Squint Hard Enough: Evaluating Perceptual Hashing with Machine Learning

Jonathan Prokos[1], Tushar M. Jois[1], Neil Fendley[2], Roei Schuster[3], Matthew Green[1], Eran Tromer[4], and Yinzhi Cao[1]

[1]Johns Hopkins University, jprokos4@gmail.com, {jois, mgreen, yzcao}@cs.jhu.edu
[2]Johns Hopkins University Applied Physics Laboratory, fendley@jhu.edu
[3]Vector Institute, roei@vectorinstitute.ai
[4]Tel Aviv University and Columbia University, et2555@columbia.edu

Abstract

Many online communications systems use perceptual hash matching systems to detect illicit files in user content. These systems employ specialized perceptual hash functions such as Microsoft's PhotoDNA or Facebook's PDQ to produce a compact digest of an image file that can be approximately compared to a database of known illicit-content digests. Recently, several proposals have suggested that hash-based matching systems be incorporated into *client-side* and end-to-end encrypted (E2EE) systems: in these designs, files that register as illicit content will be reported to the provider, while the remaining content will be sent confidentially. By using perceptual hashing to determine confidentiality guarantees, this new setting significantly changes the function of existing perceptual hashing – thus motivating the need to evaluate these functions from an adversarial perspective, using their perceptual capabilities against them. For example, an attacker may attempt to trigger a match on innocuous, but politically-charged, content in an attempt to stifle speech.

In this work we develop threat models for perceptual hashing algorithms in an adversarial setting, and present attacks against the two most widely deployed algorithms: PhotoDNA and PDQ. Our results show that it is possible to efficiently generate *targeted second-preimage attacks* in which an attacker creates a variant of some source image that matches some target digest. As a complement to this main result, we also further investigate the production of images that facilitate *detection avoidance attacks*, continuing a recent investigation of Jain et al. Our work shows that existing perceptual hash functions are likely insufficiently robust to survive attacks on this new setting.

1 Introduction

Many online service providers perform real-time content scanning to detect illicit content such as child sexual abuse material (CSAM), non-consensual pornography and terrorist recruitment videos [12,29]. The majority of these systems employ Perceptual Hash Matching (PHM) techniques. In this type of scanning, the provider evaluates each uploaded file using a perceptual hash function such as PhotoDNA [29] or PDQ [11] to produce a compact *digest*, which can be efficiently compared against a database of digests that correspond to known illicit content. This matching process is inexact by design. Nearly-identical media files can be detected either by comparing digests exactly, or by evaluating a similarity metric between digest values. Whereas a traditional cryptographic hash function is designed to produce a dissimilar output when fed a transformed media file (*e.g.,* one that has been compressed or re-encoded in a different format), perceptual hashing algorithms are designed to produce identical or similar digests when applied to files that *appear similar* to a human viewer. This makes content scanning systems robust to transformations such as resizing and re-encoding, and in some cases cropping or rotation.

(a) PhotoDNA Collision at Δ_d= 1720

(b) PDQ Collision at Δ_d= 86

(c) PhotoDNA Collision at Δ_d= 342

(d) PDQ Collision at Δ_d= 38

Figure 1: Targeted-second-preimage collision pairs for PhotoDNA and PDQ at high (top) and low (bottom) matching thresholds. The image on the left in a pair is our attack image, while the image on the right is the target. A lower matching threshold implies a better collision, potentially at the cost of image quality. Each pair was generated as described in Section 4.1.

Perceptual media hashing algorithms have been widely adopted for use in automated content scanning systems that examine image and video files uploaded to a server. Not only do they provide an efficient means for detecting illicit content, but the use of hashing eliminates the need for providers to store and exchange illicit content files. In typical deployments, positive matches will trigger a variety of different responses, including further human review, account closure, and reports to law enforcement.[1]

While there has been a great deal of research into the properties of perceptual hashing algorithms, surprisingly little work has considered these functions from a privacy or security perspective [10]. This reflects a common understanding that perceptual hash matching systems *are not expected to resist adversarial exploitation.* Indeed, this warning is given explicitly within the specification of the few widely-used perceptual hash algorithms [11] whose design goals are stated explicitly. This lack of adversarial robustness stems from the underlying hash algorithms' most useful feature, their robustness to small modifications. This feature is challenging to achieve while maintaining cryptographic security properties such as collision and preimage resistance.

Server-side content scanning systems have historically been deployed in ways that (implicitly and explicitly) minimize the impact of attacks on the underlying hash function. For example, providers maintain tight security around illicit content digest databases, which mitigates the risk of attacks that could *e.g.,* recover abusive content from digest data, or facilitate evasion of scanning systems.[2] Moreover, most server-side scanning systems have access to plaintext user content (from which digests are computed), and so preimage attacks on these digests are not relevant. The adversarial creation of collisions, including second-preimages,

[1] In the case of *Child Sexual Abuse Media (CSAM)*, US providers are legally required to manually examine suspected content, and then to submit reports to a "Cyber Tip Line" operated by the National Center for Missing and Exploited Children (NCMEC).

[2] The confidentiality of hash databases is frequently enforced by legal agreements required by content curators such as NCMEC. Moreover, cloud-based scanning systems such as Microsoft's PhotoDNA Cloud Service require providers to agree to terms that strictly limit adversarial access to algorithms, databases and even scan results [30].

poses more of a concern: such collisions might, for example, allow an attacker to generate an apparently-benign file that triggers a false match when uploaded by an unwitting victim. However, server-side scanning deployments blunt the impact of these attacks by requiring human verification of alleged matches: such verification does not affect user confidentiality, since the provider server already possesses the user's plaintext content. Despite these protections, it is notable that the design details of some widely-used hashing algorithms, most notably Microsoft's PhotoDNA, are kept tightly guarded and are available only under strict confidentiality agreements [24, 30].

E2EE content scanning: a new setting. A number of providers have recently begun to deploy large-scale implementations of *End-To-End Encryption (E2EE)* in both messaging and device backup settings. This trend poses a technical challenge for agencies that rely on leads generated from content scanning: in these systems the server may not have access to plaintext content, and hence traditional server-based content scanning systems will not operate correctly. In 2019, senior law enforcement officials in the US, UK and Australia published an open letter to Facebook, asking the company to delay its plans to deploy further end-to-end encrypted communication systems until this concern could be addressed [2]. This spurred providers and researchers to investigate the problem of deploying hash-based content scanning *within E2EE protocols*. Several protocols were subsequently published [1,25,38]. Although differing in their precise details, all share a common goal: to *selectively relax* the confidentiality guarantees of the encryption system through modifications to the client software so that *licit* content remains fully confidential, while files that match the illicit hash database can be detected by the provider (and possibly decrypted for further analysis.)

While in principle this approach appears straightforward, in practice it may have grave consequences for the confidentiality guarantees of end-to-end encryption, by effectively moving hash-based matching into the encryption mechanism itself. The security of the underlying E2EE communications system now *fundamentally depend on the properties of the underlying perceptual hash-matching algorithms*. Put succinctly: if confidentiality is determined by the output of a perceptual hash algorithm, then the security provided for even *licit* content is contingent on the properties of those algorithms. This change in setting motivates further evaluation of this technology.

