# On the Difficulty of Constructing a Robust and Publicly-Detectable Watermark

**Jaiden Fairoze** UC Berkeley
**Guillermo Ortiz-Jiménez** Google DeepMind
**Mel Vecerik** Google DeepMind
**Somesh Jha** UW-Madison
**Sven Gowal** Google DeepMind

## Abstract

This work investigates the theoretical boundaries of creating publicly-detectable schemes to enable the provenance of watermarked imagery. Metadata-based approaches like C2PA provide unforgeability and public-detectability. ML techniques offer robust retrieval and watermarking. However, no existing scheme combines robustness, unforgeability, and public-detectability. In this work, we formally define such a scheme and establish its existence. Although theoretically possible, we find that at present, it is intractable to build certain components of our scheme without a leap in deep learning capabilities. We analyze these limitations and propose research directions that need to be addressed before we can practically realize robust and publicly-verifiable provenance.

## 1 INTRODUCTION

What online content is trustworthy? Central to such a question is determining whether a piece of content is authentic. The challenge is more pressing than ever given the widespread availability of Generative AI (GenAI) technology. Powerful models from StabilityAI (Rombach et al., 2022), OpenAI (Achiam et al., 2023), Google DeepMind (Reid et al., 2024), Anthropic (Anthropic, 2024), Meta (Dubey et al., 2024) and Midjourney (Midjourney, 2022) (among others) are able to produce content that can be difficult to distinguish from human-crafted content, even for experts (Ha et al., 2024). This has led to a range of new provenance issues pertaining to trustworthiness, intellectual property, and accountability.

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. 

**Promising Approaches** The main pathways to enabling provenance are metadata-based provenance such as the C2PA standard (Coalition for Content Provenance and Authenticity, 2023), watermarking such as Steg.ai (Steg, 2019), Digimarc (Digimarc, 1995) or SynthID (Google DeepMind, 2023), retrieval such as Turnitin Similarity (Turnitin, 1998), and ML-based detection (e.g., for synthetic content) such as GPTZero (GPTZero, 2023). We consider three key properties of schemes for tracking provenance: unforgeability, robustness, and public-detectability.

*Unforgeability.* A provenance scheme is unforgeable if no adversary can produce content traced to a source without knowledge of that source's secret authentication key. This property is crucial to real-world provenance: content should only be traceable to Alice if Alice enabled traceability with her secret key. Presently, metadata-based provenance (e.g., C2PA) is the only approach that supports unforgeability due to its use of cryptographic digital signatures (Rivest et al., 1978).

*Robustness to accidental stripping.* Traced content is considered robust if it can be traced even if the content has undergone natural transformations during its lifetime. In light of widespread GenAI tools, the primary application of provenance tools is to enable online content traceability. In this setting, robustness is key: content such as text, audio, or images cannot be expected to retain its original form after initial distribution. Watermarking and retrieval mechanisms currently enable transformation-robust traceability.

*Public-detectability.* Public-detectability separates the authentication and verification functionalities. Entities that hold a secret key can authenticate content such that verification can be performed with a corresponding public key. The public key can be used by *anyone* to verify that content originated with the secret key holder. In the special case where robustness is not required, cryptographic digital signatures provide this exact functionality. Metadata-based provenance is publicly-detectable due to its use of cryptographic signatures, but it is not robust to accidental stripping.

**This Work** In this paper, we study the possibility of uniting the high-security and trustworthiness of cryptographic tools with the powerful robustness of deep learning-based provenance. In particular, we ask:

*Is it possible to design an image watermark that (a) preserves the robustness of deep watermarks and (b) meets a well-defined notion of unforgeability and public-detectability?*

We analyze the theoretical and practical feasibility of constructing an image watermark with the following properties:

- *Cryptographic unforgeability.* It should be computationally infeasible to generate adversarial content watermarked with a key that the adversary does not control. This should hold even if the adversary has full information minus the secret key.
- *Robustness to accidental stripping.* The watermark should persist even if the image is naturally transformed. This property ignores the adversarial setting and only needs to hold (on average) over naturally-occurring transformations.
- *Publicly-detectable.* The detection procedure should not contain any secret information—there should be no detriment to publicly releasing it and allowing anyone unlimited access.
- *Quality-preserving.* Watermarked images should be of similar quality to the original image.

While we focus on images, our results apply to any high-entropy data that supports post-hoc watermarking and robust embeddings, e.g., audio and video.

**Main Contributions** Our core contributions are:

1. In Section 4, we present a watermarking scheme that is provably unforgeable and publicly-detectable, but with limited robustness. This scheme is similar to metadata-based provenance but without metadata—the watermarked image is the same size as the input image.
2. In Section 5, we define the requirements for a robust, unforgeable, and publicly-detectable watermark. We prove that it can be constructed using cryptographic signatures, post-hoc watermarks, and robust embeddings as building blocks. In particular, the resulting public watermark is robust to transformations that underlying post-hoc watermark and robust embedding support.
3. In Section 6, we study the barriers to deploying our robust scheme. We find that state-of-the-art image embedding models are vulnerable to adversarial attacks that can force embeddings to collide. Despite this, we observe a weak-but-significant correlation between resistance to adversarial attacks and model performance. This suggests that if future models are able to better capture human vision, they may enjoy intrinsic adversarial robustness, thereby enabling robust and publicly-detectable watermarking.

## 2 RELATED WORK

We cover essential related work below. For wider coverage, see Supplement A. For a visual overview of the main approaches to content provenance, see Figure 1.

**Metadata-Based Provenance** Throughout the paper, we use metadata-based provenance to refer to strategies that attach signatures as additional metadata (See Figure 1). The C2PA (Coalition for Content Provenance and Authenticity, 2023) standard is the current leading approach. The attached metadata is referred to as a manifest. This manifest contains cryptographic information attesting to the content's creation, modification, and distribution history. The manifest is "hard bound" to the content using a cryptographic hash—even a one bit change in content would cause its hash to change and detach the manifest from its content. Both the content and manifest must be preserved in order for verification to succeed.

Metadata-based provenance can, in general, provide a comprehensive record of the content's history. Due to its use of standardized cryptographic primitives, the system achieves a strong and well-defined notion of security. Moreover, verifying a C2PA manifest is *fully public*—anyone can verify the authenticity of a C2PA manifest by leveraging existing public-key infrastructure (Laurie, 2014).

Conversely, the link between manifest and content is weak: it is trivial to detach the corresponding manifest from any arbitrary content. Adversarial detachment aside, existing web infrastructure cannot readily support the manifest. Updating infrastructure to accommodate C2PA manifests is a time-consuming and costly endeavor. To partially address this problem, the C2PA working group is considering a *soft binding* extension where metadata can be re-attached to content using a perceptual hash computed from the content or a watermark embedded within the digital content. The group has referenced a candidate algorithm (Pan, Titusz, 2022), but any method that enables a "similarity comparison" between content is plausible.

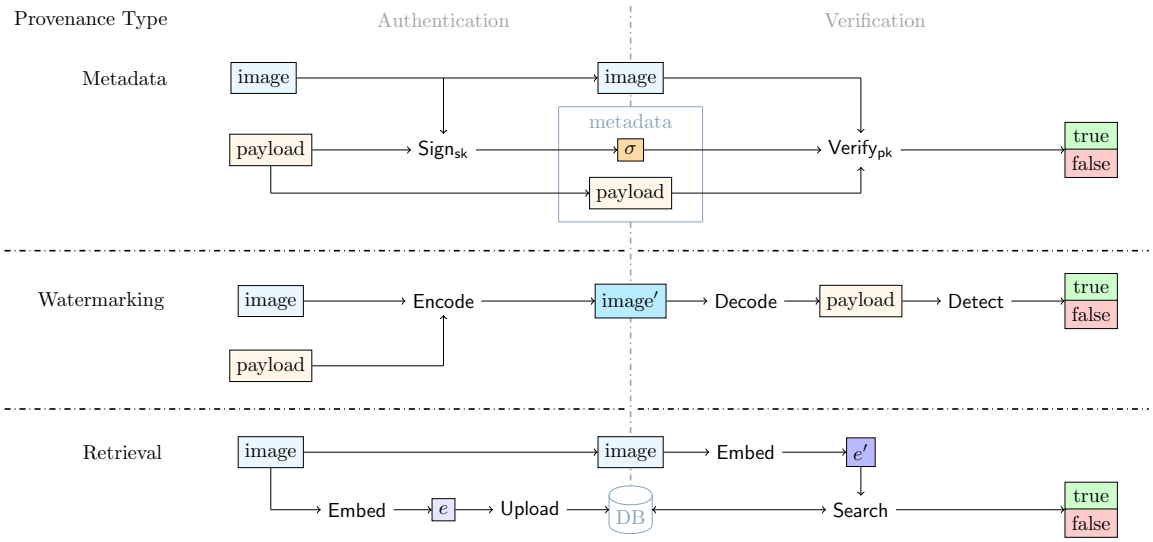


Figure 1: The three main approaches to content provenance. Metadata-based provenance (top) uses an auxiliary manifest to attach a cryptographic signature and other metadata to the image—signature authentication yields provenance. Watermarking (middle) encodes a payload with provenance information directly into the image itself, and the payload can be decoded thereafter. Retrieval-based detection (bottom) maintains a global store of image embeddings where the store is queried to check if a candidate image is known.

**Watermarking** Fundamentally, watermarking schemes aim to hide information within content itself without visibly perturbing the content.[1] The encoded information can enable provenance: in practice, the payload is usually a unique identifier for external information retrieval or, if the watermark capacity is large, a data store for origin-related information.

