Provenance Detection for AI-Generated Images: Combining Perceptual Hashing, Homomorphic Encryption, and AI Detection Models

Shree Singhi[1,2], Aayan Yadav[1,2], Aayush Gupta[3], Shahriar Ebrahimi[4], Parisa Hassanizadeh[5]
[1]Indian Institute of Technology Roorkee: {*shree_s, aayan_y*} *@mfs.iitr.ac.in*, [2]Zellic
[3]ZK Email: *aayushg@mit.edu*, [4]Newcastle University: *shahriar.ebrahimi@ncl.ac.uk*
[5]Polish Academy of Science: *parisaa.hassanizadeh@gmail.com*

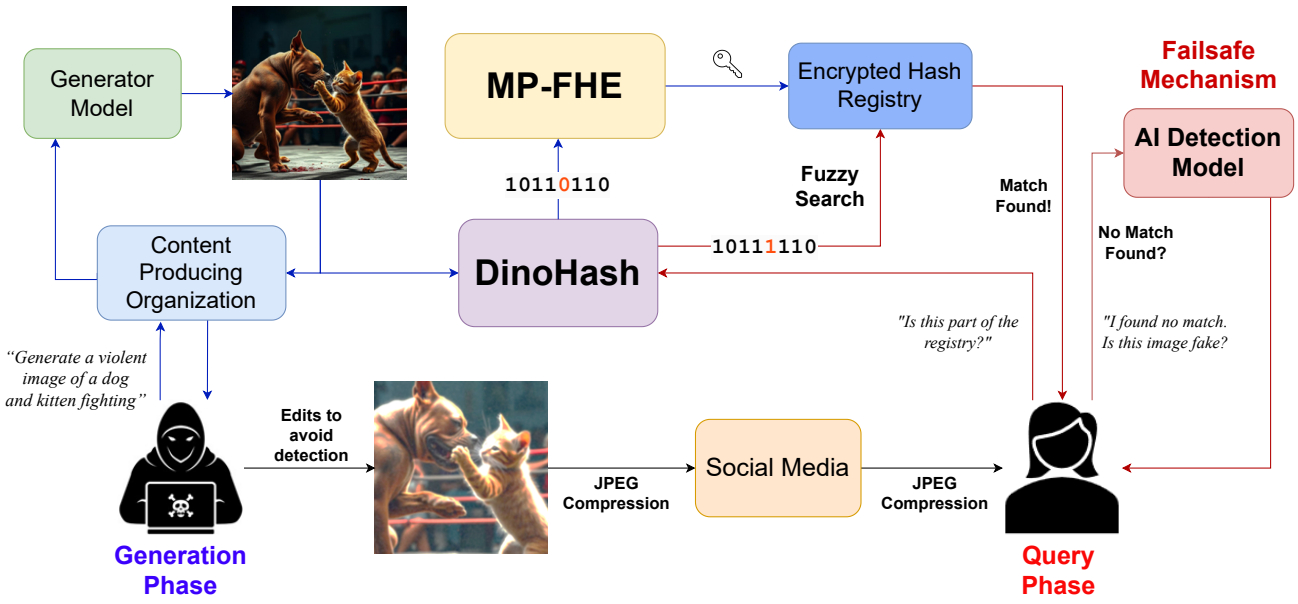

Figure 1. **System Overview.** We combine the strengths of our own perceptual hashing model, DinoHash, Multi-Party Fully Homomorphic Encryption (MP-FHE) and improved AI Detection Model to build a robust provenance framework.

Abstract

With the rise of AI-generated sensitive images, knowing the source of content is critical to distinguish AI generated images. Traditional image watermarking methods lack robustness to common image transformations such as filters, lossy compression during social-media sharing and screenshots. Watermarks can also be faked or removed if models are open-sourced or leaked, since images can be rewatermarked. We have developed a three-part framework for secure, transformation-resilient AI content provenance detection, to address these limitations. We develop an adversarially robust state-of-the-art perceptual hashing model, DinoHash, derived from DINOV2, robust to common transformations like filters, compression and crops. Additionally, we integrate a Multi-Party Fully Homomorphic Encryption (MP-FHE) scheme into our proposed framework to ensure the protection of both user queries and registry privacy. Furthermore, we improve previous work on AI-generated media detection. This approach is useful in cases where the content is absent from our registry. DinoHash achieves a significant improvement in average bit accuracy, showing a 12% increase over state-of-the-art watermarking and perceptual hashing methods, with a consistent edge in true positive rate (TPR) and false positive rate (FPR) tradeoffs across a wide range of transformations. Our AI-generated media detection results perform consistently better showing a 25% improvement in classification accuracy on commonly used real-world AI image generators than existing algorithms. By combining perceptual hashing, MP-FHE, and an AI content detection model, our proposed framework provides better robustness and privacy compared to previous work.

1. Introduction

Ensuring the authenticity and provenance of digital visual content is increasingly critical as AI-generated media becomes more prevalent. Traditional watermarking methods have limited robustness to everyday transformations and raise security concerns if models are distributed for local use. C2PA [20] does not work in practical scenarios, due to the signature becoming invalid upon any image transformation. Proof of transformation helps maintain validation after image edits [28, 35], but we do not expect many image editors to integrate this technology. To tackle these challenges, we introduce a three-part framework aimed at secure and reliable content provenance detection for AI-generated content in real world.

First, we introduce a perceptual hashing algorithm based on DINOV2 [68], a feature extraction network, that captures the semantic and structural features of images, providing an alternative to watermarking by remaining resilient to common image transformations. Second, to secure the integrity of the provenance detection process, we employ Mutli-Party Fully Homomorphic Encryption (MP-FHE) [65], which allows operations on encrypted data, enabling secure, privacy-preserving computations without compromising the system's security regarding data leakage [42]. Lastly, we extend prior work in AI-generated content detection by training a state of the art model to identify and distinguish AI-generated images directly, in case images are not found in our database.

Deep watermarking algorithms embed a nearly invisible hash into an image, which can later be extracted using a specialized network to verify and match images [93]. The main objectives of watermarking are (a) robustness against natural image distortions and (b) minimal alteration of the original image. Most deep watermarking papers report remarkable results on simple spatial distortions such as brightness or contrast, cropping, and flipping. However, they often fail when exposed to destructive frequency-domain transformations, including blurring, JPEG compression, screenshots, and other non-differentiable processes. This limitation arises because watermarking methods embed information in an image's high-frequency components, which are often degraded via compression during everyday manipulations like social media uploads [51], sharing on messaging apps [8], format conversions, resizing, and embedding in videos [18], making watermarking ineffective for content provenance detection in practical settings.

Traditionally, watermarking requires training a separate deep-learning model to embed watermarks into an image, making them easily removable. However, Meta's recently developed framework, Stable Signature [82], integrates watermarking directly into the diffusion model through fine-tuning, eliminating the need for an additional model during inference. While this approach reduces inference costs, it introduces the burden of maintaining a unique set of model weights for each user. Given that the number of customers of diffusion models is in millions, this storage requirement renders the approach impractical for large-scale industry deployment unless the model is provided to users for local inference. However, as the authors demonstrated, releasing the model to users introduces vulnerabilities due to aversion techniques like model purification and collusion, undermining the security and purpose of the watermarking system.

Additionally, watermarking methods might introduce unwanted visible artifacts [7]. In contrast, perceptual hashes offer a potential alternative. Instead of embedding data, perceptual hashing derives a hash from the image's semantic embeddings and structural features, which remain largely unaffected by common transformations, making it more suitable for content provenance applications [31].

After generating perceptual hashes, it's crucial to query them privately to protect sensitive information. For example, closed-source content generators (such as OpenAI or Canon) should be able to store perceptual hashes of user-generated images in a public database without leaking information that could lead to prompt reconstruction. Likewise, users need to query the database without revealing all of their images to the content provider. Several systems have been developed to address this private approximate nearest neighbors search (ANNS/PNNS) challenge, including Janus [36] and Worldcoin's Iris [14] matching system, Apple's Wally [9], and Panther [54]. The Worldcoin Iris Matching system uses secure multiparty computation (MPC) to perform privacy-preserving fuzzy matching of iris hash codes, which are similar to our proposed perceptual hashes. It operates under a non-collusion assumption between servers to ensure privacy. In contrast, Apple's Wally system introduces differential privacy by adding fake queries to hide the original query, and querying clusters of data rather than the entire database, thus trading off some accuracy for enhanced performance. Wally uses Somewhat Homomorphic Encryption (SHE) for secure lookups, based on the BFV [15, 37] and BGV [16] schemes, which allow a single entity to decrypt any ciphertext without restriction. In contrast, our system is built on a more advanced FHE scheme, Multi-party FHEW (MP-FHE), which requires multiple parties to participate in decryption [52, 65]. This enables the design of PNNS systems with stronger security guarantees, ensuring that no single entity can access plaintext data without the consent of all other parties. Modern tools and open-source models have democratized access to synthetic image generation. In this reality, it is unreasonable to expect every image appearing on the internet to be hashed and stored in a registry by some content-producing organization (CPO) like cameras or AI image APIs. So, we also build a purely deep learning-based detector to perform the binary classification task of detection of synthetic im-