Unfortunately, evaluating perceptual hash functions in this new setting is not straightforward, since these functions were not designed to resist adversarial attacks and have not been extensively evaluated for this purpose. In traditional PHM systems, accuracy is measured by measuring false positives when considering a natural image corpus, rather than an adversarial one. However as a component of an encryption systems, perceptual hash matching systems must be evaluated against *adversarial* actors. Hash-based matching systems could exhibit a dramatically elevated false-positive rate in circumstances where matches can be adversarially-induced (*e.g.*, by malicious service providers). For example, a useful attack might identify a pair of colliding images where one contains illicit content while the other is protected speech such as a political campaign image. This attack might cause a PHM system to trigger improperly on content that is clearly expressive speech, thus reducing the confidentiality properties of an E2EE system and exposing users to surveillance of licit speech. Attacks on a PHM system and its underlying hash functions thus hold the potential to create new surveillance vulnerabilities, as well as new forms of information leakage that can harm both users and abuse victims alike.

Our contributions. In this work we conduct an investigation into the properties of two perceptual hashing algorithms, considered from an adversarial perspective. Concretely, we consider the adversarial robustness of two widely-used perceptual image hash functions, including Microsoft's PhotoDNA [29] and Facebook's PDQ [11]. Both functions represent the current state-of-the-art in perceptual matching functions that have been deployed into production, and are today used to scan billions of user images each year.

In this work we focus primarily on the robustness of these functions, considering two attacker objectives. First, we consider targeted-second-preimage attacks, which generate a hash collision by subtly modifying an image such that its hash collides with a target hash. The attacker does not need to know the target hash's original preimage, and can thus be achieved only using the database of flagged image hashes. Second, we consider detection-evasion attacks, which create images that are semantically identical to known illicit content, but do not trigger a positive match. We accomplish this by devising a machine-learning optimization framework of hash inputs. Our framework is general and agnostic of the hash function used. One of our

findings is that different function designs have major impacts on the difficulty of crafting such attacks. Particularly, PhotoDNA and PDQ both pose a challenge compared to purely-neural approaches such as NeuralHash, because they are not end-to-end differentiable, rendering inapplicable standard adversarial-learning methods, which are guided by closed-form-computable gradients.

As mentioned above, our work is motivated by the recent development of end-to-end perceptual hash matching systems that have been devised for deployment [1] and in the research literature [25, 38]. To place our results in context, we also provide a taxonomy of attacks on these systems and demonstrate how the practical results in this paper may affect the deployed security of any systems that use them.

Concretely, our contributions are:

A formal taxonomy of attacks on E2EE-PHM systems. Finally, we formally define end-to-end encrypted perceptual hash matching (E2EE-PHM) systems and discuss a taxonomy of attacks that can be conducted against these systems. Notably, our taxonomy incorporates several potential designs that are being considered by industry, including (1) pure client-side matching systems, (2) encrypted private set intersection systems, and (3) edge hashing systems. We discuss how attacks on the underlying hash function can be used to undermine user confidentiality in each design.

A hash-function-agnostic framework for gradient optimization on perceptual-hash distances. We devise an image-perturbation optimization procedure that uses the hash as a black-box and finds a Monte-Carlo approximation of its gradients [8] in a given point, to minimize loss terms over hash distances and perturbation sizes. Specifically, to approximate function gradients of an input point, we sample small random perturbations around it and compute a per-pixel average of the measured effect on the loss, weighted by the amount the pixel was changed, across the samples. We show that, for appropriate parameterization which may be hash-function-dependent, this approximation can be used to minimize the loss terms over any perceptual-hash function we experimented with, despite its highly non-smooth surface.

Targeted-second-preimage and detection-avoidance attacks on Microsoft's PhotoDNA. We make use of a purported (and recently leaked) binary copy of Microsoft's PhotoDNA hash function [21, 22, 29] to evaluate the resilience of this function under these attack scenarios. PhotoDNA is a widely-used hash function that is currently used by Microsoft and several other providers, including Cloudflare and Dropbox. PhotoDNA is typically used in PHM systems to identify close matches using a similarity metric. We use our gradient-optimization approach to construct (1) semantically-different images that possess an arbitrarily-close similarity metric, and (2) perceptually-identical images whose hashes are above reasonable detection thresholds[3], demonstrating the viability of such attacks.

Targeted-second-preimage on Facebook's PDQ. PDQ [11] is a more recent hash function that was designed by Facebook and is used for internal hash-matching and similarity comparisons within that company's systems. Unlike PhotoDNA, the design of PDQ is public and a full specification can be found online. While the design of PDQ does not claim adversarial robustness, we show that it is in fact somewhat more robust to targeted-second-preimage attacks than PhotoDNA. Nonetheless, we are able to use our technique to devise meaningful targeted-second-preimages for PDQ-hashed images.

Concurrent work. During the course of our investigation, some concurrent projects have also considered of perceptual hash functions. In August 2021, Apple released a novel neural-network based function called NeuralHash; anonymous researchers quickly showed that it was possible to develop targeted collisions on images hashed in this system [1]. While the threat model considered in this effort is analogous to our targeted-second-preimage attacks, NeuralHash is essentially a standard convolutional neural network, and therefore by design amenable to gradient-based optimization over its input space. It is therefore not surprising that it is exposed to strong collision attacks that employ common adversarial-learning techniques. Attacking state-of-the-art PHMs used in practice like PhotoDNA and PDQ requires a more elaborate optimization framework.

[3] We discuss "reasonable" thresholds in Section 5.2.

Jain et al. [19] evaluated PHM robustness against detection-avoidance attacks, but did not consider second preimages nor looked at PhotoDNA. Finally, in a September 2021 blog post, Krawetz published a purported description of the PhotoDNA hash function [24], and claimed (without providing complete details) the ability to extract useful imagery from PhotoDNA hashes.

Ethical considerations. Our attacks potentially impact the privacy of end users who use services that employ PhotoDNA or PDQ as well as the security of such platforms. To this end, the vulnerabilities in the algorithms discussed in this work have been disclosed to the pertinent service providers (Microsoft for PhotoDNA and Facebook for PDQ).

2 Background

2.1 PHF-Based Content Scanning

Perceptual hash functions and hash-matching. A perceptual hash function (PHF) is a function $H : I \to V$ that accepts as input a media file x from a domain I and outputs a digest value D from a digest space V. An input x such that $H(x) = D$ is known as a "preimage" of D.[4]

Perceptual hash functions are used to build perceptual hash matching (PHM) systems, which make use of a second algorithm $\mathsf{comp} : V \times V \to \mathbb{Z}_+$ that accepts as input two digest values and outputs a measurement of their distance. This function implicitly defines the notion of *semantic equivalence* for two images. Two media files A, B that contain the same visual content (i.e., have the same semantic meaning), should register a small distance when their digests are compared. To compare digests, a perceptual hash matching (PHM) system will typically be parameterized by a threshold constant Δ_d: any pair of files A, B that satisfy $\mathsf{comp}(H(A), H(B)) \leq \Delta_d$ are considered to be a match.

PHFs are typically not designed to provide cryptographic security. These functions "summarize" media files into a shorter digest, and will frequently strip away many non-essential details while preserving features that represent semantic meaning. This raises the possibility of finding collisions (*i.e.,* nonidentical inputs that produce the same digest), as well as to extract useful information about the source file from its digest. Many PHFs also make it relatively easy to find *second preimages*, *i.e.,* given $H(A)$ find a file $B \neq A$ such that $H(A) = H(B)$. In inexact-match PHM systems, the condition for a hash match ($\mathsf{comp}(H(x), H(x')) \leq \Delta_d$) is even less strict, and we can therefore expect attacks that find inexact-match collisions to be even easier. Since we are primarily interested in PHM systems, we will elide the difference, and refer to any pair of inputs that satisfy the latter condition as a *collision*.