Common among watermarking schemes (see Supplement A) is their optimization to resist content modifications—the watermark payload should not be destroyed if transformed content is "reasonably similar" to the original watermarked image. In addition, the watermark meets a high degree of imperceptibility: to an untrained eye, an image and its watermarked counterpart are of the same quality. Since the watermark embeds information into the image itself rather than additional metadata, it does not require any web infrastructure changes and can be dropped in to enable provenance immediately. However, such systems are subject to the following concerns:

*Security.* There is no guarantee that proprietary algorithms are secure or correct, so end users cannot contextualize detection results. In the text setting, post-hoc detectors commonly flag human-generated text as AI-generated, directly harming individuals (Gegg-Harrison and Quarterman, 2024).

*Utility.* All known industry watermarks only permit trusted users to plant or detect watermarks—the watermark provider cannot release the detector or perform watermarking client-side as it weakens security.

**Retrieval-Based Detection** The most straightforward approach to provenance is to maintain a large, continuously-updated database containing every AI-generated image. This solution is problematic when (a) different model providers do not have unified storage or (b) scalability issues arise once the database reaches a critical size. Other issues (such as privacy) can be partially ameliorated by using *fingerprints*: instead of storing images directly, a succinct and robust representation of each image (a fingerprint) is stored that preserves the ability to measure similarity.

Similarity comparisons arise in many areas of computer science beyond content fingerprinting (Seo et al., 2004), such as perceptual hashing (Indyk and Motwani, 1998), copy detection (Chen et al., 2020), and fuzzy matching (Chaudhuri et al., 2003). In this work, we group these techniques under the umbrella of a "robust embedding." In practice, robust embeddings are deployed for explicit material detection. Examples are detection of non-consensual intimate image abuse (StopNCII, 2024), online terrorism (Saltman and Thorley, 2021), and child sexual abuse material (CSAM) (Apple, 2021; Prokos et al., 2023).

[1] We do not consider "visible" watermarking schemes as they alter content and are easily strippable.

# 3 PRELIMINARIES

We cover basic notation before presenting our schemes. We also provide a succinct definition of cryptographic signatures as they are used throughout.

**Notation** Let $\lambda$ be the security parameter, i.e., the target level of security. A system targeting $\lambda$ bits of security should be resistant to any attack that runs in at most $2^\lambda$ steps. Let $\epsilon$ be a small error tolerance. We will use $\epsilon$ to capture failure rates for various schemes. Let $\mathsf{poly}(\cdot)$ refer to any arbitrary polynomial. Define $\mathsf{negl}(\lambda)$ to be a function such that for all $\mathsf{poly}(\lambda)$, it holds that $\mathsf{negl}(\lambda) < \frac{1}{\mathsf{poly}(\lambda)}$ for all sufficiently large $\lambda$. We use superscripts to denote oracle access. For example, $A^O$ denotes that algorithm $A$ has oracle access to oracle $O$.

**Image Transformations** Let $\mathcal{T}$ be the set of all possible transformation functions that can apply to an image. We use $\Gamma(x)$ as the set of all transformations of $x$ and similarly $\Gamma(X)$ as the union of all sets of transformations of each element $\bigcup_{x\in X}\Gamma(x)$.

## 3.1 Cryptographic Digital Signatures

Given a secret key $sk$ and a public key $pk$ from a generation function $\mathsf{Generate}(1^\lambda)$[2], a signature $\sigma$ can be generated from content $x$ using the secret key: $\sigma \leftarrow \mathsf{Sign}(sk, x)$. This signature is verified by computing $\{\mathtt{true}, \mathtt{false}\} \leftarrow \mathsf{Verify}(pk, x, \sigma)$. The signature scheme must be *correct* in the sense that honestly-generated signatures must verify with overwhelming probability. That is, for all $x$,

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(pk, x, \mathsf{Sign}(sk, x)) = \mathtt{true} : \\ (sk, pk) \leftarrow \mathsf{Generate}(1^\lambda)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Additionally, the scheme must satisfy a notion of *unforgeability*[3], meaning for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ and message $x$,

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(pk, x^*, \sigma^*) = \mathtt{true} \\ \wedge\ x^* \neq x : \\ (sk, pk) \leftarrow \mathsf{Generate}(1^\lambda) \\ \sigma \leftarrow \mathsf{Sign}(sk, x) \\ (x^*, \sigma^*) \leftarrow \mathcal{A}(pk, x, \sigma)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

That is, adversary $\mathcal{A}$ accepts its input (the public key $pk$, an honest message $x$, and a digital signature of the message $\sigma$). Its goal is to produce a pair $(x^*, \sigma^*)$ that "break security" such that (a) the pair is authentic (i.e., verification checks out) and (b) $x^*$ is a different message to the given $x$. If it succeeds, the adversary has forged a signature on a new message without the secret signing key.

[2]The security parameter $\lambda$ is passed as base-1 so that the time complexity of algorithm $\mathsf{Generate}$ is polynomial in the size of the input. For reference, see Chapter 3.1.1 in Katz and Lindell (2014).

[3]For most definitions, we use a weaker notion where the adversary does not have oracle access to relevant functions—this suffices for our purpose. We present the stronger versions in Supplement B.

# 4 WARMUP: AN UNFORGEABLE AND PUBLICLY-DETECTABLE WATERMARK

Our first goal is to obtain a non-robust but unforgeable and publicly-detectable watermark. We ask:

*Is it possible to obtain a scheme analogous to metadata-based provenance for images that does not introduce additional metadata?*

We present a simple scheme that embeds a cryptographic signature within an image $x$ such that there is a natural hash function satisfying $\mathsf{Hash}(x) = \mathsf{Hash}(x')$ where $x'$ is a visually-identical version of $x$ that embeds a cryptographic signature. The scheme satisfies the equality by encoding signature bits into low-order bits of the image (Muyco and Hernandez, 2019). Image quality is guaranteed as pixel-values cannot change by more than 1, i.e., the PSNR is at worst $\approx 48.13$. We note that the hash function is strongly collision-resistant for natural images: the hash of two visually-different images will not be the same. We define our scheme in Figure 2.

**Theorem 4.1** (Informal)**.** The scheme presented in Figure 2 is correct if the underlying cryptographic signature scheme is correct.

*Proof.* We claim that $\mathsf{Detect}(pk, \mathsf{Watermark}(sk, x))$ holds for all but negligibly few choices of $x$ and $sk, pk \leftarrow \mathsf{Generate}(1^\lambda)$. First, the underlying signature scheme is correct, meaning $\mathsf{Verify}(pk, h, \mathsf{Sign}(sk, h))$ holds for almost all inputs $h$. Second, observe that the watermarking algorithm plants each bit of the signature into the lowest order integer bit of each color channel. At detection time, this bit is extracted by computing $x_{i,j,c} \mod 2$ of each pixel channel. □

**Theorem 4.2** (Informal)**.** The scheme presented in Figure 2 is unforgeable if the underlying cryptographic signature scheme is unforgeable.

*Proof.* To forge a watermark, it must be that either the signature scheme itself is forgeable, or the hash function is not collision-resistant. It is given that the signature scheme is unforgeable, so it remains to see that the hash function is collision-resistant—for any two natural images, it should be negligibly likely that

**function** $\mathsf{Watermark}(sk, x)$
  $\sigma \leftarrow \mathsf{Sign}(sk, \mathsf{Hash}(x))$
  **for** $x_{i,j,c}$ in $x$ **do**
    $x'_{i,j,c} \leftarrow 2 \cdot \lfloor \frac{x_{i,j,c}}{2} \rfloor + \sigma_{i,j,c}$
  **return** $x'$

**function** $\mathsf{Detect}(pk, x)$
  $h, \sigma \leftarrow \mathsf{Hash}(x), \emptyset$
  **for** $x_{i,j,c}$ in $x$ **do**
    $\sigma \leftarrow \sigma \parallel x_{i,j,c} \mod 2$
  **return** $\mathsf{Verify}(pk, h, \sigma)$

**function** $\mathsf{Hash}(x)$
  $h \leftarrow \emptyset$
  **for** $(r, g, b)$ in $x$ **do**
    $h \leftarrow h \parallel (\lfloor r/2 \rfloor, \lfloor g/2 \rfloor, \lfloor b/2 \rfloor)$
  **return** $h$

Figure 2: Specification of our unforgeable and publicly-detectable watermark. The keys are generated with the generation function of the signature scheme, $sk, pk \leftarrow \mathsf{Generate}(1^\lambda)$. WLOG, the input image $x$ is RGB-encoded. The Watermark encodes a signature of the image within the image itself such that the output of Hash does not change. This is achieved by encoding signature bits in the least significant bit of each color channel value—when the hash is applied (i.e., each value is divided by two and floored), its value must be the same as the plain image. Thus, Detect is able to recover both the hash value and signature bits in order to verify the signature.

the hash of the images are the same. This indeed holds: given any natural image $x \in \{0, \dots, 255\}^n$, the $\ell_\infty$-norm ball with step 1 *must* be visually the same image. In other words, changing all channel values by at most 1 cannot visually change an image (we focus on images that show clear distinguishable semantic content). □

# 5 A ROBUST AND PUBLICLY-DETECTABLE WATERMARK

Next, we augment our base scheme to add robustness. In general, we rely on ML-based tools for robustness and cryptographic signatures for unforgeability and public-detectability. We first define the ML-based tools, post-hoc watermarks and robust embeddings.

## 5.1 Post-Hoc Watermarking

We treat post-hoc watermarks as a communication channel where the image is the channel and the watermark payload is the communicated data. For this purpose, we only require a notion of *correctness*: the decoded payload should be the same payload that was encoded—the worst attack that an adversary can launch is to destroy the payload. We define post-hoc watermarks to satisfy the following interface: $\mathsf{Generate}(1^\lambda, \mathcal{T}) \to \mathsf{Encode}, \mathsf{Decode}$ is a possibly randomized algorithm that produces two functions, $\mathsf{Encode}(x, m) \to x'$ and $\mathsf{Decode}(x') \to m$, satisfying the following. Let $x$ be an image and $m \in \{0,1\}^c$ be a $c$-length binary message (where $c$ represents the watermark capacity). Then, for all choices of image $x$, message $m$, and transformation $T \in \mathcal{T}$,

$$\Pr\left[\begin{array}{c}\mathsf{Decode}(T(\mathsf{Encode}(x,m))) = m : \\ \mathsf{Encode}, \mathsf{Decode} \leftarrow \mathsf{Generate}(1^\lambda, \mathcal{T})\end{array}\right] \geq 1 - \epsilon.$$

That is, the payload is recovered with high probability.

