps the embedded message in the spatial region more consistent.

Table 12: **Robustness for Higher Payload Capacity.** We report the bit accuracy under *blind* detection for each attack, and naive Shannon capacity $k(1 - H(p))$ on clean images (denoted as Capacity). R is the number of partitions and Avg. is the mean bit accuracy over the attacks (excluding None).

R	Partition	None	Capacity	Value-metric				Geometric						Regeneration		Avg.
				Blur	JPEG	Noise	Color	Rot	Rot(dsk)	Rot+crp	Crop	H-flip	V-flip	VAE	CtrlReg+	
1	–	1.000	31.0	1.000	1.000	0.986	0.804	0.644	0.949	0.644	0.844	0.871	0.730	1.000	0.794	0.855
2	alternating	0.999	61.6	0.997	0.991	0.891	0.730	0.581	0.833	0.584	0.689	0.678	0.650	0.991	0.659	0.773
	vertical	1.000	62.0	1.000	1.000	0.965	0.776	0.600	0.922	0.600	0.774	0.575	0.697	1.000	0.714	0.802
	horizontal	1.000	62.0	1.000	0.999	0.955	0.786	0.593	0.922	0.598	0.769	0.798	0.551	1.000	0.713	0.807
	perlin	1.000	61.9	0.999	0.995	0.905	0.732	0.590	0.845	0.588	0.699	0.703	0.628	0.996	0.665	0.779
	random	1.000	61.7	0.997	0.991	0.888	0.727	0.583	0.839	0.587	0.696	0.700	0.630	0.991	0.659	0.774
4	quadrants	1.000	124.0	1.000	0.998	0.914	0.738	0.549	0.861	0.541	0.672	0.522	0.486	1.000	0.636	0.743
	diagonal	1.000	123.9	0.999	0.996	0.904	0.731	0.543	0.852	0.540	0.660	0.611	0.553	0.999	0.632	0.752
	perlin	0.994	117.2	0.983	0.967	0.806	0.651	0.530	0.748	0.518	0.562	0.556	0.543	0.963	0.566	0.699
8	grid 2×4	0.999	245.7	0.997	0.989	0.862	0.704	0.542	0.800	0.543	0.601	0.538	0.503	0.995	0.602	0.723
	grid 4×2	0.999	246.2	0.997	0.987	0.850	0.710	0.547	0.804	0.542	0.601	0.528	0.512	0.995	0.604	0.723
	windmill	0.999	245.2	0.996	0.986	0.851	0.704	0.536	0.802	0.539	0.605	0.549	0.537	0.993	0.600	0.725
16	grid 4×4	0.994	471.1	0.983	0.962	0.794	0.667	0.524	0.743	0.520	0.550	0.519	0.501	0.975	0.573	0.693
32	grid 4×8	0.975	825.3	0.950	0.916	0.730	0.630	0.513	0.692	0.510	0.519	0.502	0.504	0.929	0.546	0.662
	grid 8×4	0.974	819.6	0.949	0.913	0.728	0.628	0.512	0.693	0.511	0.520	0.507	0.500	0.928	0.547	0.661
64	grid 8×8	0.926	1232.1	0.888	0.848	0.673	0.598	0.508	0.646	0.506	0.510	0.504	0.501	0.855	0.532	0.631

Higher capacity reduces the robustness. Increasing R divides the number of tokens voting for each message bit by R, so the per-bit vote shrinks by $\sqrt{R}$ and every bit becomes less reliable. Under blind detection the threshold additionally rises with $\sqrt{k}$, since each position is one more sign the detector selects. Both effects appear in Table 12 as a gradual decline of the average from 0.86 at $R = 1$ to 0.63 at $R = 64$. Image quality is essentially unaffected, because the bias applied to each token is the same for every R and only the message it encodes changes. We provide an analysis of the FID in Figure 11. We additionally embed shorter messages into the model and report the mean bit accuracy and mean TPR in combination with the FID. Shorter messages are embedded by repeating each bit until the token length of 31 is reached, *e.g.,* message 0110 of length 4 is extended to the 31-bit message 00000000 | 11111111 | 11111111 | 0000000. For lengths with remainder, the remainder is added to the first sections, e.g. for bit length 5, the first of the 5 sections repeats the bit 7 times, whereas all other 6. We report the results in Figure 11 and find that shorter messages are more robust with similar impact on quality. The core trade-off is between robustness and capacity, with the robustness decreasing when capacity increases.