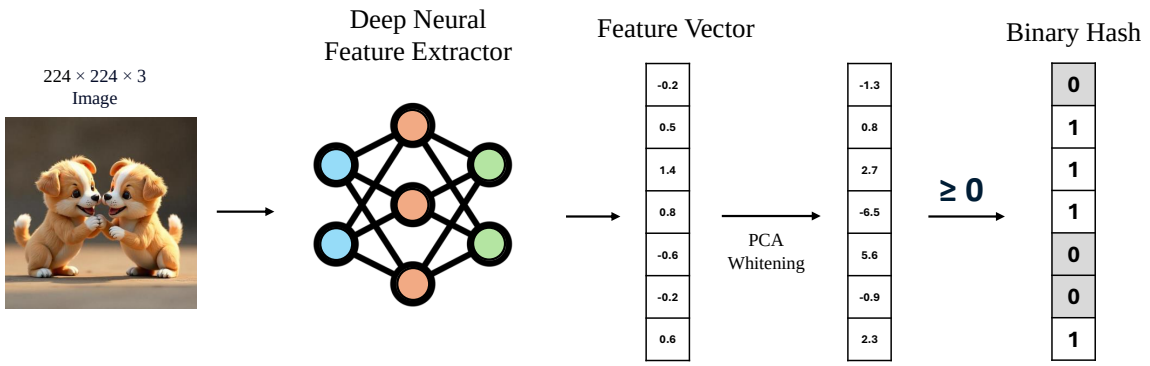

Figure 2. **Deep neural feature extractor architecture**. The pipeline consists of a feature extractor network and Locality Sensitive Hashing (LSH) step. The PCA whitening ensures that all components of the feature vector are uncorrelated so that the hashes of two random different images have the least probability of being the same.

ages.

In conclusion, we have four main contributions:

1. A deep perceptual hashing algorithm that is robust to a wide range of image transformations.
2. A novel approach to fine-tune any deep-learning-based perceptual hashing model to be adversarially robust. Although several works explore adversarial attacks against perceptual hashing models, to the best of our knowledge, we are the first to introduce an adversarial training mechanism for them.
3. A cryptographic matching system that privately looks up the provenance of a perceptual hash in a registry, without leaking any data about the registry items or the queried perceptual hash.
4. A deep learning based detector to detect synthetic images that are not stored in the registry.

2. Background

In this section, we discuss related work and the essential concepts used our framework. We provide an overview of perceptual hashing techniques, MPC methods, and deep learning-based detectors.

2.1. Perceptual Hashing

Since the MP-FHE scheme requires fuzzy hashes, we require a hashing algorithm that is mostly invariant to perceptual transformations of the image. Several perceptual hashing algorithms exist including aHash (Average Hashing), mHash (Median Hashing), dHash (Difference Hashing), bHash (Block Hashing), wHash (Wavelet Hashing) and the Discrete Fourier Transform Perceptual Hash [89]. These hashing schemes are used by services like Microsoft's PhotoDNA and Facebook's PDQ. These hashing algorithms work by dividing images into squares, converting them into a black-and-white format, and quantifying the shading of the squares in different ways. Although these hashes are mostly invariant to color manipulation and noise, they fail when the images are subjected to crops. This is because cropping the image offsets the alignment of the square blocks, heavily altering the hash.

In recent years, numerous deep hashing algorithms based on convolutional neural networks have been developed [56, 58, 91, 92]. These methods rely on deep neural networks to extract image features and subsequently compute a hash value based on those features. The resulting hashes capture the semantic content of the image and exhibit greater robustness to transformations like cropping and color adjustments. Apple's Neuralhash [79] was developed for Child Sexual Abuse Material (CSAM) detection. Figure 2 illustrates one variation of these deep hashing algorithms, which we have adopted and will now discuss in more detail.

A perceptual hash function $H\colon \mathbb{R}^{H\times W\times C} \to \{0,1\}^k$ maps an image x to a k-bit binary hash. Let $H(x)$ denote the hash function, where $H(x) = \big(h_1(x), \ldots, h_k(x)\big)$. These algorithms consist of two main components. First, a deep feature extractor $D(x)\colon \mathbb{R}^{H\times W\times C} \to \mathbb{R}^k$ generates a feature vector $z \in \mathbb{R}^k$ from the image x. This feature vector numerically represents the semantic and structural content of the image. Next, locality-sensitive hashing (LSH) [48, 85] is used to assign similar feature vectors to buckets with similar hash values. For each of the k features, the bit value $h_i(x)$ is set to either 1 or 0 depending on the sign of z_i using the heaviside step function.

The idea is that transformations applied to the original image do not perturb the feature vector and, consequently, the hash significantly. We would like it if each $h_i(x)$ follows an i.i.d Bernoulli distribution with parameter 0.5. This is achieved by applying PCA whitening to the feature vector before the LSH step. The statistics needed to compute the PCA weights are obtained by using feature vectors of images from the training data. Now, since each component

of the feature vector is independent and has an equal chance of being positive or negative after the binarization step, two randomly chosen images have the least possible probability of having the same hash, specifically 2^{-k}. To compare the similarity and distance between two hashes, we look at the Hamming distance (L_1 distance) between them. We quantify the degree of match between two hashes using the number of bits matched denoted by M where

$$M(H(x), H(x')) = k - L_1(H(x), H(x'))$$

An adversarial attack involves adding a imperceptible amount of crafted perturbation of magnitude ϵ (typically $< 8/255$) to each pixel of an image before passing it into the model [81], which causes it to behave against its objective. This perturbation is generally created using information obtained from the model's gradients (white box attacks), which requires complete access to the model's architecture and weights. Other forms of attacks either require access to the model's dataset or output logits (gray box attacks) [41, 44, 69], or only query access (black box attacks). There are two kinds of attacks relevant to perceptual hashes, hash collisions and hash aversions. A hash aversion attack would correspond to a malicious individual perturbing the original image x to generate an image x' that minimizes the similarity score M, leading to a false negative detection. A hash collision attack can by synthesized when an adversary has access to a a set of hashes stored by the CPO. This involves attacking a query image so that its hash matches one stored in the database, leading to a false positive detection. Adversarial training alone makes it extremely difficult for an adversary to perform either attack, even with full access to the model's weights. Furthermore, hash collision attacks are impossible to achieve in our framework due to the implementation of the secure MP-FHE scheme, which prevents the adversary's access to the perceptual hashes, even in the case of a data leak.

Bhatia and Meng [13], Struppek et al. [79] have successfully demonstrated that both hash collisions and hash aversions can be performed against deep hashing networks, highlighting NeuralHash as a specific example. However, it is noteworthy that NeuralHash did not deploy any system to avoid these attacks. Adversarial training combined with the MP-FHE scheme secures our system against both attacks.

2.2. Multi-Party Fully Homomorphic Encryption

Multi-party Fully Homomorphic Encryption (MP-FHE) is a cryptographic technique that extends the capabilities of single-party homomorphic encryption to multiple participants. It enables computations being executed on encrypted data without revealing their individual inputs, while the decryption requires parties to perform collaboratively decrypt a result once the computation is complete. The advantage of this approach lies in its ability to preserve privacy while performing complex computations, such as those in secure MPC [77]. Data confidentiality is an important goal in applications such as privacy-preserving data analysis [65].

MP-FHE also builds upon the concept of threshold cryptography, where the decryption process requires the collaboration of a threshold number of parties. This ensures that no single party can decrypt the data independently, enhancing security in scenarios involving untrusted parties. The MP-FHE scheme involves distributing the secret key across parties and utilizing homomorphic properties to perform computations on ciphertexts. Compared to traditional MPC methods like secret sharing, MP-FHE reduces communication overhead, as computations can be performed non-interactively on encrypted data, making it an efficient solution for large-scale, privacy-preserving applications [52, 65].

2.3. Deep Learning Based Detectors

Synthetic images have artifacts that are distinctive to the generation process [62, 88]. Existing detectors often exploit these fingerprints in spatial [23, 32, 49, 57, 63, 78] or frequency [33, 34, 40] domains to determine whether the image is synthetic or real. This method has two problems: a) Detectors trained on one type of generator do not generalize well on images from unseen generators, and b) transformations like resizing and compression destroy these fingerprints [22]. Recent work exploits VLMs like CLIP [71] as feature extractors [6, 24, 67, 76] and train a classifier to detect synthetic images. Cozzolino et al. [24] clearly shows these models are more accurate and robust under transformations. This work inspired our detector.

3. Methodology

The core database in the system is a content registry, indexed by content-producing organization (CPO) with the perceptual hashes of content. The lifecycle of watermark creation consists of the content producing organization calculating perceptual hashes for all their content, encrypting those perceptual hashes to an MP-FHE encryption key for the database, and then digitally signing those encrypted perceptual hashes to establish provenance. Then, when users want to query the provenance of some potentially modified image, they request the parties running the MP-FHE database to find the images with the lowest Hamming distance, then are returned the most likely sources for the image. If no nearby matches are found, then the system falls back to the naïve AI image classifier model.

3.1. Perceptual Hash Design

Oquab et al. [68] developed DINOv2, a self-supervised feature-extracting ViT network that is trained to generate feature vectors invariant to crops and natural perturbations like color jittering, gaussian blur and solarization.