## 5.2 Robust Embedding Functions

This primitive captures various forms of similarity search (see Supplement A.2 for details). A robust embedding provides a possibly-randomized generation procedure $\mathsf{Generate}(1^\lambda, \mathcal{T}) \to \mathsf{Embed}, \mathsf{Compare}$ that yields two functions, Embed and Compare, with respect to the set of transformations $\mathcal{T}$. Given an image $x$, $e \leftarrow \mathsf{Embed}(x)$ produces a succinct embedding $e$. In order to compare the similarity of two images $x$ and $y$, it suffices to compare them in the embedding space by checking $\{\texttt{true}, \texttt{false}\} \leftarrow \mathsf{Compare}(\mathsf{Embed}(x), \mathsf{Embed}(y))$ where the output of Compare is a binary value representing similarity. Like other primitives, we require that embedding comparisons are correct: Compare should output `true` when the input embeddings were produced from visually similar images. For all $x$ and transformations $T \in \mathcal{T}$,

$$\Pr\left[\begin{array}{c}\mathsf{Compare}(e_1, e_2) = \texttt{true} : \\ \mathsf{Embed}, \mathsf{Compare} \leftarrow \mathsf{Generate}(1^\lambda, \mathcal{T}) \\ e_1 \leftarrow \mathsf{Embed}(x) \\ e_2 \leftarrow \mathsf{Embed}(T(x))\end{array}\right] \geq 1 - \epsilon.$$

Analogous to a cryptographic hash function, a robust embedding should satisfy a (weakened) notion of collision resistance where collisions are permitted if the input images are valid transformations as determined by the set of possible transformations $\mathcal{T}$. We define collision resistance for robust embeddings as follows. For any choice of $x$, it must be that

$$\Pr\left[\begin{array}{c}\mathsf{Compare}(\mathsf{Embed}(x), \mathsf{Embed}(x^*)) = \texttt{true} \\ \wedge\ x^* \notin \Gamma(x) : \\ \mathsf{Embed} \leftarrow \mathsf{Generate}(1^\lambda, \mathcal{T}) \\ e \leftarrow \mathsf{Embed}(x) \\ x^* \leftarrow \mathcal{A}(\mathcal{T}, x, e)\end{array}\right] \leq \epsilon.$$

Note that a robust embedding is only useful if the embedding size is much smaller than the image size. If not, it would be more efficient to compute similarity directly on the images.

## 5.3 Construction

We define a watermarking scheme that is simultaneously publicly-detectable, unforgeable and robust. See Figure 3 for a graphical overview of our method.

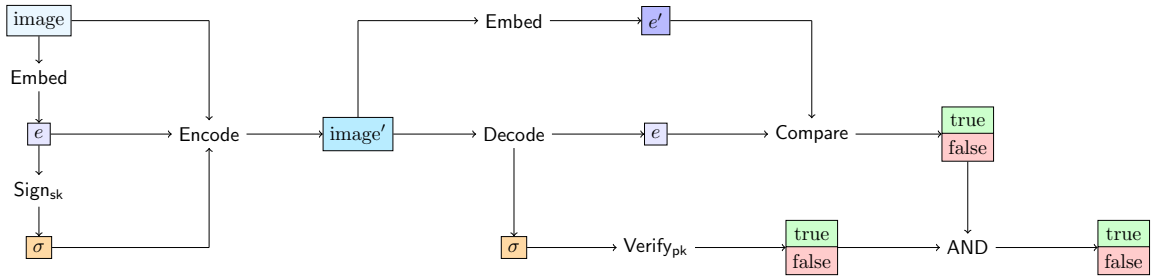


Figure 3: A robust and publicly-detectable watermark built from a cryptographic signature scheme, a post-hoc watermarking scheme, and a robust embedding model for images. Using a post-hoc watermark, the scheme encodes an embedding of the image along with a signature of the embedding within the image itself. This information can be decoded and verified thereafter.

**Approach** From our base scheme, we incrementally build our final scheme with all desired properties. First, we replace the pixel-level encoding procedure of our initial scheme with a post-hoc watermarking scheme: instead of hiding payload bits in the lower order bits of an image, we use a post-hoc watermark to plant the payload. To watermark, we compute $x' \leftarrow \mathsf{Encode}(x, \mathsf{Sign}(sk, x))$ and to verify the watermark we compute $b \leftarrow \mathsf{Verify}(pk, x', \mathsf{Decode}(x'))$. Unfortunately, this scheme does not work as-is: since $x' \neq x$, the signature is computed on a different image to the one produced by the post-hoc watermark[4].

Instead of signing the image directly, we embed it $e \leftarrow \mathsf{Embed}(x)$ and sign the embedding: $\sigma \leftarrow \mathsf{Sign}(sk, e)$. Then, we encode the signature and embedding into the image: $x' \leftarrow \mathsf{Encode}(x, \sigma \parallel e)$. To verify the watermark, we decode $\sigma$ and $e$ from the candidate watermarked image by parsing $\sigma, e \leftarrow \mathsf{Decode}(x')$. We also require the embedding of the candidate image: $e' \leftarrow \mathsf{Embed}(x')$. For the candidate image to be considered watermarked by $sk$, two conditions must be met. First, $\mathsf{Compare}(e, e') = \texttt{true}$: the watermarked image is perceptually similar to the image corresponding to the signed embedding. Second, $\mathsf{Verify}(pk, e, \sigma) = \texttt{true}$: the signature must be authentic.

### 5.4 Properties

**Threat Model** The key difference between publicly- and privately-detectable watermarks arises at detection time: in the public setting, Detect takes a public key rather than the secret key used at watermarking time. The adversary also has full details of the scheme except for the secret key: this includes white-box access to Sign and Verify from the signature scheme, Encode and Decode from the post-hoc watermark, and Embed and Compare from the robust embedding. For security, an adversary should not be able to forge a watermark corresponding to a secret key out of her control. That is, for all $x$,

$$\Pr\left[\begin{array}{c}\mathsf{Detect}(pk, x^*) = \texttt{true} \\ \wedge\ x^* \neq x' : \\ (sk, pk) \leftarrow \mathsf{Generate}(1^\lambda) \\ x' \leftarrow \mathsf{Watermark}(sk, x) \\ (x^*) \leftarrow \mathcal{A}(pk, x')\end{array}\right] \leq \epsilon.$$

**Theorem 5.1.** *If $(\mathcal{T}_{\mathsf{REF}}, m, n, \epsilon_{\mathsf{REF}})$-robust embedding functions, $(\mathcal{T}_{\mathsf{PGWS}}, c, \epsilon_{\mathsf{PGWS}})$-post-hoc watermarking schemes, and $(\delta, \lambda)$-cryptographic signatures exist such that $c \geq \delta + n$ and $\mathsf{PGWS.Encode} \in \mathcal{T}_{\mathsf{REF}}$, then $(\mathcal{T}_{\mathsf{REF}} \cap \mathcal{T}_{\mathsf{PGWS}}, \epsilon_{\mathsf{REF}} + \epsilon_{\mathsf{PGWS}} + \mathsf{negl}(\lambda))$-publicly-detectable watermarking schemes also exist.*

For the full proof of Theorem 5.1, see Supplement B.2. Given secure building blocks (i.e., robust embedding, post-hoc watermark, and signature scheme), the resulting watermark inherits key properties from each relevant underlying primitive. We provide an overview below:

**Robustness** The robustness of the resulting scheme is the set of transformations common between the robust embedding and the post-hoc watermark. For the watermark to be detectable, the data encoded in the private watermark and the robust embedding of the image need to be preserved.

**Unforgeability** We outline the high-level intuition for the unforgeability of our scheme. Imagine the definition does not hold, and it is computationally tractable to find a forged $x^*$ for any input image $x$. Given the forgery $x^*$ satisfying $\mathsf{Verify}(pk, e, \sigma) \wedge \mathsf{Compare}(e, e') = \texttt{true}$, the forged message-signature pair $(e, \sigma)$ or colliding embeddings $(e, e')$ are immediately recoverable, implying attacks against the underlying primitives: either (a) the underlying cryptographic signature scheme is forgeable, or (b) the underlying robust embedding is not collision-resistant.

[4]We discuss training a robust embedding such that $\mathsf{Embed}(x) = \mathsf{Embed}(x')$ in Section 6.3.

Since we assume that such primitives are secure to begin with, we reach a contradiction implying that the watermark is indeed unforgeable.

**Imperceptibility** Imperceptibility is an intrinsic property of the (underlying) watermarking scheme—we demonstrate how to use an existing private watermarking scheme to instantiate a publicly-detectable one. For example, if we use the TrustMark-Q (Bui et al., 2023a) watermark to instantiate our robust and public watermark, the scheme would inherit the PSNR ($43.26 \pm 1.59$) and SSIM ($0.99 \pm 0.00$) of TrustMark-Q directly. We further note that there is a tradeoff between watermark robustness or capacity and imperceptibility: watermarks with better robustness and/or capacity tend to introduce more distortions.

# 6 INSTANTIATING OUR SCHEME IN THE REAL WORLD

We analyze the feasibility of instantiating each building block of our construction in the real world.