(a) Mean Bit Accuracy vs FID vs Payload length

(b) Mean TPR vs FID vs Payload length

Figure 11: **Smaller messages are more robust and all have limited impact on quality.** We present the detection under attacks and FID against natural MS-COCO images.

F Additional Results

F.1 Watermark Robustness

Table 13 extends the bit accuracies of Table 1 with the TPR@1%FPR under *oracle* detection. ByteMark retains a TPR@1%FPR at least 0.77 under CtrlRegen, where every post-hoc baseline falls to near chance.

Table 13: **BYTEMARK is highly robust.** We report TPR@1%FPR under *oracle* detection against each attack. Avg. is the mean over the 10 attack columns (excluding None). Per column, **best** and second best results are highlighted. WAM is omitted: it decodes unwatermarked images to all-zeros, so it has no valid null for this statistic (its bit accuracy is reported in Table 1).

Method	None	Value-metric				Geometric				Regeneration		Avg.
		Blur	JPEG	Noise	Color	Rot	Crop	H-flip	V-flip	VAE	CtrlReg+	
Naive Base - 32 bit	0.99	0.73	0.81	0.35	0.52	0.03	0.02	0.02	0.04	0.42	0.01	0.29
BITMARK- 0 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.96	0.20	**1.00**	**1.00**	0.90	**1.00**	**0.96**	0.90
BYTEMARK- 31 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.94	0.41	0.99	**1.00**	0.83	**1.00**	0.92	**0.91**
BYTEMARK- 62 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.97	0.37	**1.00**	**1.00**	0.20	**1.00**	**0.95**	0.85
BYTEMARK- 124 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.96	0.20	0.95	0.81	0.24	**1.00**	0.84	0.80
BYTEMARK- 248 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.97	0.17	0.84	0.29	0.17	**1.00**	0.80	0.72
BYTEMARK- 496 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.96	0.16	0.54	0.08	0.01	**1.00**	0.81	0.66
BYTEMARK- 992 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.98	0.15	0.24	0.02	0.03	**1.00**	0.77	0.62
BYTEMARK- 1984 bit	**1.00**	**1.00**	**1.00**	**1.00**	**0.99**	0.14	0.17	0.05	0.02	**1.00**	0.83	0.62
InvisMark - 94^2 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.04	0.10	**1.00**	**1.00**	**1.00**	0.94	0.00	0.71
StegaStamp- 100 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.94	0.07	0.02	0.00	0.00	**1.00**	0.56	0.56
TrustMark - 100 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.25	0.04	0.05	**1.00**	0.00	**1.00**	0.01	0.54
VideoSeal - 256 bit	**1.00**	**1.00**	**1.00**	**1.00**	0.81	0.56	0.98	**1.00**	0.01	**1.00**	0.03	0.74
PixelSeal - 256 bit	**1.00**	**1.00**	**1.00**	0.99	0.91	**0.81**	**1.00**	**1.00**	0.00	**1.00**	0.01	0.77
ChunkySeal - 1024 bit	**1.00**	**1.00**	**1.00**	0.93	0.23	0.47	**1.00**	**1.00**	0.00	0.92	0.00	0.66

F.1.1 ROTATION CAN BE CORRECTED

While BYTEMARK is impacted by rotation, we can leverage open rotation estimation tools (Maji & Bose, 2020) to de-skew (dsk) the images from the attack rotation to their original orientation. We report the results when using this tool in Table 14. The results show that using this tool provides near perfect reconstruction of our watermark.

F.1.2 COMPOSITION WITH POST-HOC METHODS

In real-world scenarios, a provider or other user may apply a post-hoc watermark on top of the generated content. To evaluate the impact of this on BYTEMARK, we apply each post-hoc baseline to generated watermarked images and measure the bit accuracy under attacks in Table 15 and Table 16. We find that the post-hoc watermark's own bit accuracy and BYTEMARK accuracy are nearly unimpacted, with the highest losses of up to 8.8% arising when the most perceptible post-hoc marks, StegaStamp (Tancik et al., 2020) and WAM (Sander et al., 2025), are attacked with crop or flips. These are the attacks, where BYTEMARK is already weakest, so the disruptions from the watermark further decrease the robustness.

F.1.3 HYPERPARAMTER ABLATION

Table 17 reports the TPR@1%FPR of BYTEMARK under every attack as the bias strength δ grows. Detection improves with higher δ, but trades against image quality, as we explore in Table 18. The highest gain of robustness can be achieved against the geometric and regeneration attacks.