We use the pre-trained DINOv2 ViT-B/14 (with register tokens) to extract feature embeddings of the given image. Then, we apply PCA to de-correlate each component of the feature vector and simultaneously reduce the dimensionality of the feature vector from 768 to 96, then apply the LSH step to obtain a 96-bit binary string. Naturally, the resulting binary string serves the purpose of a perceptual hash since the original feature vector is highly robust to perceptual transformations of the image. We also implement our own adversarial training scheme, which is discussed in detail in section 9.

Statistical Test. Let $H \in \{0,1\}^k$ be the hash of an image x in the database of the CPO that we wish to compare the query image against and, H', be the corresponding hash of a query image x'. The detection test compares the number of matching bits between H and H' to a threshold, i.e. if

$$M(H, H') \geq \tau \quad \text{where} \quad \tau \in \{0, \ldots, k\}, \tag{1}$$

then the two images are considered to be the same.

For the case where x and x' do not correspond to same image, we assume that bits $h_1, \ldots, h_k$ are i.i.d. Bernoulli random variables with parameter 0.5. Then $M(H, H')$ follows a binomial distribution with parameters $(k, 0.5)$.

The False Positive Rate (FPR) is the probability that $M(H, H')$ takes a value bigger than the threshold τ.

$$\text{FPR}(\tau) = P(X \geq \tau), \quad X \sim \text{Binomial}(k, 0.5) \tag{2}$$

Varying τ necessitates a trade-off between the False Negative Rate (FNR) and the FPR, which can be set according to the sensitivity of the use case.

3.2. Preserving Privacy of End-Users

Problem Definition. In many real-world scenarios, maintaining a perceptual hash database poses significant privacy risks. Such a database could expose statistical information about end-users and potentially would lead to multiple privacy breaches. In our case, leaking a perceptual hash may allow partial reconstruction of an image, and therefore cause prompt leakage, which can reveal personally identifiable information (PII) [59]. For example, widely-used image generators like DALL-E are obliged to delete user data after up to 30 days, unless de-identified or retained for legal or security purposes [3]. In another example, OpenAI's Advanced Voice mode is unavailable in the EU because the voice data is eligible for zero retention [2]. Thus it is important to design a system for attesting the generated data by AI to preserve copy-right related concerns and both preserve the content owners privacy. Consequently, assuming a central and trustworthy database exists to compare new image queries is infeasible.

Design Principals. To address this challenge while still benefiting from a database-like structure, we need a system that can guarantee following:

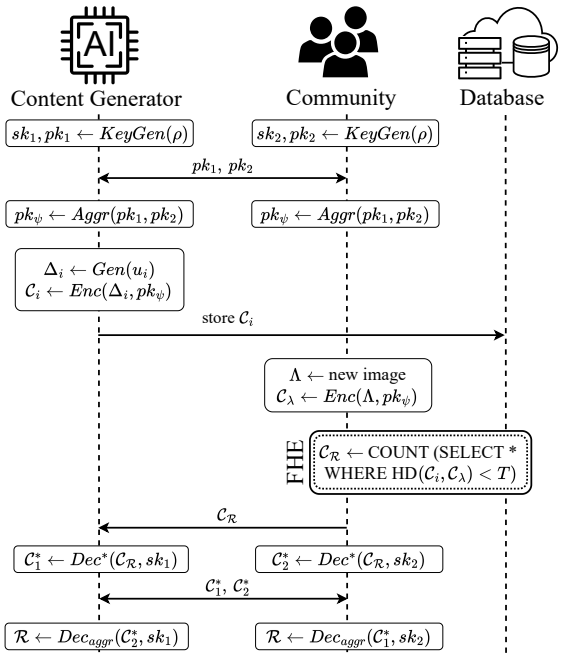

Figure 3. **Overview of the protocol.** The database is kept private, content generators can write encrypted data to it, and the public can run private queries that return only provenance data.

- **Data confidentiality:** No entity should gain direct access to any part of an image or information related to an image (such as pHash) at any time.
- **Query Leakage resilience:** No query or database operation should lead to any data being revealed or stored (temporary or permanently) on any device at any time.
- **Low Operational Risks:** Ideally, most of the heavy computations should require minimal to no security considerations, allowing the system to be trustlessly scaled up when needed.

Adversary Model. Recent studies have proposed various approaches to address this issue from different perspectives. Following practical protocols, we adopt a probabilistic polynomial-time (PPT) adversary model, wherein any entity in the protocol may eavesdrop on or manipulate accessible data, though their computational power remains polynomially bounded. Our constructions rely on several cryptographic primitives, which are assumed to be secure against any PPT adversary. Specifically, we assume that fully homomorphic encryption (FHE) [5, 26, 52], MP-FHE [65], collision-resistant hash functions, and digital signatures remain unbroken in this model.

Protocol Details Figure 9 illustrates a scenario involving two parties (for simplicity) executing a secure protocol: one as the content creator (the owner of generative model) and the other as a regulator or trusted entity (a community representative). Each party possesses their own set of private and public keys. The protocol operates in the following four

phases:

- **Setup Phase:** Parties exchange shares of their public keys to create an aggregated m-of-n MPC public key. The shared key is used universally to encrypt pHash values. Note that all parties generating and signing data are incentivized to contribute up to $(n - m)$ live servers to the protocol, so that they can help ensure that they do not decrypt values in the database. More servers means they could affect liveness, and less servers means they have less control over decryption.
- **Encryption Phase:** pHash value of each image is encrypted using the aggregated public key (P_{shared}) and stored in database. Original image is then destroyed in a verifiable manner. To enhance the accuracy of future claims of copyright violations, parties may decide to also store MPC-encrypted version of each image for a ceremony-based opening under specific conditions, like when pHash matches show very low hamming distance.
- **Evaluation Phase:** When a new image is submitted for evaluation, the pHash value of the image is encrypted using the same shared key (P_{shared}) and undergoes a fully homomorphic computation of the Hamming distance across the database. The result is an encrypted output indicating the number of pHash values with close Hamming distances. The executed FHE function is merely an example and may vary depending on the specific scenario, database size, or the quality of the pHash values.
- **Decryption Phase:** To decrypt evaluation results, both parties exchange decryption shares in an MPC process. The final value reveals whether the image has any overlaps in the database (non-zero result) or not (zero result).

We implemented our proof-of-concept (PoC) for the proposed system using the *Phantom-Zone* MP-FHE library[1] in FHEW and link to a comparable implementation of dot product in BFV in Lattigo[2].

Method Complexity. The employed MP-FHE scheme supports Boolean operations on encrypted data. For simplicity, we assume each bit of the pHash values is encrypted individually[3]. Thus, for pHash values of length l, we have l ciphertexts. However, these ciphertexts are further compressed into a single ciphertext to minimize computational complexity, requiring only a few kilobytes of storage. Our MP-FHE setup includes a parallel XOR array of length l and a bit-counter of size l, which outputs the count of 1s in an encrypted value. A comparator of size $\log l$ then compares this count with an encrypted threshold value, $Enc(\tau)$, and outputs an encrypted bit: it is 0 if the Hamming distance exceeds the threshold, indicating a difference above this limit. This comparison process is repeated in parallel for every encrypted value in the registry. Finally, an OR tree with n leaves calculates the OR value of all comparisons, where n is the number of registry entries.

One of the key security guarantees of the protocol is that the queries do not leak any information about the dataset, beyond confirming whether the pHash of the query image shares significant similarities with any value in the dataset. To achieve this, we define a set of valid queries that are allowed, with the database managed in an MPC manner.

3.3. Detector for Unknown Images

Architecture Open Clip [47] has trained various contrastive image-language models with better zero-shot performance on ImageNet [29] than OpenAI's open-sourced version. So, we choose four image encoders for our experiments: SigLIP [90] models with SoViT-400M [4] backbone with an input resolution of 224 and 384 and CLIP models with ViT-H [30] backbone with an input resolution of 224 and 378. We use SVM as our classifier and the second last layer to extract embeddings following [24].

Method We perform two sets of experiments. 1) We use the better encoders from Open CLIP and train an SVM on the features extracted from their visual encoders. 2) We finetune the entire encoder backbone along with the SVM layer. We follow the technique proposed by Wortsman et al. [86] and interpolate between the finetuned and original encoder weights for robustness. We find that the training loss converges significantly within the first two epochs. So, we finetune each model for five epochs and use the validation set to choose the epoch, after which we get the best-performing model. Table 2 details the number of epochs for which each of the four encoders that were finetuned.

4. Experimental Results

In this section, first we evaluate the robustness and accuracy of our proposed DinoHash and DL detector. Finally, we evaluate the overhead of finding the best matches for encrypted pHash values (using FHEW) queried against the encrypted database. To do this, we have implemented a proof-of-concept of the protocol using the Phantom Zone FHEW library [1]. We benchmark our implementations across different setups, ranging from midrange laptops to servers.