## 6.1 (Compact) Cryptographic Signatures

Standard cryptographic signature schemes such as RSA (Rivest et al., 1978) or ECDSA (Johnson et al., 2001) are secure, efficient, and widely supported. For use in a robust and publicly-detectable watermark, we require compact signature schemes with short signature lengths. For standard 128-bit security, RSA signatures are at least 3072 bits (Elaine et al., 2016), and ECDSA signatures are at least 512 bits (on the secp256r1 elliptic curve; Certicom Research, 2010). BLS signatures may offer better compactness (Boneh et al., 2001) though require stronger assumptions and careful choice of pairing-friendly curves. For example, the most widely-used curve, BLS12-381 (Barreto et al., 2003), supports a minimum signature size of 384 bits for 128-bit security. In our setting, it is reasonable to target lower security as breaking even 80-bit security is expected to be orders of magnitude more difficult than launching attacks on the robust embedding. This would require future work on suitable elliptic curves.

**Takeaway** Barring a leap in compact digital signature design, signature sizes are unlikely to shorten drastically in the foreseeable future.

## 6.2 Realizing a Robust Embedding Function

We evaluate a range of self-supervised image embedding models for their suitability to instantiate our robust embedding function. We ask:

*How collision-resistant are state-of-the-art image embedding models in a white-box setting?*

**Methodology** For a range of state-of-the-art embedding models (see Table 1), we instantiate a robust embedding function: (a) the embedding function is simply the model's native forward pass, and (b) embeddings are compared by computing their $\ell_2$-normalized dot product. Note that we do not binarize comparison values in order to capture fine-grained performance. See Supplement C for additional experimental data.

**Data** We use the "original" and "strong" components of the Copydays dataset (Douze et al., 2009) for copy detection. This provides base images and their strongly-transformed variants. For all base images, we randomly select a positive and negative image which respectively represent a transformed version of the original image and a completely different image. We denote the similar pair as a *positive* pair and the different pair as a *negative* pair. Thus, let $(I_{1,a}, I_{1,a'}), (I_{2,a}, I_{2,a'}), \ldots, (I_{1,a}, I_{1,b}), (I_{2,a}, I_{2,b}), \ldots \in \mathcal{D}$ be the dataset $\mathcal{D}$ of both positive $(I_{i,a}, I_{i,a'})$ and negative $(I_{i,a}, I_{i,b})$ image pairs.

**Attack** We fix a baseline projected gradient descent (PGD) (Madry et al., 2018) with momentum attack (20 steps) for various choices of $L_p$ epsilon values for $p \in \{1, \infty\}$. The attack is designed to force the $\ell_2$-normalized dot product of embedding pairs to be far (close) for positive (negative) pairs. If an adversary has access to the image embedding, then using adversarial attacks, they can forge a watermark by forcing two different images to have a high similarity. For example, if an adversary wants to use PGD and they have access to the embedding, they can perform gradient ascent over the input space up to some $\ell_p$ imperceptible norm bound.

**Evaluation** We measure the following:

1. *Clean performance.* We calculate the area under the receiver-operating characteristic curve (ROC AUC) of the binary classification problem captured by $\mathcal{D}$. We compute the score for a given pair with $\mathsf{Compare}(\mathsf{Embed}(I_{i,\cdot}), \mathsf{Embed}(I_{i,\cdot}))$ and assign a binary label depending on if the pair is positive or negative. This captures raw model performance as a robust embedding function.

2. *Attacked performance.* The dataset $\mathcal{D}$ is attacked with fixed PGD attacks for various perturbation strengths such that positive (negative) pairs are embedded far (close) in the embedding space. ROC AUC is calculated as in the clean case but on

the attacked dataset $\mathcal{A}_{\ell_p,\epsilon}(\mathcal{D})$ for $p \in \{1, \infty\}$ and $\epsilon \in \{1/255, 2/255, 4/255, 8/255, 16/255, 32/255\}$.

**Findings** We find current embedding models are well-suited to similarity search in the absence of an adversary: the best performing models, DINOv2 (Oquab et al., 2023) and SSCD (Pizzi et al., 2022), achieve clean ROC AUC values of 0.990 and 0.986 respectively for their best checkpoints. Unfortunately, model performance quickly drops in the presence of a baseline adversary. Attacking the best SSCD model with an $\epsilon = 4/255$, $\ell_\infty$ attack reduces ROC AUC to 0.057, which is completely unusable. However we observe an interesting trend: higher performance models exhibit more resistance to adversarial attack. The current state-of-the-art image embedding models, DINOv2, are noticeably more resistant to our fixed attack. Note that our baseline attack gives an upper bound on robustness—targeted attacks on specific models would likely degrade performance more effectively. Naturally, the higher resistance models result in less visibly perturbed images (see Supplement C.1 for an example). Even if one gains robustness to a specific threat model (e.g., $\ell_\infty$ perturbations), it may not generalize. For example, all evaluated models are also vulnerable to $\ell_1$ perturbations (see Figure 5). We expect similar results to hold for more exotic threat models.

**Takeaway** We want a negligibly low probability of adversarial success but find that an adversary can efficiently break any evaluated embedding model. For our scheme to be deployable, this gap would need to be closed.

### 6.3 High-Capacity Post-Hoc Image Watermarking

We require a post-hoc watermark with a large capacity to encode both a cryptographic signature and a robust embedding into an image. We find a number of ways to increase capacities or reduce capacity requirements.

**Descriptor Quantization** A "free lunch" optimization is to leverage existing quantization techniques. If the original descriptors are represented with float32 values, using 8-bit quantization, for example, immediately reduces capacity requirements by 75% with little performance degradation (Jacob et al., 2018).

**Conversion of Generation-Time Watermarks to Post-Hoc Watermarks** Publicly-accessible post-hoc watermarks currently do not support large enough payloads—for example, TrustMark (Bui et al., 2023a), was evaluated to support at most 200 bits (at which point bit error rate (BER) becomes significant). Despite this, the technological trend suggests that larger payloads may be supported through careful optimization of the capacity-robustness tension.

**Deferred Storage** Instead of storing a signature and embedding directly, one can store a database reference for remote recovery of the full payload. This optimization is necessary if a high-dimensional embedding is used (e.g., DINOv2 (Oquab et al., 2023) or SEER (Goyal et al., 2021) descriptors), and it introduces a critical assumption: there must exist a highly-available database. If the database is offline, detection is impossible. We remark that this optimization has no effect on security as it is merely an extension of the post-hoc watermark. The adversary's capabilities are the same as in the base scheme: it can destroy but not forge watermarks. If extra security is desired, one can encrypt the payload before storing it server-side with an encryption key that is encoded in the image.

**Stable Deep Hashing** Within the image itself, we store an embedding of the image and a signature. The embedding enables checking if a transformed image is still similar to the *original* watermarked image. This comes at a substantial cost: the embedding size is likely to be much larger than the signature, depending on the performance of the source model. Ideally, the robust embedding function should produce identical, discretized embeddings for visually similar images. Future research is necessary to determine whether advanced transformations can be handled by a deep stable hash, but existing schemes are already robust to quantization or re-encoding (Apple, 2021).

**Takeaway** Post-hoc watermark capacity is unlikely to prevent deploying a robust, unforgeable, and publicly-detectable watermark.

## 7 CONCLUDING REMARKS

In this paper, we explore the construction of a robust, unforgeable, and publicly-detectable watermark for images. We prove that such a scheme exists, but its deployment is limited by the white-box security of image embedding models. We now discuss promising avenues for future work.

**On Indestructibility** This work focuses on unforgeability without explicitly handling *indestructibility*—how easily a watermark can be removed without degrading image quality. No scheme is presently known to have this property in the white-box setting (Zhang et al., 2023). We remark that indestructibility may emerge organically as the set of robust

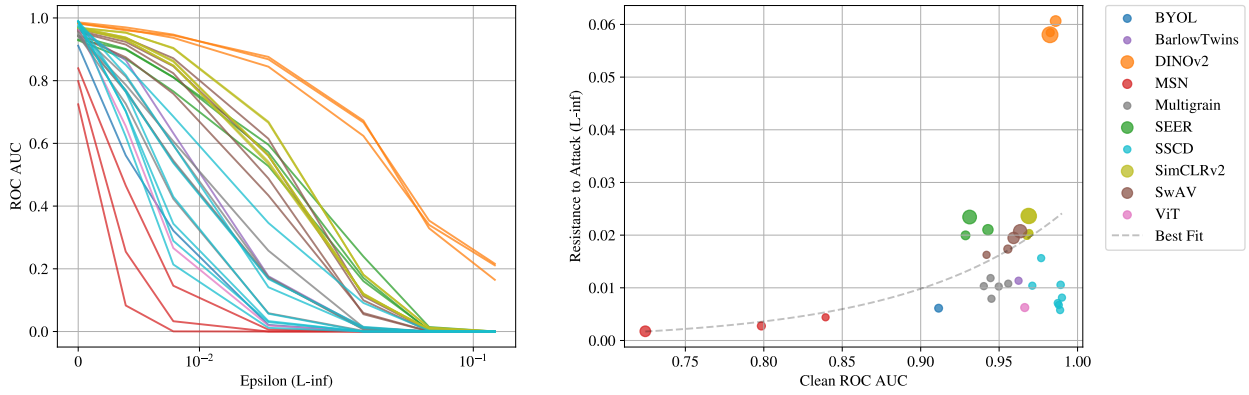


Figure 4: Resistance to $\ell_\infty$ attacks slightly increases with model performance. The y-axis of the right graph is calculated as the area under the corresponding curve in the left graph.

transformations better approximates human vision. Consider the ideal case where the underlying watermark and embedding function are perfectly robust. By definition, it follows that any invalid transformation of the original watermarked image is *not* visually similar and should not be detected as watermarked. This suggests that as robustness improves, indestructibility will also improve.

**Robust Embeddings in the Wild** Robust image embeddings are applied as perceptual hashes for explicit material detection, where care must be taken when the models are deployed in a white-box setting. As an illustrative example, Apple's NeuralHash scheme was a robust image embedding-based perceptual hash deployed to Apple systems in 2021 to detect CSAM images without revealing exact images to Apple servers (Apple, 2021). The robust embedding model was hosted directly on user machines—as a result, adversarial attacks that broke the collision resistance of NeuralHash quickly surfaced (Struppek et al., 2022), which consequently broke the "unforgeability" of the larger system for CSAM detection.