F.2 IMAGE QUALITY

Across Models. Table 4 reports FID and KID against $5{,}000$ MS-COCO validation images (ImageNet for the class-conditional BiGR), CLIP score, and GE2 (Kamath et al., 2025) for prompt following, for each model with and without BYTEMARK at the payload and δ used in Sec. 4. FID and KID are slightly lower with the watermark on three of the four models. CLIP changes by at most 0.04 and GE2 by at most 2.3 points, on Instella, whose baseline GE2 is also the lowest.

Watermark Strength. Table 18 reports the image quality, when sweeping the bias δ on Infinity-2B. Up to $\delta = 3$ every metric stays within the clean range. At $\delta = 4$ FID more than doubles and GE2

Table 14: **Open source tools enable robustness against rotation.** We report AUC / TPR@1%FPR under the *oracle* detection for the rotation attack and when using an open tool Maji & Bose (2020) to estimate the rotation and reduce it. For BYTEMARK, we report the payload in terms of the number of bits and Mode is the partition. Per column, **best** and second best results are highlighted. We follow the protocol from Table 13.

Method	**Mode**	Rot	Rot + Crp	Rot(dsk)
BYTEMARK- 1984	grid	0.64/0.14	0.65/0.11	0.99/0.98
BYTEMARK- 992	grid 4×8	0.59/0.15	0.55/0.11	0.99/0.98
BYTEMARK- 992	grid 8×4	0.61/0.12	0.59/0.09	0.99/0.98
BYTEMARK- 496	grid	0.61/0.16	0.58/0.12	**1.00/0.99**
BYTEMARK- 248	grid 2×4	0.70/0.19	0.70/0.17	0.99/0.97
BYTEMARK- 248	grid 4×2	0.72/0.26	0.67/0.17	**0.99/0.99**
BYTEMARK- 248	windmill	0.64/0.17	0.67/0.15	**1.00/0.99**
BYTEMARK- 124	diagonal	0.65/0.20	0.62/0.15	0.99/0.98
BYTEMARK- 124	perlin	0.71/0.16	0.65/0.13	0.99/0.98
BYTEMARK- 124	quadrants	0.65/0.22	0.59/0.17	**1.00/0.99**
BYTEMARK- 62	alternating	0.80/0.28	0.79/0.25	**1.00/0.99**
BYTEMARK- 62	vertical	0.79/0.27	0.79/0.27	**1.00/0.99**
BYTEMARK- 62	perlin	0.82/0.32	0.80/0.26	0.99/0.98
BYTEMARK- 62	random	0.80/0.31	0.80/0.27	1.00/0.98
BYTEMARK- 62	horizontal	0.80/0.37	0.77/0.30	**1.00/0.99**
BYTEMARK- 31	–	0.90/0.41	0.90/0.37	**1.00/0.99**
ChunkySeal	–	0.73/0.47	0.58/0.25	**1.00/0.99**
PixelSeal 0.5x	–	0.82/0.74	0.60/0.25	0.68/0.22
PixelSeal 1x	–	0.87/0.81	0.76/0.45	0.97/0.75
PixelSeal 2.5x	–	**0.96/0.82**	**0.92/0.60**	0.99/0.97
VideoSeal 1.0	–	0.78/0.56	0.67/0.34	1.00/0.98
StegaStamp	–	0.54/0.07	0.51/0.07	0.98/0.96
TrustMark-P	–	0.52/0.05	0.57/0.06	0.98/0.95
TrustMark-Q	–	0.49/0.04	0.52/0.05	0.97/0.95
VideoSeal 0.0	–	0.56/0.16	0.54/0.16	0.98/0.97
InvisMark	–	0.59/0.10	0.57/0.08	0.99/0.97

Table 15: **BYTEMARK merges seamlessly with post-hoc watermarks.** We report the difference in bit accuracy under the *blind* detection for detecting BYTEMARK after a post-hoc watermark was applied. Bold values mark a decrease of more than 0.05 against the default.