4.1. DinoHash

We benchmark DinoHash against other prominent methods like NeuralHash, Stable Signature and DCT-DWT across multiple transformations designed to mimic typical image edits. We run all experiments on a single NVIDIA GPU A100 GPU with 40GB of VRAM. We present our results for four different complex transformations, each transformation consists of a base transformation of a 20% crop from each side followed by a JPEG compression with a quality

[1] https://github.com/gausslabs/phantom-zone

[2] https://eprint.iacr.org/2024/1774.pdf

[3] In practice, multiple bits are compressed into a single ciphertext to reduce communication and computation complexity

Figure 4. Transformations used to benchmark hashing and watermarking methods. With screenshot simulation as pre-processing, and random erasing followed by text overlay as post-processing.

factor of 30%. This simulates an effect similar to a screenshot. Then we choose one of the following four different operations:

1. a brightness shift by a random factor between $\pm 30\%$
2. a contrast shift by a random factor between $\pm 30\%$
3. gaussian blurring with a kernel width of 2
4. application of a median filter with $k = 3$

These represent common image transformations and filters found in most image editing applications. To ensure even stronger robustness against complex image distortions: we then apply a random-erasing augmentation, where a square area with a side length 20% of the image is erased; followed by a random text overlay with a random string of 10 alphanumeric characters. A visualization of these transformations can be found in Figure 4.

4.2. Distortion Robustness

We use 1M randomly sampled images from the DiffusionDB dataset [84] to carry out experiments for DCT-DWT watermarking, NeuralHash and our framework. We use 1M randomly sampled prompts of the dataset to generate images to benchmark Stable Signature.

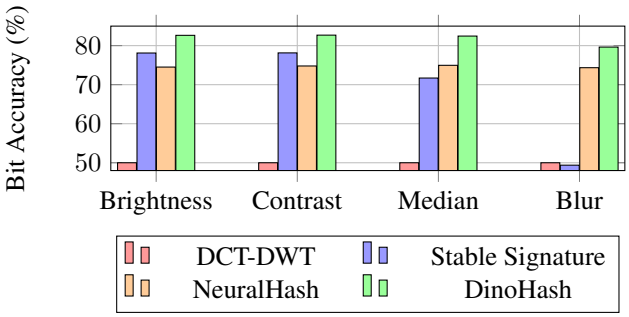

Figure 5. **Bit Accuracy Comparison.** The mean bit accuracy of different methods after a simulated screenshot followed by the respective transformations.

Figure 6 shows the Receiver Operating Characteristic (ROC) Curve for different hashing/watermarking methods under different transformations. The level of τ is varied from 0 to 96 to obtain different values for the TPR and FPR. The TPR is calculated by measuring the fraction of images whose degree of match, M, after the transformation equals or exceeds τ. Conversely, the FPR is approximated using Equation 2 since it is too small to be measured experimentally. Figure 5 reports results on the mean bit accuracy of the algorithms. We do not include the TPR-FPR plot for DCT-DWT due to its near-random performance.

We observe that all algorithms are roughly equally robust to the brightness and contrast transformations due to their non-destructive nature. We also observe that DinoHash is significantly more robust to transformations than NeuralHash and Stable Signature across both metrics. We do not report the results for the vanilla (no transformation) case since any hashing method would trivially achieve a 100% match in the cases without perturbations. This, however, does not always hold for watermarking methods since the watermark generation and extraction models may not always agree. Furthermore, all watermarking methods report the PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index) of the watermarked images. These metrics measure how perceptually similar the watermarked and the unwatermarked images are. Since hashing algorithms do not cause any modification to the underlying image, we achieve perfect PSNR and SSIM scores. Since NeuralHash is a proprietary algorithm, it is difficult to comment on why DinoHash performs better.

4.3. Experiments on DL detector

We fix a consistent setup to simulate real world conditions on a challenging dataset with postprocessing steps to evaluate our models as accurately as possible.

Dataset We report all results on the recently proposed Synthbuster dataset [11] as it has images from widely used commercial tools, i.e. Dalle2 [73], Dalle3 [12], Midjourneyv5 [64] and Adobe Firefly [39]. It uses RAISE-1k [27] as its

Method	Backbone	Resolution	DALL-E 2	DALL-E 3	Firefly	Midjourney v5	AVG
Cozzolino et al. [24]	-	-	0.706 / **0.612**	0.794 / 0.639	0.766 / 0.635	0.698 / 0.602	0.741 / 0.622
Corvi et al. [22]	-	-	0.689 / 0.502	0.287 / 0.500	0.574 / 0.501	0.481 / 0.500	0.508 / 0.501
Ours-frozen							
DFN [38]	ViT-H	378	0.767 / 0.584	**0.999 / 0.980**	**0.936 / 0.754**	**0.949 / 0.804**	**0.913 / 0.780**
DFN [38]	ViT-H	224	*0.792* / *0.605*	0.997 / *0.974*	0.921 / 0.720	0.934 / 0.771	*0.911* / 0.767
SigLIP[90]	So-ViT-400M	224	0.663 / 0.574	0.996 / 0.965	0.896 / 0.730	0.909 / 0.785	0.866 / 0.764
SigLIP[90]	So-ViT-400M	384	0.731 / 0.600	0.997 / 0.972	*0.924* / *0.742*	0.926 / 0.772	0.895 / *0.771*
Ours-finetuned							
DFN [38]	ViT-H	378	0.742 / 0.534	0.991 / 0.965	0.870 / 0.585	0.906 / 0.706	0.877 / 0.697
DFN [38]	ViT-H	224	**0.818** / 0.559	*0.998* / 0.972	0.910 / 0.623	0.880 / 0.689	0.901 / 0.711
SigLIP[90]	So-ViT-400M	224	0.781 / 0.542	0.997 / 0.966	0.887 / 0.620	0.937 / 0.737	0.901 / 0.716
SigLIP[90]	So-ViT-400M	384	0.772 / 0.592	0.993 / 0.966	0.903 / 0.664	*0.938* / 0.777	0.902 / 0.750

Table 1. AUC/Accuracy scores for various methods across different commercial tools with average values. Accuracy values here are calculated with threshold of 0.5. **Bold** represent highest value in a column and *italics* represent second highest value.

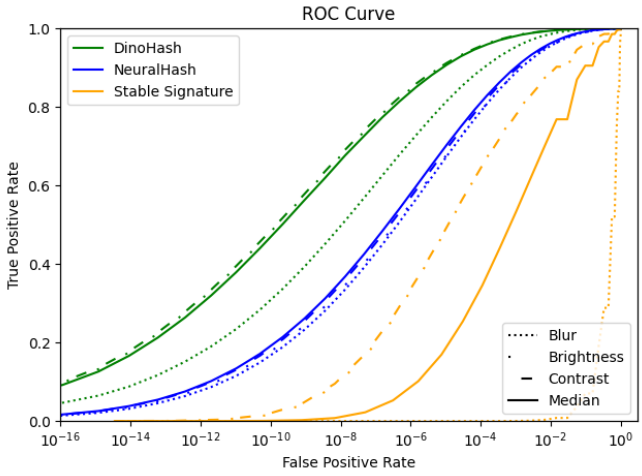

Figure 6. **Detection Results.** The TPR/FPR detection curve of different methods after a simulated screenshot followed by the respective transformations, random erasing and text overlay.

real part. Each source contains 1000 images. We use 10000 pairs of real and synthetic images for all training purposes. This set is created using the strategy proposed in [24]. We randomly select 10000 images from the train set of COCO-2017[55], generate their caption using BLIP [53], and generate images for these captions using Stable Diffusion XL [70]. We use a validation set of 500 pairs of images created similarly using the validation set of COCO-2017.

Transformations In real life, images often suffer cropping and compression when shared. We perform a set of transformations on all images to make the model more robust and to test its robustness under such attacks. We perform random crop to 62.5% to 100% of the original scale followed by resizing to 200x200 pixels. We then perform random jpeg compression to between 85% to 100% of initial quality.

Testing Protocol Our testing dataset contains synthetic images from four different sources having 1000 images each and a real set of 1000 images. While computing metrics for one synthetic image source, we consider just that subset of dataset and the real images. So, we effectively have 1000 synthetic and real images, making the dataset balanced.

Method	Backbone	Resolution	Epoch	LR	Batch Size
SigLIP [90]	So-ViT-400M	384	5	5e-6	8
SigLIP [90]	So-ViT-400M	224	3	5e-6	8
DFN [38]	ViT-H-14	378	4	5e-6	8
DFN [38]	ViT-H	224	3	5e-6	8

Table 2. Training configurations for various methods, including backbone, resolution, epochs, and learning rate.

Metrics We use area under ROC curve (AUC) and accuracy(ACC) with threshold of 0.5 as our metric. We use threshold of 0.5 to simulate a sitution where we have no prior information about the detector.

Results Our models outperform the previous SoTA [24] across both metrics. Linear SVM trained on top of frozen CLIP image encoder with ViT-H backbone, 378 input resolution and quickGELU activation function yields best results on most generators and on average. It shows improvements of 23.3% / 25.4% in AUC/ACC respectively over previous SoTA. Other models also perform better than Cozzolino et al. [24], we present detailed results in Tab. 1.

4.4. Searching pHash in the Database

We have implemented our protocol as a PoC in an open-source library, available on GitHub. For this, we utilize Phantom Zone [1], an experimental library designed for the practical realization of MP-FHE. In our benchmarking scenario, the pHash values of the images stored in the database are encrypted using a shared key between two parties. This allows us to efficiently query and match encrypted pHash

values, demonstrating the feasibility of our protocol in real-world applications.