**Adversarial Robustness** The main barrier to deploying our robust watermark is resolving (the lack of) adversarial robustness of image embedding models. We observe a weak correlation between raw model performance and resistance to attacks, suggesting that higher performing models may be more adversarially-robust than weaker models. General progress in adversarial ML may be adapted to existing models to strengthen security in the white-box setting.

**Alternative Pathways to Public Detection** While we have explored a method of combining deep learning and cryptography, we do not rule out the possibility of realizing a robust and publicly-detectable watermark through a different approach: we leave it to future work to develop new schemes that better resist white-box attack. We expect, however, that any scheme leveraging deep learning may inevitably face adversarial attack: the detection algorithm must be fully public, and thus any components used within it must resist public white-box attack.

## Acknowledgements

This work was completed while Jaiden Fairoze was a Student Researcher at Google DeepMind.

## References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alexey, D. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929*.

Anthropic (2024). The Claude 3 Model Family: Opus, Sonnet, Haiku. `https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf`. Online; accessed 7 August 2024.

Apple (2021). CSAM Detection Technical Summary. `https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf`. Online; accessed 8 August 2024.

Assran, M., Caron, M., Misra, I., Bojanowski, P., Bordes, F., Vincent, P., Joulin, A., Rabbat, M., and Ballas, N. (2022). Masked siamese networks for label-efficient learning. In *European Conference on Computer Vision*, pages 456–473. Springer.

Barreto, P. S., Lynn, B., and Scott, M. (2003). Constructing elliptic curves with prescribed embedding degrees. In *Security in Communication Networks: Third International Conference, SCN 2002 Amalfi, Italy, September 11–13, 2002 Revised Papers 3*, pages 257–267. Springer.

Berman, M., Jégou, H., Vedaldi, A., Kokkinos, I., and Douze, M. (2019). Multigrain: a unified image embedding for classes and instances. *arXiv preprint arXiv:1902.05509*.

Boneh, D., Lynn, B., and Shacham, H. (2001). Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security*, pages 514–532. Springer.

Bui, T., Agarwal, S., and Collomosse, J. (2023a). Trustmark: Universal watermarking for arbitrary resolution images. *arXiv preprint arXiv:2311.18297*.

Bui, T., Agarwal, S., Yu, N., and Collomosse, J. (2023b). Rosteals: Robust steganography using autoencoder latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 933–942.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924.

Certicom Research (2010). Sec 2: Recommended elliptic curve domain parameters. `https://www.secg.org/sec2-v2.pdf`. Online; accessed 25 September 2024.

Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388.

Chaudhuri, S., Ganjam, K., Ganti, V., and Motwani, R. (2003). Robust and efficient fuzzy match for online data cleaning. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 313–324.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Coalition for Content Provenance and Authenticity (2023). C2PA Technical Specification. `https://c2pa.org/specifications/specifications/1.3/specs/_attachments/C2PA_Specification.pdf`. Online; accessed 16 July 2024.

Cox, I. J., Kilian, J., Leighton, F. T., and Shamoon, T. (1997). Secure spread spectrum watermarking for multimedia. *IEEE transactions on image processing*, 6(12):1673–1687.

Digimarc (1995). Revolutionize your business with digimarc digital watermarks. `https://www.digimarc.com/`. Accessed: 2024-09-30.

Douze, M., Jégou, H., Sandhawalia, H., Amsaleg, L., and Schmid, C. (2009). Evaluation of gist descriptors for web-scale image search. In *Proceedings of the ACM international conference on image and video retrieval*, pages 1–8.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Elaine, B., Barker, W., Burr, W., Polk, W., and Smid, M. (2016). Recommendation for key management, part 1: General. *NIST Special Publication*, 800:57.

Fairoze, J., Garg, S., Jha, S., Mahloujifar, S., Mahmoody, M., and Wang, M. (2023). Publicly detectable watermarking for language models. *arXiv preprint arXiv:2310.18491*.

Fernandez, P., Couairon, G., Jégou, H., Douze, M., and Furon, T. (2023). The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22466–22477.

Fernandez, P., Sablayrolles, A., Furon, T., Jégou, H., and Douze, M. (2022). Watermarking images in self-supervised latent spaces. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3054–3058. IEEE.

Ge, T., He, K., Ke, Q., and Sun, J. (2013). Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2946–2953.

Gegg-Harrison, W. and Quarterman, C. (2024). Ai detection's high false positive rates and the psychological and material impacts on students. In *Academic Integrity in the Age of Artificial Intelligence*, pages 199–219. IGI Global.

Gionis, A., Indyk, P., Motwani, R., et al. (1999). Similarity search in high dimensions via hashing. In *Vldb*.

Google DeepMind (2023). Identifying ai-generated content with synthid. `https://deepmind.google/technologies/synthid/`. Accessed: 2024-09-30.

Goyal, P., Caron, M., Lefaudeux, B., Xu, M., Wang, P., Pai, V., Singh, M., Liptchinsky, V., Misra, I.,

Joulin, A., et al. (2021). Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*.

GPTZero (2023). GPTZero. `hhttps://gptzero.me/`. Online; accessed 7 August 2024.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284.

Ha, A. Y. J., Passananti, J., Bhaskar, R., Shan, S., Southen, R., Zheng, H., and Zhao, B. Y. (2024). Organic or diffused: Can we distinguish human art from ai-generated images? *arXiv preprint arXiv:2402.03214*.

Haitsma, J. and Kalker, T. (2002). A highly robust audio fingerprinting system. In *Ismir*, volume 2002, pages 107–115.

Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

Johnson, D., Menezes, A., and Vanstone, S. (2001). The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1:36–63.

Katz, J. and Lindell, Y. (2014). *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition.

Laurie, B. (2014). Certificate transparency. *Communications of the ACM*, 57(10):40–46.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Midjourney (2022). Midjourney. `https://www.midjourney.com/home`.

Muyco, S. D. and Hernandez, A. A. (2019). Least significant bit hash algorithm for digital image watermarking authentication. In *Proceedings of the 2019 5th International Conference on Computing and Artificial Intelligence*. Association for Computing Machinery.

Navas, K., Ajay, M. C., Lekshmi, M., Archana, T. S., and Sasikumar, M. (2008). Dwt-dct-svd based watermarking. In *2008 3rd international conference on communication systems software and middleware and workshops (COMSWARE'08)*, pages 271–274. IEEE.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. (2023). Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*.

Pan, Titusz (2022). ISCC - Enhancement Proposals (IEPs). `https://ieps.iscc.codes/`. Online; accessed 16 July 2024.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Pizzi, E., Roy, S. D., Ravindra, S. N., Goyal, P., and Douze, M. (2022). A self-supervised descriptor for image copy detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14532–14542.

Prokos, J., Fendley, N., Green, M., Schuster, R., Tromer, E., Jois, T., and Cao, Y. (2023). Squint hard enough: Attacking perceptual hashing with adversarial machine learning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 211–228.

Reid, M., Savinov, N., Teplyashin, D., Lepikhin, D., Lillicrap, T., Alayrac, J.-b., Soricut, R., Lazaridou, A., Firat, O., Schrittwieser, J., et al. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Rivest, R., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Saltman, E. and Thorley, T. (2021). Practical and technical considerations. *Broadening the GIFCT Hash-Sharing Database Taxonomy: An Assessment and Recommended Next Steps*, page 12.

Seo, J. S., Haitsma, J., Kalker, T., and Yoo, C. D. (2004). A robust image fingerprinting system using the radon transform. *Signal Processing: Image Communication*, 19(4):325–339.

Steg (2019). Forensic watermarking for digital media. `https://steg.ai/`. Accessed: 2024-09-30.

StopNCII (2024). Stop non-consensual intimate image abuse. `https://stopncii.org/`. Accessed: 2024-09-30.

Struppek, L., Hintersdorf, D., Neider, D., and Kersting, K. (2022). Learning to break deep perceptual hashing: The use case neuralhash. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 58–69.

Tancik, M., Mildenhall, B., and Ng, R. (2020). Stegastamp: Invisible hyperlinks in physical photographs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2117–2126.

Turnitin (1998). Plagiarism prevention trusted by educators worldwide. `https://www.turnitin.com/products/similarity/`. Accessed: 2024-09-30.

Van Den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Venkatesan, R., Koon, S.-M., Jakubowski, M. H., and Moulin, P. (2000). Robust image hashing. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 3, pages 664–666. IEEE.

Wan, W., Wang, J., Zhang, Y., Li, J., Yu, H., and Sun, J. (2022). A comprehensive survey on robust image watermarking. *Neurocomputing*, 488:226–247.

Wang, A. (2006). The shazam music recognition service. *Communications of the ACM*, 49(8):44–48.

Wang, A. et al. (2003). An industrial strength audio search algorithm. In *Ismir*, volume 2003, pages 7–13. Washington, DC.

Wen, Y., Kirchenbauer, J., Geiping, J., and Goldstein, T. (2023). Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *arXiv preprint arXiv:2305.20030*.

Wolfgang, R. B. and Delp, E. J. (1996). A watermark for digital images. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 3, pages 219–222. IEEE.

Yang, Z., Zeng, K., Chen, K., Fang, H., Zhang, W., and Yu, N. (2024). Gaussian shading: Provable performance-lossless image watermarking for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12162–12171.

Yeo, I.-K. and Kim, H. J. (2003). Generalized patchwork algorithm for image watermarking. *Multimedia systems*, 9:261–265.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR.

Zhang, H., Edelman, B. L., Francati, D., Venturi, D., Ateniese, G., and Barak, B. (2023). Watermarks in the sand: Impossibility of strong watermarking for generative models. *arXiv preprint arXiv:2311.04378*.

Zhu, J., Kaplan, R., Johnson, J., and Fei-Fei, L. (2018). Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672.

## Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Not Applicable
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Not Applicable

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes
   (b) Complete proofs of all theoretical results. Yes
   (c) Clear explanations of any assumptions. Yes

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). No. Not necessary.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Not Applicable
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes
   (b) The license information of the assets, if applicable. Yes
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable
   (d) Information about consent from data providers/curators. Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable

# Supplementary Materials

## A EXTENDED RELATED WORK

Here we expand on the watermarking and robust embedding literature.

### A.1 Watermarking

The literature on watermarking can be grouped into two categories: classical watermarking and deep watermarking. We highlight key papers as follows.

**Classical Watermarking** Classical watermarking primarily aims to embed watermarks to enforce copyright. An early idea was to embed secret information into the least significant bits of each pixel, leading to a low-distortion watermark (Wolfgang and Delp, 1996)—this is similar to our simple non-robust scheme. Like ours, since the watermark is embedded in the 'least important" bits, the watermark is non-robust. Cox et al. (1997) developed the spread-spectrum technique where the watermark is instead embedded in the frequency domain. This led to robustness to re-encoding, minor crops, and lossy compression. Many subsequent works followed with the aim of increasing payload size and robustness. The patchwork algorithm (Yeo and Kim, 2003) and DWT-DCT-SVD (Navas et al., 2008) are particularly well adopted solutions.

**Deep Watermarking** More recent watermarks turned to deep learning for better performance. Such watermarks demonstrated higher payloads, higher robustness, and lower image distortion than classical methods (Wan et al., 2022). HiDDeN (Zhu et al., 2018) was the first encoder-decoder watermark for images. StegaStamp (Tancik et al., 2020) refined the encoder-decoder architecture by leveraging spatial information. This led to better robustness against geometric transformations. SSL (Fernandez et al., 2022) uses a different approach and instead watermarks images in the latent space during inference—this trades off image quality for inference speed. RoSteALS (Bui et al., 2023b) similarly applies the watermark in the latent space but instead uses a VQ-VAE architecture (Van Den Oord et al., 2017). Finally, TrustMark (Bui et al., 2023a) is resolution invariant and supports re-watermarking out of the box.

More recently, watermarking schemes have been tailor-made for specific generation processes. In the image setting, the Stable Signature (Fernandez et al., 2023) introduced an active strategy to embed the watermark during diffusion. The latent decoder is fine-tuned to embed a binary signature which a pre-trained watermark extractor can identify. This lower-level approach led to higher performance than post-hoc methods. The Tree-Ring method (Wen et al., 2023) also performs watermarking in the latent space and is designed to minimize distortions at the cost of model performance. Gaussian Shading (Yang et al., 2024) is the latest image watermark for diffusion models. The approach is claimed to be computationally distortion-free: the watermarked distribution is computationally close to the native output distribution. In the text setting, Fairoze et al. (2023) gave an unforgeable and publicly-detectable watermark with weak robustness.

### A.2 Robust Embeddings

There are a wide range of problems that come under the umbrella of robust embeddings. The fundamental task in any one of these formulations is to determine if two objects (images) are similar or not from a succinct representation of the original objects. If this core problem is solvable, then it can be applied to various applications such as perceptual hashing, fingerprinting, or copy detection. We summarize the literature for each of these tracks.

**Perceptual Hashing** Indyk and Motwani (1998) introduced locality-sensitive hashing (LSH). Gionis et al. (1999) gave a more efficient construction that requires asymptotically fewer queries when applied to nearest neighbor search. Random hyperplane-based perceptual hashing is the most widespread method of performing

LSHs (Charikar, 2002). The technique involves dividing up the embedding space by randomly partitioning it with hyperplanes—each cell produced by the partitioning process is assigned a unique hash value. This method was adopted by Apple for their NeuralHash system (Apple, 2021). To hash an image using NeuralHash, features are first extracted using a convolutional neural network. The features are then perceptually hashed with a hyperplane-based LSH. Struppek et al. (2022) later demonstrated that it is easy to perturb a arbitrary image to be closely embedded to any arbitrary different image, breaking NeuralHash completely. ITQ (Ge et al., 2013) is a deep learning approach to achieving the same effect as a classical LSH, i.e. ITQ produces a binary hash.

**Fingerprinting** Venkatesan et al. (2000) introduced the notion of a robust image fingerprint and used wavelet representations of images to gain robustness to common transformations. Seo et al. (2004) improved their approach by instead using Radon transforms for robustness to affine trasformations. In the audio setting, similar results exist (Haitsma and Kalker, 2002) that make optimizations specific to the audio domain. In a similar vein, Wang et al. (2003) developed a highly effective audio fingerprint for identifying music. Their techniques formed the backbone of Shazam (Wang, 2006).

**Feature Extraction** We highlight key works that are relevant to this paper. We refer the reader to the DINOv2 paper (Oquab et al., 2023) for a more comprehensive overview of the technical state of image feature extraction. The Simple Framework for Contrastive Learning of Visual Representations (SimCLR) (Chen et al., 2020) demonstrated that many prior self-supervised learning algorithms could be simplified, removing the need for specialized architectures or memory banks. This simplification led to better performing models that required fewer weights than prior approaches. Pizzi et al. (2022) augmented SimCLR with entropy regularization, InfoNCE loss (Oord et al., 2018), and inference-time score normalization to develop SSCD. The model was recently used by the DINOv2 and Llama 3 teams to identify duplicate images (Oquab et al., 2023; Dubey et al., 2024). Berman et al. (2019) developed Multigrain: a network architecture designed to produce compact descriptors for image classification and object retrieval downstream tasks. It crucially leverages different levels of image "granularity" to learn generalized features. Bootstrap Your Own Latents (BYOL) (Grill et al., 2020) uses two sub-networks, denoted the online and target networks. The online network learns to predict the target network representation of the same image under a particular transformation. BYOL descriptors were found to outperform SimCLR descriptors. SwAV (Caron et al., 2020) is designed to learn constrastively without directly making pairwise comparisons. The method aims to cluster transformations of the same image together: this has a similar effect as direct constrastive comparison. The approach was also found to outperform SimCLR. Zbontar et al. (2021) developed Barlow Twins. The method aims to avoid learning trivial solutions (e.g., constant embeddings) by optimizing cross-correlations between outputs of two identical networks (each fed with transformed images) to be close to a target cross-correlation. Aside from avoiding collapse, this has an additional effect of minimizing redundant embedding information. Barlow Twins descriptors were found to outperform BYOL and SwAV descriptors. SEER (Goyal et al., 2021) is another self-supervised learning method that further closed the gap with supervised methods. Their models are based on SwAV and are optimized to scale—SEER achieved new state-of-the-art across a range of model size classes, scaling to billions of parameters. Assran et al. (2022) developed Masked Siamese Networks (MSN). The framework aims to match the representation of masked images with the original image. This naturally interfaces with Vision Transformers (ViT) (Alexey, 2020) as they only handle unmasked images by default. This combination was found to scale well and outperformed prior work. DINOv2 (Oquab et al., 2023) is another ViT-based self-supervised method that aimed to consolidate research following SEER to optimize performance at scale. They demonstrate that careful orchestration of the training pipeline coupled with recent advancements in self-supervision led to new state-of-the-art embeddings. In particular,DINOv2 image patches were shown to semantically capture various aspects of human vision.

## B FORMALISM

We present the necessary primitives for formal analysis of robust and publicly-detectable watermarking schemes (RPWS). In doing so, we precisely define what properties each primitive must satisfy—this allows us to parameterize the resulting RPWS later on.

### B.1 Definitions

**Definition B.1** (Robust embedding function)**.** A $(\mathcal{T}, m, n, \epsilon)$-robust embedding function ($\mathsf{REF}$) is a triple of algorithms ($\mathsf{Generate}, \mathsf{Embed}$) defined as follows:

1. $\mathsf{Generate}(1^\lambda, m, n, \mathcal{T}) \to (\mathsf{Embed}, \mathsf{Compare})$. Generate takes the input size $m$, output size $n$, and the security parameter $\lambda$, outputting the robust embedding function in time $\mathsf{poly}(\lambda)$.
2. $\mathsf{Embed}(x) \to e$. The function $\mathsf{Embed} : \{0,1\}^m \to \{0,1\}^n$ takes an arbitrary object $x$ and maps it to a discrete point in the embedding space $e$.
3. $\mathsf{Compare}(x, y) \to b$. The function $\mathsf{Compare} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ takes two embeddings $x$ and $y$ and outputs a single bit $b$ such that $b$ is `true` when $x$ and $y$ are similar, and `false` otherwise.

A valid robust embedding function must satisfy the following two properties:

1. **Correctness.** For all $x$ and transformations $T \in \mathcal{T}$,

$$\Pr\left[\ \mathsf{Compare}(\mathsf{Embed}(x), \mathsf{Embed}(T(x))) = \texttt{true} \quad : \quad \mathsf{Embed}, \mathsf{Compare} \leftarrow \mathsf{Generate}(1^\lambda, m, n, \mathcal{T})\ \right] \geq 1 - \epsilon.$$

2. **Collision-resistance.** For any choice of $x$, it must be that

$$\Pr\left[\begin{array}{c} \mathsf{Compare}(\mathsf{Embed}(x), \mathsf{Embed}(x^*)) = \texttt{true} \\ \wedge\ x^* \notin \Gamma(x) \end{array} : \begin{array}{c} \mathsf{Embed}, \mathsf{Compare} \leftarrow \mathsf{Generate}(1^\lambda, m, n, \mathcal{T}) \\ x^* \leftarrow \mathcal{A}^{\mathsf{Embed}(\cdot)}(m, n, \mathcal{T}, x) \end{array}\right] \leq \epsilon.$$

**Definition B.2** (Cryptographic signature scheme)**.** A $(\delta, \lambda)$-cryptographic signature scheme $\mathsf{SIG}$ is a 3-tuple ($\mathsf{Generate}, \mathsf{Sign}, \mathsf{Verify}$) defined as follows:

1. $\mathsf{Generate}(1^\lambda) \to (sk, pk)$. Takes in the target security bits number $\lambda$ and outputs a fresh secret key $sk$ and public key $pk$ pair in time $\mathsf{poly}(\lambda)$.
2. $\mathsf{Sign}(sk, x) \to \sigma$. Given the secret key $sk$ and a object to sign $x$, $\mathsf{Sign}$ computes $\sigma \in \{0,1\}^\delta$: a signature of $sk$ on $x$.
3. $\mathsf{Verify}(pk, x, \sigma) \to b$. Given a public key $pk$, object $x$, and signature $\sigma$, $\mathsf{Verify}$ checks if $\sigma$ is a valid signature of the corresponding secret key of $pk$ on $x$ and encodes the result into one bit $b$. It returns either `true` or `false` depending on if verification succeeded or not.