Method	**None**	**Value-metric**				**Geometric**						**Regeneration**		Mean
		Blur	JPEG	Noise	Color	Rot	Rot(dsk)	Rot+crp	Crop	H-flip	V-flip	VAE	CtrlReg+	
BYTEMARK	1.0000	1.0000	0.9998	0.9856	0.8042	0.6428	0.9494	0.6443	0.8439	0.8707	0.7301	1.0000	0.7937	—
+ TrustMark	+0.0000	−0.0001	−0.0001	−0.0023	−0.0064	−0.0101	−0.0042	+0.0040	−0.0182	−0.0115	+0.0054	+0.0000	−0.0129	−0.0043
+ StegaStamp	+0.0000	+0.0000	+0.0001	−0.0010	+0.0016	−0.0374	−0.0303	−0.0146	**−0.0642**	**−0.0714**	**−0.0556**	−0.0002	−0.0232	−0.0228
+ VideoSeal	+0.0000	+0.0000	+0.0001	−0.0005	−0.0074	−0.0063	−0.0028	+0.0127	−0.0093	−0.0097	+0.0034	+0.0000	−0.0066	−0.0020
+ WAM	−0.0018	−0.0047	−0.0023	−0.0064	−0.0208	−0.0168	−0.0486	−0.0035	**−0.0883**	−0.0406	−0.0320	−0.0043	−0.0043	−0.0211
+ InvisMark	+0.0000	+0.0000	+0.0001	−0.0020	−0.0084	−0.0071	−0.0006	+0.0063	−0.0269	−0.0097	−0.0017	+0.0000	−0.0073	−0.0044

drops by 20 points. We provide visualizations of images generated at different bias strengths in Figure 12.

Message length. Table 19 sweeps the message length from the default 31 to 1984 bits at $\delta = 2$. We find that the image quality is essentially unimpacted by message length. The variation in FID, KID and GE2 is minor. This is expected, since the bias applied to each token is the same for every payload and only the message it encodes is changed. However, since our construction keeps the balance of

Table 16: **BYTEMARK does not disrupt post-hoc watermarks.** We report the difference in bit accuracy for detecting the post-hoc watermarks when applied on top of BYTEMARK-ed images or clean generated images. Bold values mark a decrease of more than 0.01 against the default.

Method	None	Value-metric				Geometric						Regeneration		Mean
		Blur	JPEG	Noise	Color	Rot	Rot(dsk)	Rot+crp	Crop	H-flip	V-flip	VAE	CtrlReg+	
TrustMark	−0.0004	−0.0005	−0.0005	−0.0019	−0.0069	+0.0044	−0.0059	−0.0033	−0.0019	−0.0005	−0.0049	−0.0010	−0.0002	−0.0018
StegaStamp	−0.0004	−0.0002	−0.0004	−0.0006	−0.0067	+0.0018	−0.0038	+0.0008	+0.0009	+0.0012	−0.0006	−0.0002	+0.0020	−0.0005
VideoSeal	−0.0025	−0.0026	−0.0033	+0.0074	−0.0093	**−0.0126**	**−0.0142**	−0.0061	**−0.0354**	−0.0034	+0.0017	−0.0056	+0.0024	−0.0064
WAM	+0.0000	+0.0000	+0.0000	+0.0000	+0.0028	−0.0054	−0.0019	+0.0040	−0.0013	+0.0000	−0.0050	−0.0000	+0.0037	−0.0002
InvisMark	+0.0000	+0.0000	+0.0002	+0.0034	+0.0091	+0.0005	−0.0012	−0.0010	−0.0002	−0.0000	−0.0000	**−0.0251**	+0.0041	−0.0008

Table 17: **Watermark strength on Infinity-2B.** We report TPR@1%FPR under *oracle* detection against each attack for BYTEMARK at a fixed 31-bit payload as the bias strength δ grows. Avg. is the mean over the 10 attack columns. Per column, **best** and second best results are highlighted.

δ	Value-metric				Geometric				Regeneration		Avg.
	Blur	JPEG	Noise	Color	Rot	Crop	H-flip	V-flip	VAE	CtrlReg+	
1	**1.00**	**1.00**	0.98	0.86	0.31	0.83	0.95	0.52	**1.00**	0.61	0.81
2	**1.00**	**1.00**	**1.00**	0.94	0.41	0.99	**1.00**	0.83	**1.00**	0.92	0.91
3	**1.00**	**1.00**	**1.00**	0.93	**0.57**	**1.00**	**1.00**	0.90	**1.00**	0.98	**0.94**
4	**1.00**	**1.00**	**1.00**	**0.95**	0.52	**1.00**	**1.00**	**0.96**	**1.00**	**1.00**	**0.94**

zeros and ones as required for high image quality, the cost of longer payloads is paid entirely in robustness.