In our implementation, we use the `FheBool` class from the PhantomZone library to perform homomorphic boolean logic. In this approach, each bit of the vectors is encrypted individually. As a result, encrypting a 96-bit vector involves creating 96 individual ciphertexts. A key advantage of treating each bit separately is the potential for parallelization during the evaluation phase of MP-FHE, which significantly speeds up the computation.

When a specific pHash value is queried, it is first encrypted locally by the client before being sent to the server for comparison with the encrypted database. The server then performs an XOR operation between the queried pHash value and each database entry. This operation is executed bit by bit and in parallel across all bits of the vectors. After the XOR operations are completed, the Hamming distance for each result is computed. This step is optimized using a 6-layer parallel boolean logic adder tree, which efficiently calculates the Hamming distance for all entries. If the distance for an entry is below the specified threshold (e.g., 8 bits), the result for that entry is set to True (1); otherwise, it is set to False (0). Once all entries are processed, the results are combined using a parallel OR tree. The final result is then sent back to the querying party for partial decryption. If the decrypted result is 1, it indicates that the queried pHash value is sufficiently similar to one or more entries in the encrypted database.

Table 3 presents the average latency cost of determining whether two encrypted 96-bit vectors are close in FHEW, measured on a dataset with 1,000 entries. The table reports a cost breakdown of each phase of the FHEW evaluation. The columns "XOR," "HD," and "Full Query" represent the average latency of calculating the XOR value, Hamming distance, and determining whether two 96-bit encrypted vectors are close or not, respectively. We conducted experiments on three different machines: 1) A midrange Dell Latitude 5531 with a 12^{th} Gen Intel Core-i5 processor, 2) A Macbook Pro with a 10-core Apple Silicon M4 processor, and 3) A Google Cloud server equipped with 56 vCPUs and 112 GB memory running on an AMD EPYC 7B13 processor.

One of the key features of the proposed method is that each entry in the data is homomorphically encrypted using the same multi-party key. Consequently, FHE evaluations can be trustlessly distributed across multiple untrusted parties, as no single entity can leak or decrypt any data during query execution without access to all shared key fragments. This property enables the system to scale with minimal security considerations, making it cost-efficient.

Although our current PoC implementation results indicate that a CPU-only device (without GPU acceleration) requires over 100 seconds to execute a query on a database with only 1,000 entries, the system can be scaled inexpensively by distributing computations across multiple trustless parties. The results from each party can then be merged using a parallel OR tree. The concept of clustering databases has also been utilized in previous work to enhance scalability [9].

Related work. There are two categories of related work relevant to our constructions. The first group focuses on preserving the privacy of user queries against a database that is visible to the query executor [9, 45]. In such setups, it is crucial to ensure that the database owner cannot infer any information about the user's queries.

However, our system model is fundamentally different from these works. Specifically, in our target application, the database contains confidential user history that must remain undisclosed to any entity at all times. The key distinction is that, in our setting, even the data owner should not be able to reconstruct the query database. This constraint makes our protocol and the underlying problem significantly more complex than those addressed in [9, 45].

Another line of work targets the same type of scenario as ours, where the database must remain unreconstructible due to its sensitive nature of stored data. The most notable recent work in this category are [14, 54], in which the authors propose secretly sharing the original database among multiple parties, enabling user queries to be executed in an MPC-based manner. However, our MP-FHE-based protocol achieves stronger security guarantees. Notably:

- Query execution can be performed by any party, not just the shareholders.
- There are no costs associated with maintaining database shares, as the entire database remains encrypted and does not leak any information.
- No interaction is required among shareholders during query execution; the only interaction occurs when decrypting the final query result. This results in significantly lower communication complexity as pointed in recent work, such as [54].

However, it is important to acknowledge that a pure MPC-based setup generally achieves higher performance compared to our FHE-based approach [54]. Therefore, while our construction represents an ideal and highly secure solution, it may not be practical for large-scale deployments. For this reason, we also extend and refine the protocol of [14], as discussed in Appendix 10.

5. Limitations and Future Work

Incorporating FHE into the querying process increases computational overhead, which could impact performance in real-time applications. Several algorithms, similar to KD-Trees, allow indexing for nearest-neighbor search of

System	CPU	Cores	XOR	HD	Full Query
Laptop	12^{th}Gen i5	4	200 ms	1.29 s	1.3 s
Laptop	12^{th}Gen i5	8	105 ms	747 ms	773 ms
Laptop	12^{th}Gen i5	16	67 ms	467 ms	472 ms
MB Pro	Apple M4	10	59 ms	421 ms	440 ms
Server	AMD 7B13	56	26 ms	134 ms	137 ms

Table 3. The average latency cost of determining whether two encrypted 96-bit vectors are close in FHEW (Measured on a Dataset with 1,000 entries).

multi-dimensional vectors, however these frameworks cannot be used in conjunction with FHE schemes and keep perfect privacy. Future research could focus on optimizing FHE-based indexing methods for quick retrieval to meet computational demands, or augmenting the Worldcoin Iris matching system or Apple's Wally for our usecase.

One advantage of watermarking over hashing is that watermarking only requires storing *one hash per image creator* for provenance, as it needs only to identify the image's creator, not each specific image. In contrast, perceptual hashing requires storing *one hash per image*, which increases the search space and reduces computational efficiency in terms of memory usage and speed. However, in cases where each user generates only one image, the memory and computation requirements for nearest-neighbor searches are equivalent for both methods.

Publicly releasing deep watermark encoders or decoders pose huge security risks, as that allows images to be dewatermarked easily. On the contrary, even though deep hashing methods are typically vulnerable to adversarial attacks, these risks can be mitigated through adversarial defense mechanisms so that our framework maintains robustness even against informed attacks, i.e. when the adversary has full access to the model weights, dataset, and components of the defense framework. Future work could include combining adversarial training techniques—such as Ensemble Adversarial Training [80], Adversarial Logit Pairing [50], and PGD Adversarial Training [60]—with input transformation defenses like DefenseGAN [74], BaRT [72], Feature Squeezing [87] and Randomized Smoothing [21].

6. Conclusion

Our three-part framework addresses significant challenges in AI content provenance detection by combining perceptual hashing, fully homomorphic encryption (FHE), and a robust AI-content detection model. Our perceptual hashing method with DINOV2 overcomes the limitations of traditional watermarking, by capturing semantic and structural image features that persist through common transformations, without altering the base image. Applying FHE within this framework ensures that users can query content databases securely, preserving privacy without compromising the system's integrity, even if some data is exposed. Finally, the AI-generated content detection model increases the framework's robustness by identifying AI-generated images that may be generated outside the known database.

References

[1] phantom-zone. https://github.com/gausslabs/phantom-zone. Accessed: 2024-10-26. 6, 8

[2] How we use your data. https://platform.openai.com/docs/models/how-we-use-your-data, 2023. Accessed: 2024-10-08. 5

[3] Data usage for consumer services faq. https://help.openai.com/en/articles/7039943-data-usage-for-consumer-services-faq, 2024. Accessed: 2024-10-26. 5

[4] Ibrahim M Alabdulmohsin, Xiaohua Zhai, Alexander Kolesnikov, and Lucas Beyer. Getting vit in shape: Scaling laws for compute-optimal model design. *Advances in Neural Information Processing Systems*, 36, 2024. 6

[5] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. Homomorphic encryption standard. *Protecting privacy through homomorphic encryption*, pages 31–62, 2021. 5

[6] Roberto Amoroso, Davide Morelli, Marcella Cornia, Lorenzo Baraldi, Alberto Del Bimbo, and Rita Cucchiara. Parents and Children: Distinguishing Multimodal DeepFakes from Natural Images. *arXiv preprint arXiv:2304.00500v1*, 2023. 4

[7] Bang An, Mucong Ding, Tahseen Rabbani, Aakriti Agrawal, Yuancheng Xu, Chenghao Deng, Sicheng Zhu, Abdirisak Mohamed, Yuxin Wen, Tom Goldstein, and Furong Huang. Waves: Benchmarking the robustness of image watermarks, 2024. 2

[8] Fahmi Anwar, Abdul Fadlil, and Imam Riadi. Image quality analysis of png images on whatsapp messenger sending. *Telematika*, 14(1):1–12, 2021. Accreditated SINTA "2" Kemenristek/BRIN, No. 85/M/KPT/2020. 2

[9] Hilal Asi, Fabian Boemer, Nicholas Genise, Muhammad Haris Mughees, Tabitha Ogilvie, Rehan Rishi, Guy N. Rothblum, Kunal Talwar, Karl Tarbe, Ruiyu Zhu, and Marco Zuliani. Scalable private search with wally, 2024. 2, 9

[10] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, 2018. 3

[11] Quentin Bammey. Synthbuster: Towards detection of diffusion model generated images. *IEEE OJSP*, 2023. 7

[12] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, Wesam Manassra, Prafulla Dhariwal, Casey Chu, and Yunxin Jiao. https://openai.com/dall-e-3, 2023. 7

[13] Jagdeep Singh Bhatia and Kevin Meng. Exploiting and defending against the approximate linearity of apple's neuralhash, 2022. 4

[14] Remco Bloemen, Bryan Gillespie, Daniel Kales, Philipp Sippl, and Roman Walch. Large-scale MPC: Scaling private iris code uniqueness checks to millions of users. Cryptology ePrint Archive, Paper 2024/705, 2024. 2, 9