A valid $\mathsf{SIG}$ needs to satisfy the following properties:

1. **Correctness.** It holds that for all $x$,

$$\Pr\left[\ \mathsf{Verify}(pk, x, \mathsf{Sign}(sk, x)) = \texttt{true} \quad : \quad (sk, pk) \leftarrow \mathsf{Generate}(1^\lambda)\ \right] \geq 1 - \mathsf{negl}(\lambda).$$

2. **Unforgeability**. Let $X$ denote the set of oracle queries that the adversary makes to $\mathsf{Sign}(sk, \cdot)$. It holds that for all $x$,

$$\Pr\left[\begin{array}{c} \mathsf{Verify}(pk, x^*, \sigma^*) = \texttt{true} \\ \wedge\ x^* \notin X \end{array} : \begin{array}{c} (sk, pk) \leftarrow \mathsf{Generate}(1^\lambda) \\ (x^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk, \cdot)}(pk) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

**Definition B.3** (Post-hoc watermarking scheme)**.** A $(\mathcal{T}, c, \epsilon)$-post-hoc watermarking scheme $\mathsf{PGWS}$ is a 3-tuple of algorithms defined as follows:

1. $\mathsf{Generate}(1^\lambda, c, \mathcal{T}) \to (\mathsf{Encode}, \mathsf{Decode})$. Takes in the target security bits number $\lambda$, the capacity size in bits $c$, and the set of possible transformations $\mathcal{T}$ and outputs the encode and decode algorithms in time $\mathsf{poly}(\lambda)$.

2. $\mathsf{Encode}(x, m) \to x'$. Given a message $m$ such that $m \in \{0,1\}^c$ and target object $x$, $\mathsf{Encode}$ plants $m$ a watermark in $x$ producing $x'$.

3. $\mathsf{Decode}(x^*) \to m$. Given a potentially watermarked object $x^*$, $\mathsf{Decode}$ recovers $m \in \{0,1\}^c$ if it exists.

Let $\mathcal{X}$ and $\mathcal{M}$ be the set of all valid objects and messages respectively. A valid $\mathsf{PGWS}$ must satisfy correctness with optional private unforgeability:

1. **Correctness.** For all choices of $x$, $m$, and $\mathcal{T}$,

$$\Pr\left[\ \mathsf{Decode}(T(\mathsf{Encode}(x,m))) = m \quad : \quad \mathsf{Encode}, \mathsf{Decode} \leftarrow \mathsf{Generate}(1^\lambda, c, \mathcal{T})\ \right] \geq 1 - \epsilon.$$

2. **Private Unforegability.** Let $X$ be the list of object queries that $\mathcal{A}$ makes to the decoding oracle and let $\Gamma(X)$ be the set containing all neighboring query objects. It must be that for all $x$, $m$, and $\mathcal{A}$,

$$\Pr\left[\begin{array}{c}\mathsf{Decode}(x^*, m) = m \\ \wedge\ x^* \notin \Gamma(X)\end{array} \quad : \quad \begin{array}{c}\_, \mathsf{Decode} \leftarrow \mathsf{Generate}(1^\lambda, c, \mathcal{T}) \\ x^* \leftarrow \mathcal{A}^{\mathsf{Decode}(\cdot,\cdot)}(c, \mathcal{T}, x, m)\end{array}\right] \leq \epsilon.$$

   Note that this property is not a strict requirement for the purpose of constructing a robust and publicly-detectable watermark. We include a definition of private unforgeability for completeness.

**Definition B.4** (Robust publicly-detectable watermarking scheme)**.** A $(\mathcal{T}, \epsilon)$-robust publicly-detectable watermarking scheme $\mathsf{RPWS}$ consists of three algorithms defined as follows:

1. $\mathsf{Generate}(1^\lambda) \to (sk, pk)$. Takes in the security parameter $\lambda$ and outputs a fresh secret key $sk$ and public key $pk$ pair.

2. $\mathsf{Watermark}(sk, x) \to x'$. Given a secret key $sk$ and target object $x$, $\mathsf{Watermark}$ plants a watermark in $x$ under $sk$ producing $x'$.

3. $\mathsf{Detect}(pk, x^*) \to b$. Given a potentially watermarked object $x^*$, $\mathsf{Detect}$ checks if there is valid watermark under the corresponding secret key to $pk$. It outputs `true` or `false` as a single bit depending on if verification succeeds or not.

A valid $\mathsf{RPWS}$ must satisfy the following properties:

1. **Correctness.** It holds that for all $x$,

$$\Pr\left[\ \mathsf{Detect}(pk, T(\mathsf{Watermark}(sk, x))) = \texttt{true} \quad : \quad (sk, pk) \leftarrow \mathsf{Generate}(1^\lambda)\ \right] \geq 1 - \epsilon.$$

2. **Unforegability.** Let $X$ denote the set of oracle queries that the adversary makes to $\mathsf{Watermark}(sk, \cdot)$. It holds that for all $x$,

$$\Pr\left[\begin{array}{c}\mathsf{Detect}(pk, x^*) = \texttt{true} \\ \wedge\ x^* \notin \Gamma(X)\end{array} \quad : \quad \begin{array}{c}(sk, pk) \leftarrow \mathsf{Generate}(1^\lambda) \\ (x^*) \leftarrow \mathcal{A}^{\mathsf{Watermark}(sk,\cdot)}(pk)\end{array}\right] \leq \epsilon.$$

### B.2 Constructing a Robust and Publicly-Detectable Watermark

In this section we will prove the following theorem:

**Theorem 5.1.** *If $(\mathcal{T}_{\mathsf{REF}}, m, n, \epsilon_{\mathsf{REF}})$-robust embedding functions, $(\mathcal{T}_{\mathsf{PGWS}}, c, \epsilon_{\mathsf{PGWS}})$-post-hoc watermarking schemes, and $(\delta, \lambda)$-cryptographic signatures exist such that $c \geq \delta + n$ and $\mathsf{PGWS.Encode} \in \mathcal{T}_{\mathsf{REF}}$, then $(\mathcal{T}_{\mathsf{REF}} \cap \mathcal{T}_{\mathsf{PGWS}}, \epsilon_{\mathsf{REF}} + \epsilon_{\mathsf{PGWS}} + \mathsf{negl}(\lambda))$-publicly-detectable watermarking schemes also exist.*

*Proof.* To construct a PGWS, we compose the REF, SIG and PGWS in the natural way: the REF is used to compute a stable representation (embedding) of the image. This embedding is cryptographically signed using SIG and both the signature and embedding are encoded within the image using the PGWS. In order to detect the watermark in an arbitrary image, it suffices to (a) check that the embedding of the image at hand is similar to the encoded embedding, and (b) check that the signature is authentic. We formalize this simple construction as follows.

Generate. On input $1^\lambda$, run $\mathsf{SIG.Generate}(1^\lambda)$ to obtain $sk$ and $pk$. Output $(sk, pk)$.

Watermark. Given the secret key $sk$ and cover image $x \in \mathcal{X}$,

1. Compute the robust embedding $e = \mathsf{REF.Embed}(x)$.
2. Sign the embedding $\sigma \leftarrow \mathsf{SIG.Sign}(sk, e)$.
3. Plant $\sigma$ and $e$ into $x$: $x' \leftarrow \mathsf{PGWS.Encode}(x, \sigma \parallel e)$.
4. Output $x'$.

Detect. Given a public key $pk$ and candidate watermarked object $x'$,

1. Compute the aggregated hash $e = \mathsf{REF.Embed}(x')$.
2. Decode the embedded payload if it exists $\sigma' \parallel e' \leftarrow \mathsf{PGWS.Decode}(x')$.
3. Attempt signature verification $b_1 \leftarrow \mathsf{SIG.Verify}(pk, e', \sigma')$.
4. Attempt closeness verification $b_2 \leftarrow \mathsf{REF.Compare}(e, e')$.
5. Output $b_1 \wedge b_2$.

*Claim* 1. The above scheme is correct if the underlying signature scheme and post-generation watermarking scheme are both correct.

*Proof.* Observe that this RPWS is robust to the intersection of the input transformation sets, i.e., $\mathcal{T}_{\mathsf{RPWS}} = \mathcal{T}_{\mathsf{REF}} \cap \mathcal{T}_{\mathsf{PGWS}}$—any transformation applied to the image that is not in this set will result in either incorrect decoding using the PGWS or REF.Compare outputting `false`. Correctness immediately follows from the construction: we calculate the success probability loss as follows. Given $(sk, pk) \leftarrow \mathsf{Generate}(1^\lambda)$, $\mathsf{Detect}(pk, x) \iff \mathsf{SIG.Verify}(pk, \cdot, \cdot) \wedge \mathsf{REF.Compare}(\cdot, \cdot)$ must output true for any honestly generated $x$. Signature verification fails with probability $\mathsf{negl}(\lambda)$. The similarity check fails with probability $\epsilon_{\mathsf{REF}}$. Decoding fails with probability $\epsilon_{\mathsf{PGWS}}$. Thus the overall check will succeed with probability at least $1 - (\epsilon_{\mathsf{REF}} + \epsilon_{\mathsf{PGWS}} + \mathsf{negl}(\lambda)) =: 1 - \epsilon_{\mathsf{PGWS}}$. □

*Claim* 2. If the scheme is forgeable then either the underlying signature scheme is forgeable or the robust embedding function is not collision-resistant.