Since we seek to identify potential attacks against content scanning systems, arbitrary collisions and second preimages may not be meaningful (e.g., it is not problematic if flipping a bit preserves the digest). Instead, we are interested in attacks that produce *meaningful* collisions and second-preimages, as well as attacks that allow users to bypass content scanning systems. We discuss this in detail in Section 3.

E2EE content scanning. We seek to identify potential attacks against E2EE systems that incorporate PHM-based scanning: we refer to such schemes as *E2EE-PHM*. To understand what sort of attacks might affect the security of these systems, we will first discuss the proposals that have been put forward for how to combine E2EE with PHM:

Client-local matching. The simplest approach to incorporating PHM into E2EE systems is to take advantage of the fact that plaintext media files are exposed on both the sending or receiving endpoints. In these systems, clients are provisioned with a database of illicit content digests: when an endpoint attempts to upload (or download) a media file, these systems compute a digest and compare it to the local database. A downside of this approach is that illicit content digests are available on the client endpoint, making them vulnerable to extraction by reverse-engineers. To date we know of no *media* scanning platforms that employ this paradigm, but this technique has been used to implement keyword-based scanning in some Xiaomi phones [37].

[4] Note that $x \in V$ may be an "image" file (i.e., pixel data) but that term is unrelated to "preimage" of the function H.

Edge hashing. An alternative approach, exemplified by Microsoft's "Edge Hash" [28], can also be used to hash user content at the client endpoint. These systems implement an API for computing either full or partial digests at the client, and transmitting the results to a centralized provider for processing and comparison. This approach avoids the need to store illicit content digests at the endpoint, at the cost of transmitting digests of *all* user content to the provider.

Cryptographic matching. To address the privacy deficiencies of the above proposals, recent work has devised privacy-preserving cryptographic protocols to implement hash matching [3, 13, 25, 38]. A key goal in these systems is to preserve the privacy of *both* the user's content and the illicit digest database itself. As in the above proposals, user content digests are computed at the endpoint. However, the resulting digest is compared with the provider's database using a 2-party computation (2PC) protocol, which can be based on homomorphic encryption [38] or on private set intersection [1, 25]. A variant of this approach was recently prototyped by Apple [1] and is scheduled to be deployed in Apple's iCloud Photos system in the near future.

What happens in the event of a match? Each paradigm described above dictates how a system will detect content matches, but does not specify what the system will do when a match is found. In essence this is a deployment decision that individual providers will make.

In the case of illicit content such as CSAM, current server-side content scanning systems flag matching files to the provider for manual examination and reporting. For obvious reasons, these systems do not request the user's permission to perform this notification, and users are not alerted to the fact that their content has been flagged for examination. A similar approach has been realized in client-side PHM systems as well. For example, Apple's recent *client-side* CSAM scanning proposal emulates this feature, and makes dataset privacy an explicit security goal: even a malicious client must not learn any additional information about the server's dataset, including the fact that their content has been matched [3, §4.4]. Once a sufficient number of matches has been locally identified for the same user, Apple's system delivers a decrypted low-resolution copy of the transmitted content to the provider for review.

Hypothetical E2EE-PHM systems may choose to operate differently, *e.g.,* by simply blocking illicit content rather than alerting the provider. However, in this work we apply the precautionary principle: we assume that some providers will deploy E2EE-PHM systems that enable investigatory capabilities that match current (non-E2EE) content scanning systems. In the worst case this entails E2EE-PHM client automatically uploading *the plaintext contents of any matching file* to the provider immediately upon identifying a match.

2.2 Case Studies: PhotoDNA and PDQ

We now describe the PhotoDNA and PDQ PHM schemes.

Microsoft PhotoDNA. PhotoDNA [29] was invented in 2009 by Microsoft Research and Dartmouth College. The precise design of the algorithm was never published, and IP ownership was subsequently donated to NCMEC [29] who have maintained the secrecy of the technology by licensing it to technology providers under non-disclosure agreement. While the precise operation of PhotoDNA is not published, several sources have provided algorithm descriptions of varying completeness: these include partial descriptions published by Microsoft [29], as well as more complete purported descriptions derived from reverse-engineering and/or partial descriptions published by researchers [23, 24]. In 2020 a binary implementation of PhotoDNA was extracted from forensic software [21, 22], and we use this algorithm in its binary form for our experiments in Section 5.

Although our attacks use the PhotoDNA binary in a black-box manner, we now briefly summarize the description drawn from public documents, as summarized by Krawetz [24]. The algorithm works by first reducing the image size to 26 pixels per side, and then converting it into grayscale. The resulting is then split into a 6×6 grid of separate "bins". These bins are laid out in an overlapping pattern to ensure that minor trimming does not severely affect the hash, resulting in 36 bins across the entire image. Each grid is then passed through the Sobel operator [39], a type of 1-dimensional convolutional filter based on the

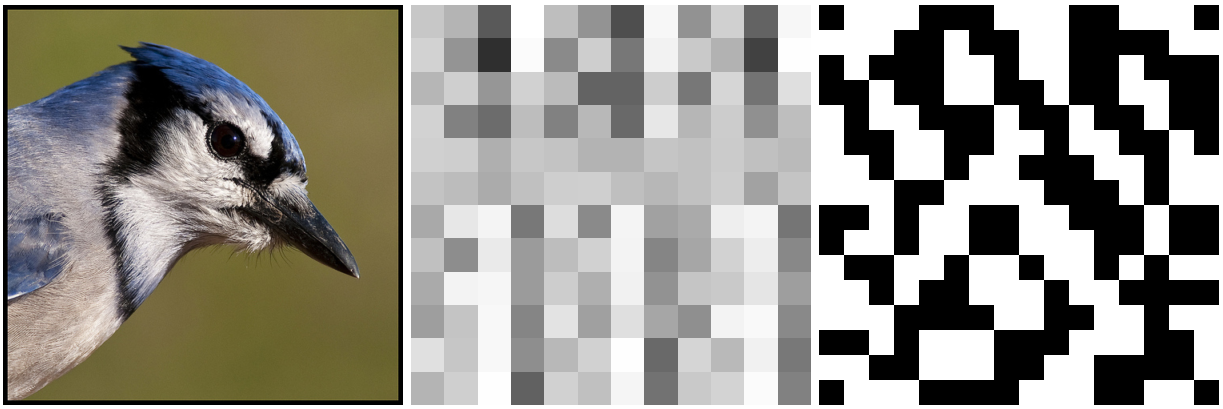


Figure 2: An image from ImageNet [36] (left) and a visualization of its associated PhotoDNA (center) and PDQ (right) hashes.

difference between pixel values. This has the effect of intensifying (and therefore delineating) the edges of an image. The hash for a grid is a 4-tuple of the sum of these Sobel outputs leftwards, rightwards, upwards, and downwards across the grid. The final hash consists of the 36 4-tuples for each of the grids, for a total of 144 values.

The output digest of PhotoDNA is a vector of 144 numeric values that can be compared to another digest using a simple distance metric. The exact parameters of the PHM system as deployed by online service providers, namely the distance metric comp and the chosen distance bound Δ_d (Section 2.1), are not public. Our experiments utilize an L_1 distance metric (as used in [13]); to evaluate the appropriate bound Δ_d we conduct experiments with false-positive rates in §5.2.

Facebook PDQ. The PDQ algorithm was published in 2019 by Facebook, in response to the company's need for an updated and more flexible perceptual hash algorithm. Unlike PhotoDNA, the implementation and design goals of PDQ are public [11].