F.3 INFERENCE SPEED

Since BYTEMARK changes the sampling process of the generator, we quantify the computational overhead during generation. To this end, we compute the inference speed on a NVIDIA A100 and report the average across 1000 images. Table 20a compares the inference speed of BYTEMARK against BITMARK on the different models. The watermark adds a constant amount of work per generated bit, namely one list lookup and one addition to a logit. Consequently, the overhead is generally small on all models. Most importantly, the difference between BITMARK and our multi-bit BYTEMARK stays low across all four models. Carrying a message is therefore essentially free, compared to carrying only zero-bit information. Similarly we show the detection speed in Table 20b, and for each model the watermarks can be detected in less than one second.

Table 18: **Image quality at different δ.** The metrics follow Table 4 on Infinity-2B at a fixed 31-bit payload. Per column, best is **bold** and second best underlined.

δ	FID ↓	KID $\times 10^3$ ↓	CLIP ↑	GE2 (AM) ↑	GE2 (GM) ↑
0	37.3	13.15	16.48	61.6	15.9
1	36.3	12.63	16.51	**61.9**	15.3
2	**35.3**	11.39	16.52	61.0	15.7
3	36.5	**10.54**	16.52	58.7	**16.8**
4	92.8	59.08	**16.53**	40.6	11.4

Table 19: **Image quality at different message lengths.** The metrics follow Table 4 on Infinity-2B with fixed $\delta = 2$. Per column, best is **bold** and second best underlined.

Payload	FID ↓	KID $\times 10^3$ ↓	CLIP ↑	GE2 (AM) ↑	GE2 (GM) ↑
Clean	37.3	13.15	16.48	61.6	15.9
31 bits	**35.3**	11.39	**16.52**	61.0	15.7
62 bits	35.7	11.37	16.50	60.8	15.6
124 bits	**35.3**	11.62	16.47	61.2	16.2
248 bits	35.8	**11.33**	16.48	**62.1**	16.3
496 bits	35.5	11.58	16.47	61.1	14.4
992 bits	35.9	11.76	16.48	61.3	15.6
1984 bits	36.5	12.20	16.49	**62.1**	**16.6**

Table 20: **BYTEMARK incurs low inference and detection overhead.** Generation and detection costs are measured in seconds per image at batch size 1, with mean ± std. reported over 1000 images.

(a) Generation speed (in seconds per image).

Model	No watermark	BITMARK	BYTEMARK
Infinity-2B	2.669 ± 0.275	3.054 ± 0.278	3.099 ± 0.311
Infinity-8B	4.189 ± 0.258	4.772 ± 0.343	4.791 ± 0.379
BiGR	2.793 ± 0.002	2.856 ± 0.005	2.881 ± 0.004
Instella	3.396 ± 0.005	3.497 ± 0.007	3.488 ± 0.007

(b) Detection speed (in seconds per image).

Model	Re-encode	Scoring	Total
Infinity-2B	0.164 ± 0.021	0.283 ± 0.010	0.447 ± 0.024
Infinity-8B	0.173 ± 0.084	0.487 ± 0.007	0.661 ± 0.085
BiGR	0.080 ± 0.014	0.031 ± 0.000	0.111 ± 0.014
Instella-T2I	0.043 ± 0.053	0.013 ± 0.000	0.056 ± 0.053

G VISUAL EXAMPLES

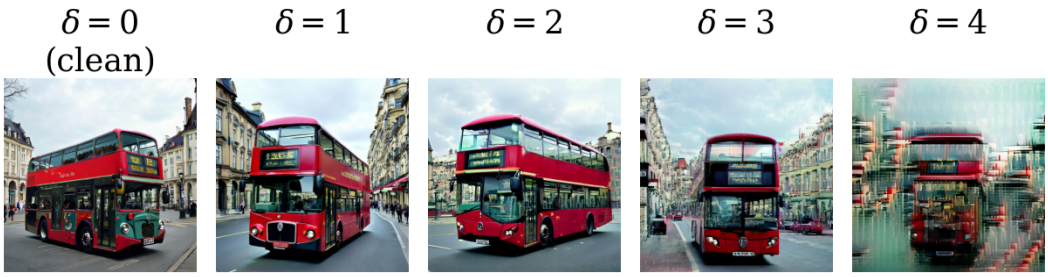


Figure 12: Impact of the bias δ on the image generation quality for BYTEMARK on Infinity-2B. Prompt: "A double decker passenger bus gives an excellent view of the area."



Figure 13: BYTEMARK changes the composition of the image, but keeps quality and semantics unaffected. Prompt: "A very big cute dog running on the beach." and class index "bee" for BiGR.