[15] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual cryptology conference*, pages 868–886. Springer, 2012. 2

[16] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6 (3):1–36, 2014. 2

[17] Lucy Chai, David Bau, Ser-Nam Lim, and Phillip Isola. What makes fake images detectable? understanding properties that generalize. In *Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part XXVI 16*, pages 103–120. Springer, 2020. 1

[18] Xianhui Che, Barry Ip, and Ling Lin. A survey of current youtube video characteristics. *IEEE MultiMedia*, 22(2):56–63, 2015. 2

[19] Jiaxuan Chen, Jieteng Yao, and Li Niu. A single simple patch is all you need for ai-generated image detection. *arXiv preprint arXiv:2402.01123*, 2024. 1

[20] Coalition for Content Provenance and Authenticity. C2PA technical specification. *Content Authenticity Initiative*, 2023. 2

[21] Jeremy M Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing, 2019. 10

[22] Riccardo Corvi, Davide Cozzolino, Giada Zingarini, Giovanni Poggi, Koki Nagano, and Luisa Verdoliva. On the detection of synthetic images generated by diffusion models. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023. 4, 8

[23] Davide Cozzolino, Justus Thies, Andreas Rössler, Christian Riess, Matthias Nießner, and Luisa Verdoliva. ForensicTransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510v2*, 2018. 4

[24] Davide Cozzolino, Giovanni Poggi, Riccardo Corvi, Matthias Nießner, and Luisa Verdoliva. Raising the Bar of AI-generated Image Detection with CLIP. In *CVPR Workshops*, 2024. 4, 6, 8, 1, 2

[25] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 37th International Conference on Machine Learning*. JMLR.org, 2020. 2

[26] Ivan Damgard, Martin Geisler, and Mikkel Kroigard. Homomorphic encryption and secure comparison. *International Journal of Applied Cryptography*, 1(1):22–31, 2008. 5

[27] Duc-Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter, and Giulia Boato. Raise: A raw images dataset for digital image forensics. In *Proceedings of the 6th ACM multimedia systems conference*, pages 219–224, 2015. 7

[28] Trisha Datta, Binyi Chen, and Dan Boneh. VerITAS: Verifying image transformations at scale. Cryptology ePrint Archive, Paper 2024/1066, 2024. 2, 4

[29] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 6

[30] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 6

[31] Ling Du, Anthony T.S. Ho, and Runmin Cong. Perceptual hashing for image authentication: A survey. *Signal Processing: Image Communication*, 81:115713, 2020. 2

[32] Mengnan Du, Shiva Pentyala, Yuening Li, and Xia Hu. Towards Generalizable Deepfake Detection with Locality-Aware AutoEncoder. In *CIKM*, page 325–334, 2020. 4

[33] Ricard Durall, Margret Keuper, and Janis Keuper. Watch Your Up-Convolution: CNN Based Generative Deep Neural Networks Are Failing to Reproduce Spectral Distributions. In *CVPR*, pages 7890–7899, 2020. 4

[34] Tarik Dzanic, Karan Shah, and Freddie D. Witherden. Fourier spectrum discrepancies in deep network generated images. In *NeurIPS*, pages 3022–3032, 2020. 4

[35] Stefan Dziembowski, Shahriar Ebrahimi, and Parisa Hassanizadeh. VIMz: Verifiable image manipulation using folding-based zkSNARKs. Cryptology ePrint Archive, Paper 2024/1063, 2024. 2, 4

[36] Kasra EdalatNejad, Wouter Lueks, Justinas Sukaitis, Vincent Graf Narbel, Massimo Marelli, and Carmela Troncoso. Janus: Safe biometric deduplication for humanitarian aid distribution. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 655–672. IEEE, 2024. 2

[37] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012. 2

[38] Alex Fang, Albin Madappally Jose, Amit Jain, Ludwig Schmidt, Alexander Toshev, and Vaishaal Shankar. Data filtering networks. *arXiv preprint arXiv:2309.17425*, 2023. 8, 1

[39] Adobe Firefly. https://www.adobe.com/sensei/generative-ai/firefly.html, 2023. 7

[40] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging Frequency Analysis for Deep Fake Image Recognition. pages 3247–3258, 2020. 4

[41] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery. 4

[42] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009. 2

[43] Diego Gragnaniello, Davide Cozzolino, Francesco Marra, Giovanni Poggi, and Luisa Verdoliva. Are gan generated images easy to detect? a critical analysis of the state-of-the-art. In *2021 IEEE international conference on multimedia and expo (ICME)*, pages 1–6. IEEE, 2021. 1

[44] Zecheng He, Tianwei Zhang, and Ruby B. Lee. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*, page 148–162, New York, NY, USA, 2019. Association for Computing Machinery. 4

[45] Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, and Nickolai Zeldovich. Private web search with tiptoe. In *Proceedings of the 29th symposium on operating systems principles*, pages 396–416, 2023. 9

[46] Chih-Hui Ho and Nuno Vasconcelos. Disco: Adversarial defense with local implicit functions. In *Advances in Neural Information Processing Systems*, pages 23818–23837. Curran Associates, Inc., 2022. 3

[47] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 6, 1

[48] Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, and Chidambaram Crushev. A survey on locality sensitive hashing algorithms and their applications, 2021. 3

[49] Hyeonseong Jeon, Young Oh Bang, Junyaup Kim, and Simon Woo. T-GD: Transferable GAN-generated Images Detection Framework. pages 4746–4761, 2020. 4

[50] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing, 2018. 10

[51] Asif Ali Laghari, Hui He, Muhammad Shafiq, and Asiya Khan. Assessment of quality of experience (QoE) of image compression in social cloud computing. *Multiagent and Grid Systems*, 14(2):125–143, 2018. Received: 13 September 2017; Accepted: 25 February 2018; Published: 26 June 2018. 2

[52] Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. Efficient fhew bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 227–256. Springer, 2023. 2, 4, 5

[53] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022. 8

[54] Jingyu Li, Zhicong Huang, Min Zhang, Jian Liu, Cheng Hong, Tao Wei, and Wenguang Chen. Panther: Private approximate nearest neighbor search in the single server setting. *Cryptology ePrint Archive*, 2024. 2, 9

[55] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 8

[56] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2475–2483, 2015. 3

[57] Bo Liu, Fan Yang, Xiuli Bi, Bin Xiao, Weisheng Li, and Xinbo Gao. Detecting generated images by real images. In *ECCV*, pages 95–110, 2022. 4

[58] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2064–2072, 2016. 3

[59] Jordan Madden. Assessing the adversarial security of perceptual hashing algorithms. *arXiv preprint*, 2024. arXiv:2406.00918. 5

[60] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019. 10

[61] Sara Mandelli, Nicolò Bonettini, Paolo Bestagini, and Stefano Tubaro. Detecting gan-generated images by orthogonal training of multiple cnns. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 3091–3095. IEEE, 2022. 1

[62] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do GANs Leave Artificial Fingerprints? pages 506–511, 2019. 4

[63] Francesco Marra, Cristiano Saltori, Giulia Boato, and Luisa Verdoliva. Incremental learning for the detection and classification of GAN-generated images. pages 1–6, 2019. 4

[64] Midjourney. https://www.midjourney.com/home, 2023. 7

[65] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty homomorphic encryption from ring-learning-with-errors. *Proceedings on Privacy Enhancing Technologies*, 2021(4):291–311, 2021. 2, 4, 5

[66] Dat Nguyen, Nesryne Mejri, Inder Pal Singh, Polina Kuleshova, Marcella Astrid, Anis Kacem, Enjie Ghorbel, and Djamila Aouada. Laa-net: Localized artifact attention network for quality-agnostic and generalizable deepfake detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17395–17405, 2024. 1

[67] Utkarsh Ojha, Yuheng Li, and Yong Jae Lee. Towards universal fake image detectors that generalize across generative models. In *CVPR*, pages 24480–24489, 2023. 4, 1

[68] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. In *International Conference on Computer Vision and Pattern Recognition*, 2023. 2, 4

[69] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks, 2016. 4

[70] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 8

[71] Alec Radford, JongWook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, and Jack Clark et al. Learning transferable visual models from natural language supervision. pages 8748–8763, 2021. 4, 1

[72] Edward Raff, Jared Sylvester, Steven Forsyth, and Mark McLean. Barrage of random transforms for adversarially robust defense. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6521–6530, 2019. 10

[73] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. https://openai.com/dall-e-2, 2022. 7

[74] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models, 2018. 10

[75] Christian Schlarmann, Naman Deep Singh, Francesco Croce, and Matthias Hein. Robust clip: Unsupervised adversarial fine-tuning of vision embeddings for robust large vision-language models. *arXiv preprint arXiv:2402.12336*, 2024. 3

[76] Zeyang Sha, Zheng Li, Ning Yu, and Yang Zhang. DE-FAKE: Detection and Attribution of Fake Images Generated by Text-to-Image Diffusion Models. In *ACM SIGSAC Conference on Computer and Communications Security*, page 3418–3432, 2023. 4

[77] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11): 612–613, 1979. 4

[78] Sergey Sinitsa and Ohad Fried. Deep image fingerprint: Accurate and low budget synthetic image detector. 2024. 4