PDQ begins with a normalization step, which processes the luminance data from the input image, followed by a two-pass Jarosz filter to downsample the image to 64×64. Within the downsampled image, compute a sum of absolute values of horizontal and vertical gradients along with a two-dimensional discrete cosine transform, resulting in a 16×16 DCT. For each of the 16x16 bits of the output hash, emit a 1 if the corresponding element of transform is greater than the median, otherwise emit a 0; the resulting 256 bits form the output digest.

Similar to PhotoDNA, we compare the hash output of PDQ to another digest using an L_1 distance metric (as used in [11]). The designers of PDQ provide a metric of choice: since the output is a 256-bit binary string, they recommend that the two hashes should be compared utilizing the Hamming distance metric. Our experiments use this to implement the function comp.

Discussion. PhotoDNA and PDQ both specify *inexact* matching algorithms. Notably, both employ Sobel gradients in their analysis of an image. This filter, applied horizontally and vertically, prominently detects the edges of an image, as these edges have high gradient disruption along them. This is useful in perceptually identifying an image, as pixel-level changes, such as compression or minor cropping, will not fundamentally change an image's edges, and therefore should have a minimal effect on the digest.

In both algorithms, the hash value is calculated as a sum of the value of gradients. This can be equivalently understood as a count function, and is therefore *non-differentiable*. While this can be estimated using tanh, our attacks below do not require differentiablity, as we abstract this portion of the hash as a black box.

3 Taxonomy of Attacks on E2EE-PHM

The aforementioned proposals for E2EE-PHM content scanning systems raise new confidentiality vulnerabilities that can affect user security, the effectiveness of scanning, and perhaps even impact the privacy of abuse victims. More critically, the nature of these vulnerabilities cannot be evaluated without a clear understanding of the properties in the underlying PHM. We proceed to provide a taxonomy of attacks on E2EE-PHM systems, their threat model, and the corresponding attacks on the underlying PHM scheme.

3.1 Targeted-Collision Surveillance Attacks

Threat model. All E2EE-PHM systems are potentially vulnerable to attacks in which a surveillance adversary constructs meaningful files with semantically *non-equivalent* content, yet that will identify as a "match" to the underlying PHM system. For example: an attacker could identify two files A, B such that A comprises illicit content stored within the provider's database, while file B contains licit content that the attacker wishes to surveil. Given such a collision, two form of surveillance attacks are possible.

First, the attacker may induce targeted users to transmit file B, e.g., by broadcasting it on a bulletin board they frequent, or by sending it to them on some anonymous messaging system. Any user who takes the bait and sends file B via the E2EE-PHM system, will be (mis)detected as having sent the illicit file A. If the detector can be notified of these false positives, then they can deduce who transmitted file B and when.

Such notification may happen in several ways: the attacker may be a nefarious E2EE-PHM service provider, or may have acquired access to the service provider's system via an intrusion or an insider. The attacker may even partially control an organization (such as NCMEC) which receives notifications.[5]

A second attack variant, using such a collision, is to use the illicit file A as a means of including the digest in the E2EE-PHM database (e.g., by presenting it directly to the service provider, or to an organization such as NCMEC that is trusted to curate the database). This will cause any transmission of file B to be flagged, causing a notification which may be intercepted as above.

When a user transmits file B through a private channel, the transmission would trigger the PHM system and reveal this transmission to the service provider. Such attacks can be realized by a malicious (or corrupted) service provider as a means to enable surveillance of users with specific interests. However, they can also be implemented by an unrelated third party using second-preimage attacks on known content digests.

The importance of targeted-collision surveillance attacks. The debate around E2EE-PHM has been vigorous. Some have argued that targeted collision attacks are not a concern, since providers can manually review such false-positives before they are reported to law enforcement. However, this argument is inconsistent with the stated goal of end-to-end encryption systems, which are *explicitly designed to prevent* the leakage of plaintext data to the provider. A number of previous works have proposed or addressed attacks on E2EE systems that are operable only by attackers who compromise the provider's system [14, 26, 35], and many of the attacks have resulted in protocol repairs by providers; hence this threat model is not unusual. Moreover, even if match results never leave the provider's systems, they can still place users at risk: in multiple instances, state-sponsored attackers have infiltrated major US tech firms with the goal of conducting surveillance against platform users [9, 32].[6]

Generating perceptual second-preimages. Targeted-collision surveillance attacks require suitable collisions to be generated. If the PHM's digests are statistically concentrated around a small space (which is a vulnerability), then such collisions may be found by chance just by trying a large corpus. Finding regular preimages is typically trivial for PHM, where tiny local changes in an image may not perturb its digest, but a collision of perceptually-similar images is not useful for the surveillance attacks. A second-preimage attack may not be useful either, if the generated preimages are "garbage" images that would not pass muster with

[5] Notifications may be monitored by humans, as in Apple's system, but this is unlikely to be foolproof and moreover creates additional attack vectors.

[6] Most notably, in 2019 several Twitter customer service employees were prosecuted for allegedly spying on critics of the Kingdom of Saudi Arabia [9]. Content moderation employees and and their infrastructure, including operations outsourced to third-party firms, represent an ideal access point for sophisticated attackers.

target users (in the first attack variant) or the digests' curators (in the second variant). What is desired is finding a collision between images that are perceptually different and also have the aforementioned requisite semantics, which leads us to the following form of attack on PHM.

Targeted collision attacks. In this special form of second-preimage attack, the attack algorithm takes as input a known digest D corresponding to the hash of an *unknown* target image T, as well as a starting image S. The goal is to identify a new image T' that is perceptually similar to the starting image S but whose digest is close to that of the target image T: $\mathsf{comp}(H(T'), D) < \Delta_d$.

If the adversary can achieve this at a practical cost, then they can conduct the second attack variant by setting T to the specific image they wish to surveil, and the source image S to an illicit image of their choice. They can also conduct the first attack variant by setting the source image to any image they believe is likely to be circulated by their targeted users, and the target image to any illicit image of their choice; or vice versa.

3.2 Framing and Censorship

Threat model. A variant of the previous attack scenario considers an adversary wishes to frame an innocuous user for trafficking in illicit content. We assume that the adversary knows some target image T the user will send using the E2EE-PHM system (e.g., the attacker may send an apt meme image to the user). We also assume that the adversary can cause an image of their choice to be added to the database of illicit image digests (e.g., by submitting it to the database's curator, or planting somewhere it would be found and sent to the curator).

In this case, the adversary can use the a targeted collision attack as above, setting the source image S to some arbitrary illicit image. The resulting image T' is perceptually similar to S and thus will be considered illicit when submitted; but its digest is close to the target image T, causing the user to be flagged.

Similarly, an adversary may abuse the PHM-based content scanning to prevent legitimate content from being freely communicated. Assume the adversary knows an image they want to censor (or even just the digest D of that image), and as above, they can cause illicit-looking images to be added to the database. Then using a targeted collision attack, the adversary can generate and submit an illicit-looking image T' whose digest is close to that of T, thereby causing any sending of T to be flagged.

3.3 Detection Avoidance

Threat model. Access to illicit content digests, as in client-local matching, also creates a risk that dishonest users will abuse this knowledge to evade detection.

The simplest, and inherent, attack is where an attacker wishes to send an illicit image, but is willing to abort if they would be flagged. Given DB, a perceptual hash database of illicit content, the attacker can simply compute the digest of their illicit image and check it against DB before sending it. Notably, this attack is done locally on the attacker's device, and does not leave any externally-visible trace.