*Proof.* Assume for the sake of contradiction that there exists an adversary $\mathcal{A}_{\mathsf{RPWS}}$ that can break the unforgeability of the above RPWS. We will show that there exists an adversary that can produce an object $x^*$ such that $\mathsf{RPWS.Detect}(x^*) = \mathtt{true}$ that was not honestly-generated or within the neighborhood of transformed honestly-generated objects. Consider the following two reductions—we will prove that at least one of them will succeed in their respective security game if $\mathcal{A}_{\mathsf{RPWS}}$ is successful in breaking the RPWS.

First, we construct an adversary $\mathcal{A}_{\mathsf{SIG}}$ against the underlying signature scheme's unforgeability.

Construction of $\mathcal{A}_{\mathsf{SIG}}$. On input $pk$ (and oracle access to the private signing algorithm):

1. Run $\mathcal{A}_{\mathsf{RPWS}}(pk)$ to obtain $x^*$ such that with high probability $\mathsf{RPWS.Detect}(pk, x^*) = \mathtt{true}$.
2. Compute $\sigma^* \parallel e^* \leftarrow \mathsf{PGWS.Decode}(x^*)$.
3. Return $(e^*, \sigma^*)$ as a forgery.

Similarly, we construct $\mathcal{A}_{\mathsf{REF}}$ that breaks the collision-resistance of the underlying robust embedding function as follows:

Construction of $\mathcal{A}_{\mathsf{REF}}$. On input $x$, $\mathcal{T}$:

1. Run $\mathcal{A}_{\mathsf{RPWS}}(pk)$ to obtain $x^*$ such that with high probability $\mathsf{Detect}(pk, x^*) = \texttt{true}$.
2. Compute the output of REF.Embed: $e \leftarrow \mathsf{REF.Embed}(x^*)$.
3. Decode the payload $\sigma^* \parallel e^* \leftarrow \mathsf{Decode}(x^*)$
4. Submit $(e, e^*)$ as a collision.

We know that with high probability, $\mathsf{Detect}(pk, x^*) = \mathsf{SIG.Verify}(pk, e^*, \sigma^*) \wedge \mathsf{REF.Compare}(e, e^*) = \texttt{true}$ implying that either Verify or Compare has been forged such that $x^* \notin \Gamma(x)$. It is easy to see that at least one of $\mathcal{A}_{\mathsf{SIG}}$ or $\mathcal{A}_{\mathsf{REF}}$ will succeed: in order for Detect to be true for some $x^*$ that is not in the neighborhood of an honestly-watermarked image, the adversary must have forged a signature or broken the embedding function's collision resistance (or both). Since we know that REF and SIG are respectively secure, we have our contradiction and it holds that the RPWS is unforgeable. □

Finally, by assumption, the capacity $c$ of the watermark is large enough to support a signature and a robust embedding, i.e., $c \geq n + \delta = |\sigma| + |e|$ and so it is valid to store both objects within the image using the PGWS.

Combining Claim 1 and Claim 2 yields the proof. □

# C EXPERIMENTAL DATA

We provide additional data from our evaluation of state-of-the-art pre-trained image embedding models to instantiate the robust embedding function.

**Platform** All model checkpoints were evaluated using PyTorch (Paszke et al., 2019) on a virtual machine running Debian 11 equipped with a single NVIDIA A100 GPU with 40GB VRAM.

**Models** We enumerate all evaluated models in Table 1.

| *Model* | **Architecture** | **Data** | **Parameter Count** | **Dimensions** |
|---|---|---|---|---|
| *BYOL* (Grill et al., 2020) | ResNet-50 | ImageNet | 72M | 2048 |
| *DINOv2* (Oquab et al., 2023) | ViT-g/14 | LVD | 86M – 1136M | 257 × 768 – 257 × 1536 |
| *SEER* (Goyal et al., 2021) | RG256 | IG | 141M – 637M | 3712 × 12 × 12 – 7392 × 12 × 12 |
| *MSN* (Assran et al., 2022) | ViT-L/7 | ImageNet | 85M – 303M | 197 × 384 – 197 × 1024 |
| *BarlowTwins* (Zbontar et al., 2021) | ResNet-50 | ImageNet | 25M | 1000 |
| *SwAV* (Caron et al., 2020) | RX101-32x16d | ImageNet | 25M – 586M | 1000 – 10240 |
| *ViT* (Alexey, 2020) | Vit-B/16 | JFT | 86M | 197 × 768 |
| *Multigrain* (Berman et al., 2019) | ResNet-50 | ImageNet | 25M | 2048 |
| *SimCLRv2* (Chen et al., 2020) | ResNet-152x3 | ImageNet | 25M – 800M | 2048 – 6144 |
| *SSCD* (Pizzi et al., 2022) | ResNeXt101-32x4 | DISC | 24M – 44M | 512 – 1024 |

Table 1: Image embedding model metadata. If multiple architectures or checkpoints are evaluated for a given model, the minimum and maximum parameter counts and dimensions are reported along with the largest architecture.

## C.1 Attacked Imagery

Recall that our PGD-based adversarial attack seeks to perform the following for a given image triple ($I_a$, $I_{a'}$, $I_b$): for the positive pair $I_a$ and $I_{a'}$ that are visually similar, it aims to force $\mathsf{Embed}(I_a)$ and $\mathsf{Embed}(I_{a'})$ to have a *low* $\ell_2$-normalized dot product. For the negative pair $I_a$ and $I_b$ that are not visually similar, it aims to force $\mathsf{Embed}(I_a)$ and $\mathsf{Embed}(I_b)$ to have a *high* $\ell_2$-normalized dot product. We provide examples of attack perturbations against DINOv2 (giant) and SSCD (imagenet-advanced) checkpoints for a select image triple below. In Figure 6 we

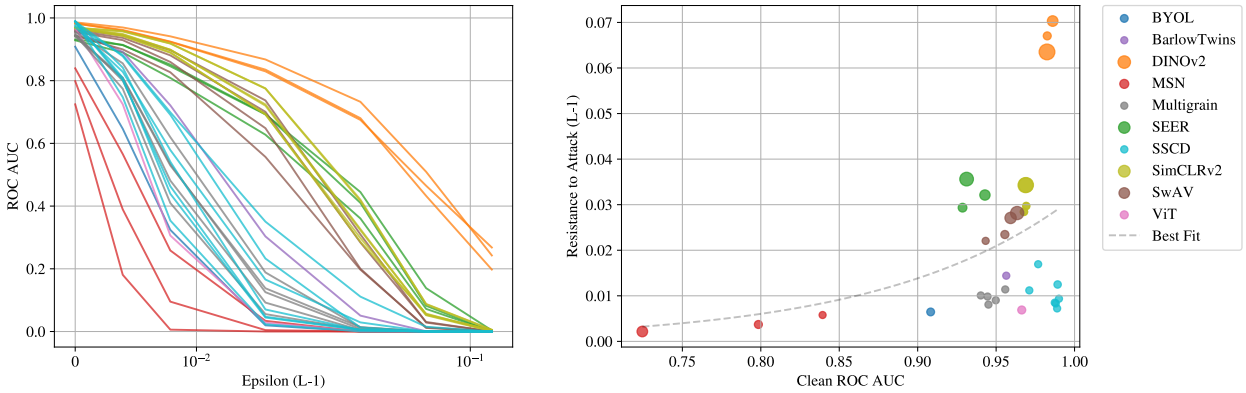


Figure 5: Resistance to $\ell_1$ attacks slightly increases with model performance. The y-axis of the right figure is calculated as the area under the corresponding curve in the left figure.

show the base image $I_a$. In Figures 7, 9, 11 and 13, we show examples of attacked $I_{a'}$ images for various epsilon choices. In Figures 8, 10, 12 and 14, we show examples of attacked $I_b$ images for various epsilon choices. Note that $I_{a'}$ is a "cropped and affine-transformed" version of $I_a$ and $I_b$ is a completely unrelated image.

Figure 6: Base image.

Figure 7: $\ell_\infty$-attacked positive image for $\epsilon \in \{0/255, 1/255, 2/255, 4/255, 8/255, 16/255, 32/255\}$ over DINOv2 (giant).

Figure 8: $\ell_\infty$-attacked negative image for $\epsilon \in \{0/255, 1/255, 2/255, 4/255, 8/255, 16/255, 32/255\}$ over DINOv2 (giant).

Figure 9: $\ell_1$-attacked positive image for $\epsilon \in \{0/255, 1/255, 2/255, 4/255, 8/255, 16/255, 32/255\}$ over DINOv2 (giant).

Figure 10: $\ell_1$-attacked negative image for $\epsilon \in \{0/255, 1/255, 2/255, 4/255, 8/255, 16/255, 32/255\}$ over DINOv2 (giant).

Figure 11: $\ell_\infty$-attacked positive image for $\epsilon \in \{0/255, 1/255, 2/255, 4/255, 8/255, 16/255, 32/255\}$ over SSCD (imagenet-advanced).

Figure 12: $\ell_\infty$-attacked negative image for $\epsilon \in \{0/255, 1/255, 2/255, 4/255, 8/255, 16/255, 32/255\}$ over SSCD (imagenet-advanced).

Figure 13: $\ell_1$-attacked positive image for $\epsilon \in \{0/255, 1/255, 2/255, 4/255, 8/255, 16/255, 32/255\}$ over SSCD (imagenet-advanced).

Figure 14: $\ell_1$-attacked negative image for $\epsilon \in \{0/255, 1/255, 2/255, 4/255, 8/255, 16/255, 32/255\}$ over SSCD (imagenet-advanced).