[79] Lukas Struppek, Dominik Hintersdorf, Daniel Neider, and Kristian Kersting. Learning to break deep perceptual hashing: The use case neuralhash. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1795–1807, 2022. 3, 4

[80] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses, 2017. 10

[81] Vijay Veerabadran, Josh Goldman, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alexey Kurakin, Ian Goodfellow, Jonathon Shlens, Jascha Sohl-Dickstein, Michael C. Mozer, and Gamaleldin F. Elsayed. Subtle adversarial image manipulations influence both human and machine perception. *Nature Communications*, 14(1):4933, 2023. 4

[82] Jie Wang, Mingdeng Wang, Zhen Yang, Hao Wang, Yujun Li, Lei Zhang, and Gui-Song Qi. Stable signature: Structural watermarking for diffusion models. In *arXiv preprint arXiv:2312.12391*, 2023. 2

[83] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8695–8704, 2020. 1

[84] Zijie J. Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. Diffusiondb: A large-scale prompt gallery dataset for text-to-image generative models, 2023. 7

[85] Jiuqi Wei, Botao Peng, Xiaodong Lee, and Themis Palpanas. Det-lsh: A locality-sensitive hashing scheme with dynamic

encoding tree for approximate nearest neighbor search. *Proceedings of the VLDB Endowment*, 17(9):2241–2254, 2024. 3

[86] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7959–7971, 2022. 6

[87] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, 2018. 10

[88] Ning Yu, Larry Davis, and Mario Fritz. Attributing Fake Images to GANs: Learning and Analyzing GAN Fingerprints. In *ICCV*, pages 7555–7565, 2019. 4

[89] Christoph Zauner. Implementation and benchmarking of perceptual image hash functions. 2010. 3

[90] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023. 6, 8, 1

[91] Jian Zhang and Yuxin Peng. Ssdh: Semi-supervised deep hashing for large scale image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):212–225, 2019. 3

[92] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval, 2015. 3

[93] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. *European Conference on Computer Vision*, pages 682–697, 2018. 2

Provenance Detection for AI-Generated Images: Combining Perceptual Hashing, Homomorphic Encryption, and AI Detection Models

Supplementary Material

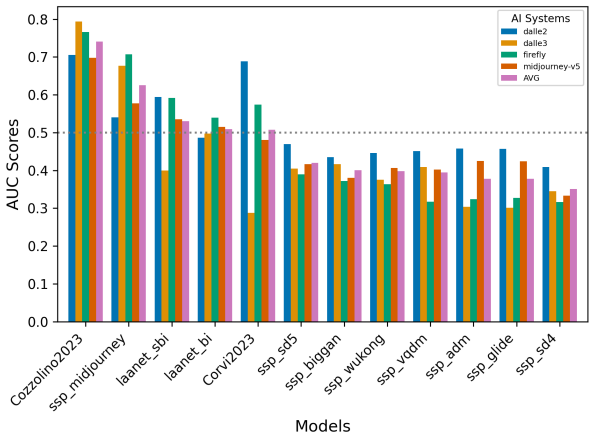

Figure 7. Area Under ROC curve for various models on postprocessed images. Arranged in descending order of average AUC.

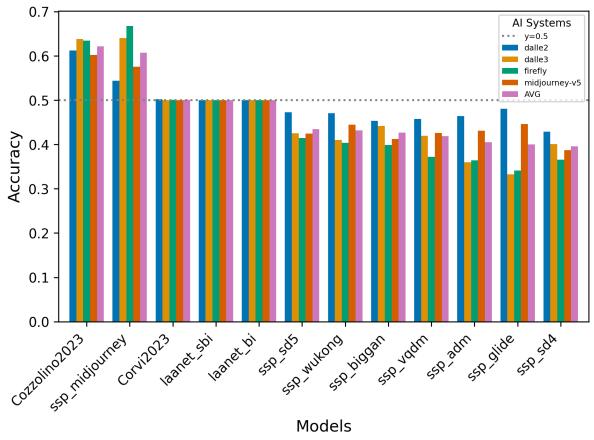

Figure 8. Accuracy for various models on postprocessed images. Arranged in descending order of average accuracy.

7. Benchmarking existing DL detectors

Existing litreature contains various methods for detecting AI generated images using machine learning models. These methods can be classified primarily into a) Artifact based methods and b) VLM based methods. Artifact based methods [17, 43, 61, 66, 83] leverage fingerprints of image generators or noise patterns of camera [19] to detect AI generated image. While this method is effective, it often does not generalize well to unseen generators and artifacts are often lost under operations like JPEG compression, resizing and random crop. VLM based methods [24, 67] leverage the join image-language feature space to detect fake images.

In this era of social media and freely available text-to-image generators the task of detection AI generated images has gained a lot of attention. A lot of different approaches are available but we found a lack of common benchmark. A lot of authors often do not report results on realistic scenarios like highly compressed images. Cozzolino et al. [24] benchmark state-of-the-art detectors and show through extensive experiments that VLM based method perform well on postprocessed images while performance of artifact based methods often fall to near random chance. Despite this effort there still existed new detectors [19, 66] that were not benchmarked under the same condition. We benchmark all detectors on the dataset and under transformations described in Section 4.3. The Area Under Curve and Accuracy for the models not benchmarked in [24] are visualized in Figure 7 and Figure 8 respectively. Our benchmarking reinforced the findings of Cozzolino et al. [24] and shows the superiority of VLM based detectors. Hence, we choose to leverage VLM based detector in our framework.

8. Ablation Study

Section 7 clearly motivates the use of VLMs for generalizable and robust detector. Choice of architecture and exact method of detection can significantly affect detection. We have to choose which VLM to use, size of backbone (B/L/H) to use and the method of detection. We can either use zero shot capablities of VLMs or train a classifier on top of features extracted from VLMs. If we choose zero shot methods we could either use fixed text template or generate custom text using state of the art models like BLIP. We explain each choice in detail below.

8.1. Choice of Backbone

Radford et al. [71] introduced contrastive learning of joint language image embedding space. The pioneering CLIP model was adopted and used extensively due to its generalization and few shot capabilities. Follow-up work [38, 90] improved this through various strategies, including additional loss terms, a new dataset created using novel data filtering strategies and changes in normalization. Open CLIP [47] has trained a range of these models on varying backbones, datasets and resolutions. We find that DFN [38] and SigLIP-based models [90] have the highest zero-shot imagenet accuracies. Our task involves using these models

VLM	Backbone	Resolution	Classifier	Text	DALL·E 2	DALL·E 3	Firefly	Midjourney v5	Avg
CLIP	ViT-L	224	Zero-shot	Fixed	0.524	0.558	0.550	0.534	0.542
CLIP	ViT-L	224	Zero-shot	BLIP	0.508	0.476	0.475	0.474	0.483
DFN	ViT-H	224	Zero-shot	BLIP	0.609	0.818	0.670	0.663	0.690
CLIP	ViT-L	224	SVM	-	0.680	0.74	0.646	0.627	0.673
DFN	ViT-H	224	SVM	-	**0.792**	**0.997**	**0.921**	**0.934**	**0.911**
SigLIP	so-ViT-400M	224	SVM	-	0.663	0.996	0.896	0.909	0.866

Table 4. Ablation study on AI detection model architecture with individual model performance. Table contains accuracy values on non postprocessed images.

for downstream tasks, so we use zero-shot ImageNet accuracies to indicate which model could perform well on our task. Thus, we experiment with DFN and SigLIP-based models along with Open AI's CLIP. We have three different backbones, i.e. ViT-L, ViT-H and so-ViT-400M. Results are detailes in Table 4. We observe that the DFN-based ViT-H backbone outperforms the CLIP-based ViT-L backbone and SigLIP-based so-ViT-400M backbone when using the same classification strategy, i.e. SVM. It also outperforms CLIP-based ViT-L when we do zero-shot classification. So, the DFN-based ViT-H backbone is the clear choice for our detection model. We note that the SigLIP-based so-ViT-400M backbone has performance comparable performance, so we include it in further experiments.

8.2. Method of Detection

Cozzolino et al. [24] tested Support Vector Machine, Logistic Regression (LR), Mahalanobis distance (MAH), Gaussian Naive Bayes classifier (GNB), Soft voting k-Nearest Neighbor (SNN) with Open AI's CLIP and found out that SVM performs the best on the task of detection AI generated images. However, CLIP's zero shot capablities were not tested. So we test three classification strategies, namely zero-shot classification with fixed text template, zero-shot classification with BLIP generated text template and SVM. Table 4 clearly shows SVM performs better, so we use SVM in our final detector.

9. Adversarial Training

In this section we discuss the adversarial fine-tuning implementation details for DinoHash and DL based detector.

9.1. DinoHash

Given a hashing model, H, we would like to finetune it such that an adversarial perturbed image, x', produces the same hash as the original image, x where $\|x - x'\|_\infty \leq \epsilon$, i.e. we would like that $H(x') \approx H(x)$. Hence, we initially formulated the following objective for finetuning:

$$H_{robust} = \arg\min_H \sum_{i=1}^{n} \max_{\|x'_i - x_i\|_\infty \leq \epsilon} \|H(x'_i) - H(x_i)\|_1. \tag{3}$$

The downside of this approach was that it led to a collapse of the hashing function and caused it to produce the same hash for all images, which is undesirable. This behaviour can be explained since the proposed loss function is perfectly minimized.