A more nefarious attack is to create a different image that is perceptually similar but avoids detection, as follows.

Perceptual detection avoidance. In this attack on PHM, the attacker has a starting image S. Their goal is to generate an *avoidant* image S' that is perceptually similar to S but whose digest is far from any in DB: $\mathsf{comp}(H(S'), D) > \Delta_d$ for all $D \in DB$.

If this can be done at a practical cost, then the attacker can effectively send arbitrary images without ever being flagged, and moreover further dissemination of those images by others will not be flagged either (until a different DB or detection system are encountered).

3.4 User Data Leakage

Threat model. In the edge hashing approach to E2EE content scanning, the provider gets the PHM digest of every transmitted user image. Even if the digest does not match the provider's database, it may

reveal information about the content. Depending on the properties of the hash function, this approach could potentially leak a substantial amount of information about a user's communication pattern.

The following attacks on the PHM scheme would facilitate such leakage from edge-hashing E2EE-PHM.

Preimage attribute recovery. In this attack model, the adversary receives a digest $D = H(S)$ and tries to recover attributes of the unknown input image, S. This attacker is not expected to recover the entirety of the original preimage, but is able to recover the attributes that give the image its meaning. For example, the attribute can be an image label from a classification dataset, the fundamental attribute distinction being a binary classification task on a set of images.

Consider an adversary who has access to an attribute function attr and the digest D. attr corresponds to some aspect of the image, such as identifying objects, locations, or number of people. The adversary attempts to recover the value of $\mathsf{attr}(S)$ using only $D = H(S)$. This implies that the hash $H(S)$ leaks some information about the image such that the attacker can recover $\mathsf{attr}(S)$. In the context of E2EE-PHM, this could reveal (for example) the topic of an encrypted conversation between users from the categories of images they exchange.

Preimage reconstruction. An even stronger attack is to recover the original image S used to compute the digest $D = H(S)$. They could then inspect it manually, or provide it as input to any attr function of their choice and thereby completely violate user's privacy.

This setting is somewhat analogous to the notion of preimage attack resistance in cryptographic hash functions, but with several differences. First, we cannot hope to perfectly reproduce the original image S from a digest that is much smaller; instead, we seek to generate an image S' that is perceptually and semantically similar to S, but expect it to be subtly different at the fine-grained pixel level where most entropy lies. Similarly, we would not be satisfied by just any image S' that has a digest D (as would be the case for cryptographic hash function applications such as password hashing [4]); rather, we wish to generate a meaningful, "realistic" image, given some prior assumptions about the structure of images sent by users.

3.5 Illicit-Content Data Leaks

Threat model. E2EE-PHM using client-local matching exposes the database of illicit content digests at each client endpoint, which increases the risk that these digests will be extracted and made public. In this model, we consider an attacker who has access to DB, a perceptual hash database of illicit content as above. "Access" is defined loosely here, as the attacker may have a local copy of a publicly-available dataset, may extract such a dataset from a device or an app, or may download it from a dataset via a cloud service. Indeed, client-local matching *necessitates* such access to be given by the service provider or software/hardware vendor.

This poses a risk that attackers might detect attributes of this highly-sensitive content, or even reconstruct it. In a worst-case outcome, this could identify abuse victims and expose them to further harm.

Attacks in this model are essentially the mirror image of those in Section 3.4. The attacker here is a user who gained possession of DB, and attempts to recover information from DB to reveal illicit content: whether by reconstructing an entire preimage of a digest in DB, or by attempting to detect an attribute of an input image. Even simple attributes could be devastating to the privacy of the victims depicted, indirectly, through the digest values. The PHM-level attacks are thus the same as in Section 3.4, applied to digests in DB instead of user-supplied images.

4 Attacking Perceptual Hash Functions

In the remainder of this work, we focus on the attacks described in Section 3.1 through Section 3.3. This includes attacks that identify targeted collisions, as well as detection avoidance attacks against perceptual hash functions. Our attacks in this paper do not consider data leakage or pre-image reconstruction attacks, though we include them in our taxonomy for completeness. We now discuss the methodology for performing these attacks against PhotoDNA and PDQ.

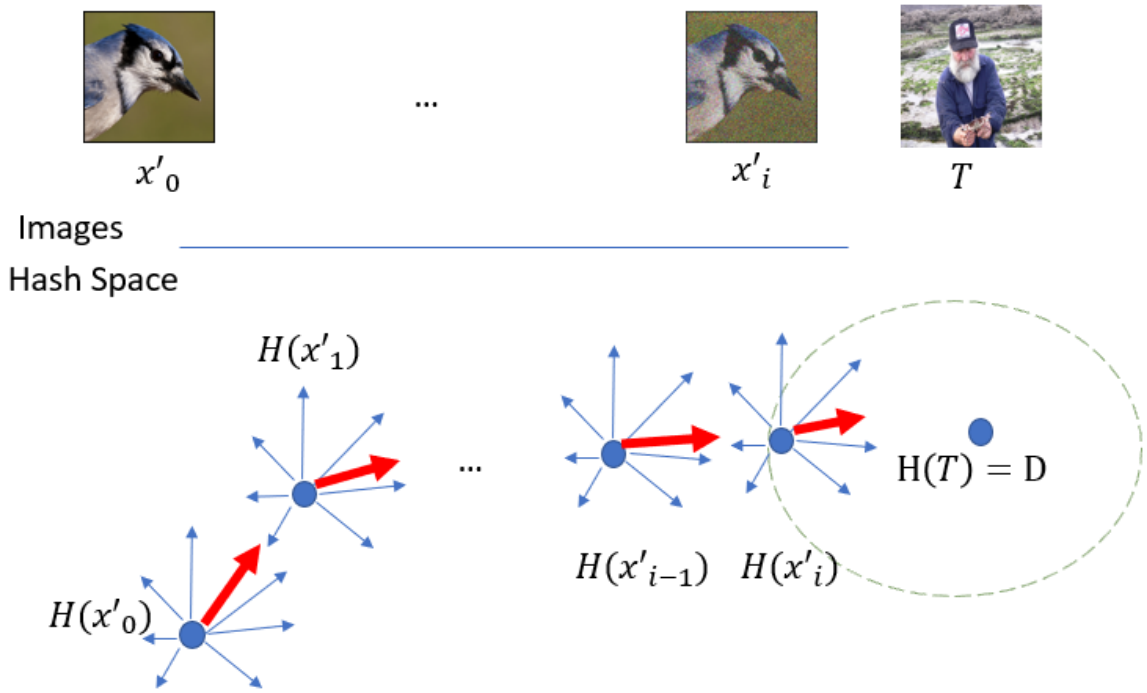


Figure 3: A visualization of the targeted-second-preimage attack and gradient estimation in image and hash space. For each iteration of $H(x'_i)$ the blue (thin) arrows represent query image changes to the hash and the red (bold) arrow represents the calculated gradient at $H(x'_i)$. Note that hash space H is a non-continuous space, so each red arrow is a local approximation of the gradient and may not point directly at the target hash $H(T)$.

4.1 Targeted-Second-Preimage Attack

To find collisions in the aforementioned perceptual hash functions, we developed a novel method for constructing meaningful collisions by identifying targeted second preimages of known digests. Our attack algorithm takes as input a known digest D corresponding to the hash of an *unknown* target image T, as well as a starting image S. We consider T as unknown because in real world deployment we will only have access to the hash. We generate a new image T' that is semantically similar to the starting image S but such that $\mathsf{comp}(H(T'), D) < \Delta_d$. We do so using indirect Monte-Carlo approximations of the hash-function gradients to minimize loss terms over hash distances and perturbation sizes. A graphical representation of our process is found in Fig. 3.

Our attacker's objective is to find a minimal δ such that:

$$\mathsf{comp}(H(S+\delta), H(T)) < \Delta_d \tag{1}$$

Gradient-based optimization. The attack begins by setting $x'_0 \leftarrow S$, and calculating its hash's distance to the target hash $\Delta_{d'_0} \leftarrow \mathsf{comp}(H(x'_0), H(T))$. Our goal now is to find a vector of small changes to each pixel δ_0 so as to minimize $\mathsf{comp}(H(x'_0 + \delta_0), H(T))$. We then set $x'_1 \leftarrow x'_0 + \delta_0$ and repeat this process for $x_i, i >= 1$ until $\mathsf{comp}(H(x'_i), H(T)) < \Delta_d$, i.e. x'_i and T will be hash matches. For the final index m, our optimization outputs $\delta \leftarrow x'_m - T$.

For differentiable functions, the natural choice for δ_i is the gradient vector with opposite sign $-\frac{\partial H}{\partial x'_i}$, normalized and multiplied by a "learning rate" γ that modulates the change size.

Estimating gradients. Since hashes are not necessarily differentiable, we use a gradient estimate via Monte Carlo simulation on each candidate image x'_i. At the i^{th} step we begin with a candidate image x'_i and calculate its hash $H(x'_i)$. We then calculate Δ'_i, the distance to the target hash, $H(x)$. We then generate q perturbations $p_1, \ldots, p_q$. For each perturbation j we calculate the change in our objective $c_j \leftarrow \mathsf{comp}(H(x'_i + p_j), H(T)) - \mathsf{comp}(H(x'_i), H(T))$. These terms are then used as the coefficients to calculate a weighted average of all the perturbations to serve as the gradient.

$$\delta'_i \leftarrow \frac{1}{q} \sum_{1}^{q} c_j \cdot p_j \tag{2}$$

We normalize and multiply by the learning rate to produce δ_i, i.e., $\delta_i \leftarrow \frac{\delta'_i}{|\delta'_i|_2} \cdot \gamma$.

A higher number of of samples q corresponds to increased approximation accuracy. We vary it across iterations, to improve accuracy as we approach the objective. In the i-th iteration, $q = \min(1000, 100\sqrt{i+1})$.

Differences from HopSkipJumpAttack. Our optimization framework was inspired by Chen et al.'s HopSkipJumpAttack [8] that estimates gradients in a black-box fashion using a binary decision boundary. Our attacks benefit from the ability to use hash distances rather than model decisions. Hash distances are typically more fine-grained and change across the input space (not just near a classifier's decision boundary), thus providing our Monte-Carlo approximation with a strong signal of pixel-value-to-output influence trends.

4.2 Detection-Avoidance Attack

In this attack we attempt to find the smallest perturbation that will cause an image to no longer match the original non-perturbed image. Given a target image T, this amounts to finding a *semantically equivalent* image S where $\mathsf{comp}(H(S), H(T)) > \Delta_d$, meaning the two images do not trigger a hash match at the given threshold. Here, we conservatively assume that the attacker does not have access to the hash at all, but can check if any number of images are considered a hash match with the original image for a given Δ_d. This attack is thus black-box with respect to the details of the hash distance comparison, instead relying on a decision between a match or not. To this end, we adopt the full HopSkipJumpAttack adversarial framework to hash collision avoidance. We hereby shortly describe this attack.

For a given target image, we begin with a random second image S'_0 which is not a hash collision as a starting point. We then repeatedly update S'_i to be as close as possible to T while still not being a hash collision. For each step, we begin with a non-colliding image S'_i and the target image T. We define the vector v between T and S'_i, and move from T along v until we move a distance the smallest distance l such that $b = T + l * v$ is not a collision (the "decision boundary" in machine learning classification). Since T is guaranteed to be a hash collision with itself, and S'_i is not by definition, there is a point on v which we call l at which the hash distance switches from colliding to a non collision. After the boundary point b is found, the change vector δ_i is computed at b by estimating the gradient as before, and a step of size γ from the boundary is taken in the direction of the gradient. This new point, $S'_{i+1} = b + \delta_i$ will serve as the starting point of the next step. The gradient is calculated the same as the collision threat model, except each coefficient is either -1 or 1, corresponding to which side of the decision boundary the query point falls on on. This process is repeated for a set number of steps or until convergence.

5 Evaluation

5.1 Experimental Setup

As discussed in Section 4, Microsoft has not publicly disclosed the PhotoDNA algorithm and has not provided any open implementation. However, a putative implementation of PhotoDNA was leaked on GitHub in 2021 [21,22], and we use it for our attacks. The implementation appears legitimate; our analysis in Section 5.2 shows that it has behavior characteristic of perceptual hashes, and others have shown that it acts similarly [21,

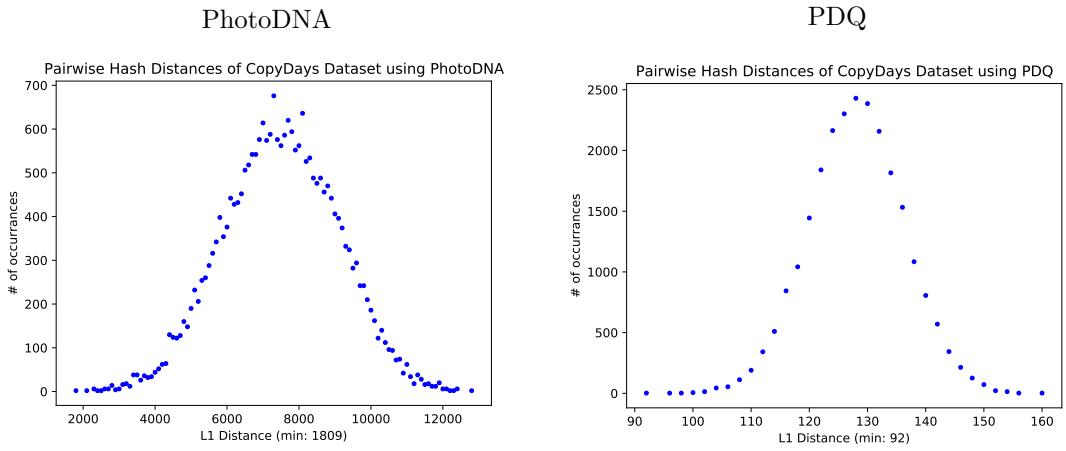


Figure 4: Frequency of pairwise hash distances derived from 157 perceptually distinct images' pairs in from CopyDays dataset. PhotoDNA shown in bins of 100. Used to determine threshold for a hash match.

24] to a high-level description of PhotoDNA by one of the algorithm's authors [13]. If Microsoft were to officially release PhotoDNA, we would be easily able to modify our attack against their implementation. For our implementation of PDQ, we use the official source implementation published by Facebook [11].

All of the experiments in this section were compiled and run using PyTorch 1.9 and Python 3.6 on two separate machines. PhotoDNA attacks were run using a 6-core AMD Ryzen 5 5600X CPU and Nvidia RTX 3070 GPU, while attacks on PDQ were run using a 72-core Intel Xeon CPU E5-2695 v4 without a dedicated GPU. We use a GPU-accelerated machine for PhotoDNA as we found that the bottleneck was machine learning operations, not the dynamic-link library of PhotoDNA used to compute hashing. On the other hand, we found that PDQ was far more CPU intensive due to hash computations being run in Python, instead of a compiled library.