This led us to formulate our second optimization problem

$$H_{robust} = \arg\min_H \sum_{i=1}^{n} \max_{\|x'_i - x_i\|_\infty \leq \epsilon} \|H(x'_i) - H_{orig}(x_i)\|_1. \tag{4}$$

Where H_{orig} denotes the original, non-robust hashing model. This ensured that the model retains its original hashing capability. The inner maximization problem was solved by adapting APGD 25, a popular algorithm for adversarial attacks, to our use case.

Since the heaviside step function is non-differentiable, we used cross-entropy loss on the logits before the binarization step. Furthermore, we observed that adding a clean-loss term aided the optimization process and resulted in faster convergence. The final loss function used in code was:

$$\begin{aligned} L(x_i) = \text{CE}(\hat{H}(A(x_i)),\ \hat{H}_{orig}(x_i)) + \\ \alpha \cdot \text{CE}(\hat{H}(x_i),\ \hat{H}_{orig}(x_i)) \end{aligned} \tag{5}$$

Here $A(x)$ represents the attacked version of image x generated by APGD with $\epsilon = 8/255$, CE represents the cross-entropy loss with the second term as the target logit, $\hat{H}$ represents the deep hashing model without the binarization step, and α represents the loss weightage for clean examples and was set to 500 in our experiments. We fine-tune DinoHash on DiffusionDB using the loss function defined

‘	Clean Data		Adversarial		Adversarial	
ε	-		4		8	
	Non-Robust	Robust	Non-Robust	Robust	Non-Robust	Robust
DALL·E 2	0.559	0.54	0.645	0.838	0.662	0.822
DALL·E 3	0.972	0.847	0.66	0.867	0.685	0.864
Firefly	0.623	0.574	0.632	0.826	0.641	0.825
Midjourney v5	0.689	0.614	0.651	0.85	0.662	0.838
Average	0.711	0.644	0.647	0.845	0.663	0.837

Table 5. Comparison of Non-Robust and Robust Models Across Attack Types and Generators

above for $20,000$ steps with a batch size of 256 using the AdamW optimizer. The adversarial examples were generated using APGD with 10 steps.

We compare the adversarial robustness of DinoHash against different baseline models in Table 6. We note that for all baseline models, the adversarial attack achieves near-perfect accuracy, significantly underscoring the insecurity of their application as compared to DinoHash.

We would like to mention that we were initially experimenting with DISCO 46 to enhance robustness against adversarial attacks. However, our experiments indicated that DISCO is extremely vulnerable to standard PGD attacks, despite prior claims that the Backward Pass Differentiable Approximation Attack 10 was the strongest. This suggests that its reported robustness is heavily overstated.

ϵ	DinoHash	NeuralHash	Stable Signature
$\frac{4}{255}$	**62.0%**	0.1%	0.0%
$\frac{8}{255}$	**64.8%**	0.0%	0.0%

Table 6. Performance of different models under adversarial perturbations with the APGD attack performed using 100 steps

9.2. DL based detector

Schlarmann et al. [75] proposed an unsupervised method to adversarially finetune CLIP that preserves performance on downstream tasks. We use this method for adversarial finetuning of our model. We first discuss the method below. Then we discuss our results.

Method We aim to minimize the distance between the embeddings of unattacked images for original CLIP model and corresponding adversarially attacked image for the robust model. The idea is that if embeddings are unaffacted by attack then performance on downstream task will be preserved. In the following we denote with ϕ_{Org} the original CLIP encoder which is frozen. $\phi(z)$ is the encoder that is being trained. x is the input image and z is the corresponding adversarially attacked image. We propose the following embedding loss:

$$L_{\text{ours}}(\phi, x) = \max_{\|z-x\|_\infty \leq \epsilon} \|\phi(z) - \phi_{\text{Org}}(x)\|_2^2. \quad (6)$$

Training Details We use PGD attack for training with ϵ 4 and l_∞ norm. We use 10 iterations of attack per image with batch size of 128. We train for 20000 iterations on ImageNet dataset. We use ADAM optimizer with a learning rate of 1e-5, weight decay of 1e-4 and learning rate warmup of 1400 steps. We use L2 loss for attack. The training parameters are consistent with [75].

Results We do adversarial evaluation for just the finetuned version of our detector with DFN ViT-H backbone and 224 resolution. We evaluate using 100 iterations of APGD attack with binary cross entropy loss. We evaluate non-robust and adversarially finetuned model on clean data, attacked data with epsilon 4 and attacked data with epsilon 8. We present the results in Table 5. We observe that the adversarially trained model's performance does not degrade much on clean data but it performs much better on attacked data.

10. Securing User Data without FHE

Figure 9 presents an overview of our second proposal, comprising the following key steps:

① The *model provider* is responsible for fine-tuning and updating the generative model (e.g., OpenAI) with write-only access to the TEE. This ensures that the *model provider* cannot eavesdrop on user queries. Additionally, the *model provider* can update the perceptual hash function as needed.

②, ③ The user u_i submits a prompt to the server and receives the generated image as the result.

④ The TEE calculates the pHash of the resulting image Δ_i and distributes the shared

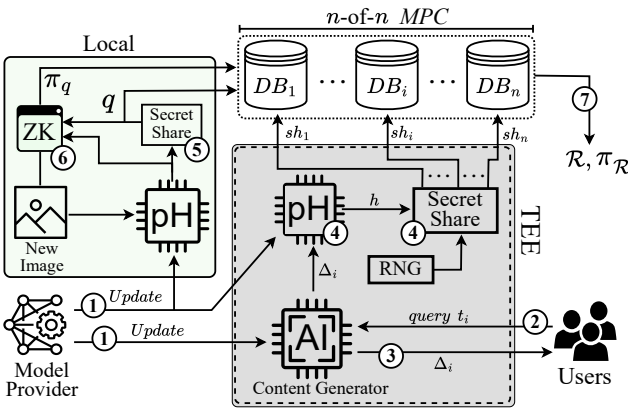

Figure 9. High level overview of the protocol.

pHash components $(sh_1, \ldots, sh_n)$ among n parties $(DB_1, \ldots, DB_n)$. During this phase, neither the generated image nor its related information is stored. No single party can retrieve any information about the image without collaboration from the others.

⑤ When a new image is checked against the shared database, its pHash is first calculated, and then this value is distributed among the n parties.

⑥ The proof of ownership for the original image is calculated and sent to the databases alongside the shared pHash value. Given the input size of the pHash model is 224×224, the owner of the query image x proves knowledge of a larger image X that has been resized to 224×224. The user also demonstrates that the resized image x has a pHash value of q, which is shared as $[q] = \{q_1, \ldots, q_n\}$. Moreover, since no party will ever gain access to all of these shares $(q_1, \ldots, q_n)$, it is necessary to prove to each party that they posses a unique and correct share of of $[q]$. To this end, we propose building a Merkle tree Δ out of these shares and prove each party is given a certain unique leaf of Δ. Finally, to connect these claims, commitments for both the original and resized versions of the image are needed, for which we employ collision-resistant image hashing techniques from prior work [28, 35]. More precisely, the user asserts the following statement:

$$S\big[\mathcal{R}_\Delta, \underline{[q]}, i, h_X, h_x\big] : \big\{\exists\, X, x \;\; s.t. \;\; x{=}f{\downarrow}(X) \text{ and } h_X = \mathcal{H}_\mathcal{C}(X) \text{ and } h_x = \mathcal{H}_\mathcal{C}(x) \text{ and } q = \mathcal{H}_\mathcal{P}(x) \text{ and } \exists\, \mathcal{O}(\mathcal{R}_\Delta, q_i, i)\big\} \quad (7)$$

We note that the prover generates n distinct proofs for $i \in \{1, \ldots, n\}$ to ensure each party has received a unique share of $[q]$.

⑦ The parties in the MPC-based database verify their received proofs and execute the query for the shared pHash value $[q]$. They then participate in the MPC to execute the given query on the shared DB. Based on the query result, the parties engage in a collaborative zkSNARK protocol to achieve public auditability, as follows:

– **Match:** If there is a match, the result of the execution includes the index of the matched values. Therefore, the parties claim the following statement:

$$S\big[\mathcal{R}_\Delta, [q], t\big] : \Big\{\exists\, id \in \mathbb{N} \;\; s.t. \;\; \gamma = \prod_0^n DB_i[id] \text{ and } \mathcal{R}_\Delta = \texttt{Merkle\_Root}(\cup\, q_i) \text{ and } \texttt{HD}(q, \gamma) {\leq} t\Big\} \quad (8)$$

– **No Match:** If there are no matches found in the database, then parties will claim following statement:

$$S\big[\mathcal{R}_\Delta, [q], t\big] : \Big\{\forall\, id \in \mathbb{N}_{\{DB\}} \;\Big|\; \gamma = \prod_0^n DB_i[id] \text{ and } \mathcal{R}_\Delta = \texttt{Merkle\_Root}(\cup\, q_i) \text{ and } \texttt{HD}(q, \gamma) {>} t\Big\} \quad (9)$$

It should be noted that while proving the statement in equation 8 is relatively feasible, the statement in equation 9 is computationally expensive. However, we argue that in most scenarios, the most critical supporting proof would be the case where a match is found, as expressed in equation 8.