5.2 Baseline Experiments

Our analysis consists of two stages. First, to establish a baseline for hashing accuracy and distance thresholds in *non-adversarial* settings, we evaluated the hash algorithms on unmodified image files drawn from a standard dataset, and then consider the same images with light (but non-adversarial) transforms.

Each of PhotoDNA and PDQ uses inexact matching and a distance metric to compare digest. The threshold Δ_d is adjustable, and can vary by deployment. The specific value used in a system has a complex impact for our purposes: higher thresholds are likely to make a PHM system more sensitive to re-encoded content, with the possibility of a higher natural false-positive rate. At the same time, lower thresholds can potentially make collisions more difficult to find, while improving the success of avoidance attacks.

Determining match thresholds. We conduct our attacks at a variety of collision thresholds for each hash function. To provide some context for these thresholds, we conducted a series of experiments as described by Facebook in its evaluation of PDQ [11]. These involve computing the pairwise distances of a set of the hashed CopyDays dataset, which consists of 157 perceptually-distinct images [18, 20]. The resulting 24,492 pairwise results are plotted in Fig. 4 for both PhotoDNA and PDQ. We find that a matching distance of approximately 92 for PDQ, and 1,809 are an appropriate threshold to eliminate false positives in this representative dataset. For simplicity, in our later experiments we will use 90 as our baseline matching threshold for PDQ and 1,800 for PhotoDNA.

Accuracy under simple transforms. To evaluate the matching efficacy of these algorithms under simple image transforms, we compared the unmauled CopyDays images with transformed images from that dataset.

	Precision	Recall
Original	1.00	1.00
JPEG/5,8	1.00	0.43
JPEG/50,75	1.00	0.34
Crops/10,15	1.00	0.71
Crops/70,80	1.00	0.43
Strong	1.00	0.51

(a) PhotoDNA matching using $\Delta_d = 1800$

	Precision	Recall
Original	1.00	1.00
JPEG/5,8	1.00	0.54
JPEG/50,75	1.00	0.34
Crops/10,15	1.00	1.00
Crops/70,80	1.00	0.99
Strong	1.00	0.53

(b) PDQ matching using $\Delta_d = 90$

Table 1: Precision and Recall values for matching tasks utilizing various transformed datasets. For JPEG compression we create two datasets, one containing JPEG-compressed images with 5% and 8% degradation, and a second containing JPEG-compressed images 50% and 75% degradation. For crops, we similarly create two datasets, one with crops missing 10% and 15% of the image, and another dataset removing 70% or 80%. These four datasets are of size 471. Strong is adversarially perturbed, size 314, as defined by [18].

These include a first set of JPEG compressed images with 5% and 8% degradation respectively, and a second set containing JPEG compressed images 50% and 75% degradation. For crops, we similarly compare two datasets, one with crops missing 10% and 15% of the image, and another dataset removing 70% and 80%. These four datasets are each of size $471 = 3 \times 157$. Finally, we compared against a "Strong" that is adversarially perturbed, size $314 = 2 \times 157$. By computing the pairwise distances of both the original and transformed versions of the dataset, we can see how those transforms affect the resulting hash distances under each algorithm. These results are shown in Table 1.

Using the baseline thresholds, we observe that PDQ performs slightly better at matching minor JPEG compression, but much better than PhotoDNA when matching both minor and major cropping. This behavior is likely due to the underlying primitives used to perform the perceptual hashing; PDQ applies a modified version of the Discrete Cosine Transform (DCT), the same as the transform in JPEG compression [42], while PhotoDNA relies primarily on its binning algorithm. Interestingly, PhotoDNA performed relatively poorly on crops, even though it takes specific steps to mitigate changes in hashing due to crops [24].

5.3 Attack Results

We implement the following two attacks utilizing images from the ImageNet Validation dataset [36][7].

5.3.1 Targeted-Second-Preimage Generation

Given a target image we attempt to generate an image which collides in hash space, but remain perceptually distant (kayak to toilet bowl in Fig. 1a, boy to shark in Fig. 1b, etc.) utilizing the attack described in Section 4. This corresponds to being able to cause false positive illicit content hash matches in a deployed system using benign images and the database of illicit-content hashes.

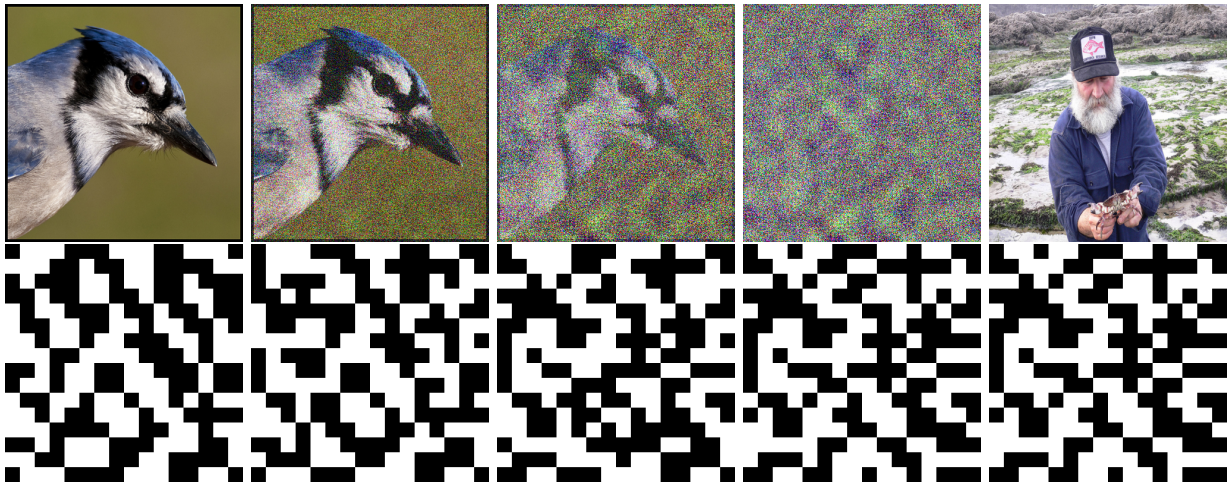
Results. We were able to find collisions at various levels below the baseline Δ_d discovered in Section 5.1 for complete attacks on PhotoDNA and PDQ. We successfully created targeted preimages on randomly-chosen ImageNet sample image pairs, using our minimal threshold parameter $\Delta_d = 1,800$ for PhotoDNA and $\Delta_d = 90$ for PDQ. Figures 5 and 6 show two such example images, with total hash distances $1,800$ and 88 respectively. We ran each of our algorithms until Δ_d approached zero – a perfect, exact collision on the hash function – but with visible effects on image quality.

Attack progression. We performed additional experimentation to characterize the relationship between the number of iterations our attack executes and the resulting hash distance. Figure 7 show the progress of our attack on 30 randomly selected ImageNet image pairs. For PhotoDNA, 17 image pairs reached the

[7]This is a common image dataset, and has been used in prior perceptual hashing work [19,31].

(a) Start: 6162 (b) Step 8200: 1797 (c) Step 12000: 963 (d) Step 20000: 342 (e) Target Image

Figure 5: A set of intermediate collision images using the PhotoDNA hash function, with visualizations of their respective PhotoDNA hashes. The left image shows the original source image, and the rightmost image represents the desired target image. Each hash distance represents the difference between that image and the target image. The intermediate images show generated collisions at hash distances of 1797, 963 and 342 respectively. Image quality is only slightly degraded even for very low hash distances.



(a) Start: 120 (b) Step 300: 88 (c) Step 800: 38 (d) Step 1600: 0 (e) Target Image

Figure 6: A set of intermediate collision images using the PDQ hash function, with visualizations of their respective PDQ hashes. The left image shows the original source image, and the rightmost image represents the desired target image. Each hash distance represents the difference between that image and the target image. The intermediate images show generated collisions at hash distances of 88, 38 and 0 respectively. It is easy to see that image quality is greatly reduced as hash distance decreases.

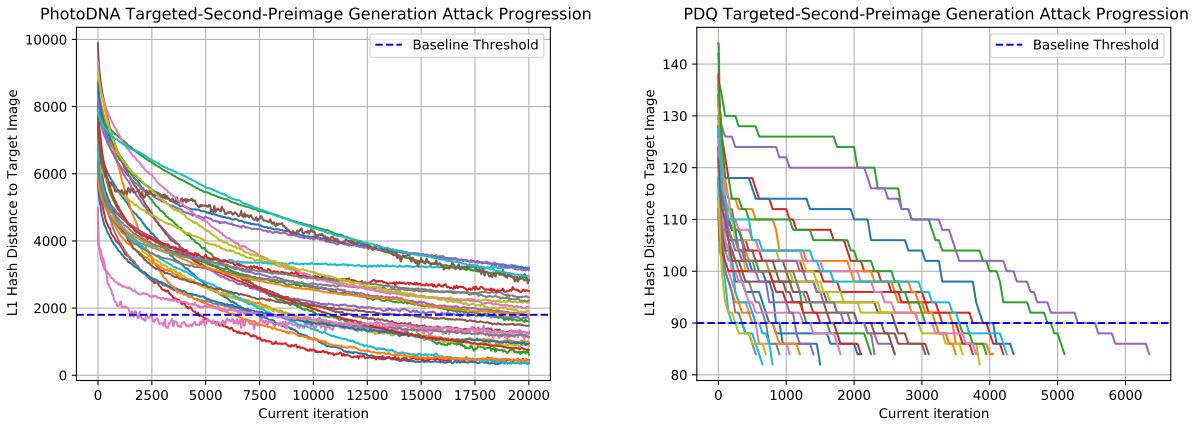


Figure 7: Targeted Second-Preimage-Generation Attack progression for 30 image pairs selected at random from ImageNet validation set [36]. PhotoDNA (left) achieved the baseline collision threshold within 20k iterations for 17 image pairs, PDQ (right) is able to reach the baseline threshold for all 30 runs.

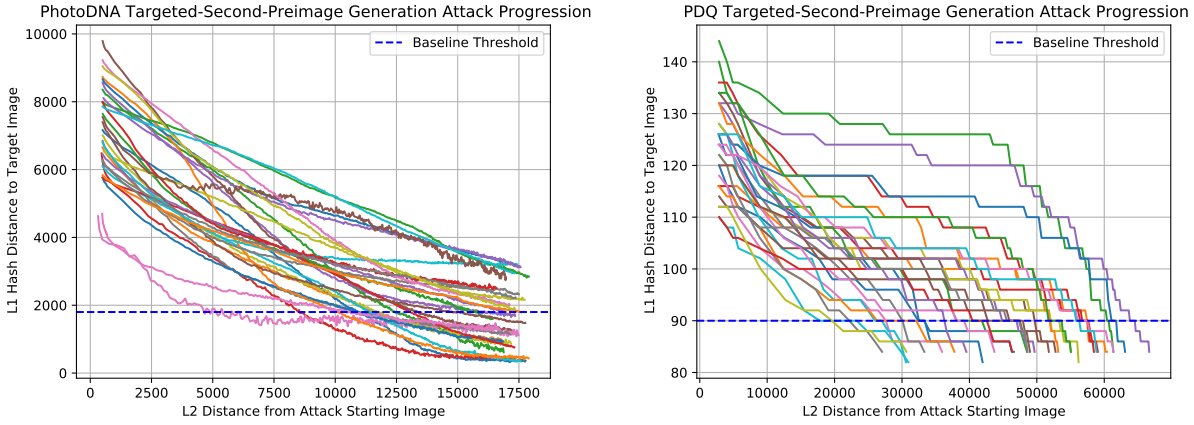


Figure 8: Targeted Second-Preimage-Generation Attack for the same 30 image pairs as above showing the relationship between L_1 hash distance and L_2 attack image distance for PhotoDNA (left) and PDQ (right).

(a) Starting Image
L_2 Dist: 0

(b) $\Delta_d = 1800$ (BL)
L_2 Dist: 15.2

(c) $\Delta_d = 4000$
L_2 Dist: 40.2

Figure 9: Detection avoidance attack on PhotoDNA. The starting image is on the left. The center image is the baseline avoidant image with a hash distance of 1800 from the starting image, and the rightmost image is an avoidant image with a hash distance of 4000. The images also show the L_2 similarity metric between the starting image and each avoidant image.

baseline threshold Δ_d within 20,000 iterations. For PDQ, all 30 image pairs reached the baseline threshold within 6,350 iterations.

Figure 7 shows the distance from the hash of the attack image to the hash of the target image as it progresses over the duration of the attack. Due to the hash domain being significantly smaller for PDQ, small changes to the attack image do not affect the hash as much, requiring increased hyperparameters. This causes our attack to learn in coarse steps for PDQ, never taking a step which increases the hash distance, whereas PhotoDNA has more fine-grained direction, allowing for smoother convergence.

We also would like to understand how the quality of the image changes as the attack progresses. We use the L_2 distance metric between the source image and the perturbed attack image for this purpose. We plot this L_2 distance versus the L_1 hash distance in Fig. 8. We can see that as the L_2 distance increases (*i.e.*, image quality degrades), the hash distance decreases. In other words, the worse the image, the better the collision.

Two hyperparameters differed between our attacks - PDQ required a learning rate of 5 and δ of 100 whereas PhotoDNA required 0.5 and 1 respectively to converge[8]. Due to these changes, additional learning on PDQ past the baseline led to noisy images, thus runs were stopped early. All image pairs that converged (17 for PhotoDNA, 30 for PDQ) can be found in Appendix A.

Attack runtime. We aimed for a 4 hour completion time for each of our targeted-second-preimage attacks in our progression experiments above. On average each of our PhotoDNA attacks took approximately 4 hours to complete 20,000 iterations, after which 17 of our 30 randomly-selected image pairs converged (*i.e.*, $\Delta_d < 1,800$) On the other hand, our PDQ attacks achieved convergence (*i.e.*, $\Delta_d < 90$) for all 30 images in approximately 3 hours on average, corresponding to 600–2,000 iterations. The PhotoDNA attack time is not optimal: the need to execute PhotoDNA as a binary on CPU no doubt increased the attack time significantly, and further optimization and parallelization is likely to substantially improve overall runtime.

5.3.2 Detection Avoidance

Jain et al. [19] evaluated a detection evasion attack against PDQ and other hashes but not PhotoDNA; we complement their work by presenting results on PhotoDNA. Using the methodology described in Section 4,

[8]This refers to the perturbation parameter δ described in [8]. This has no relation to our hash distance notation Δ_d.

we are able to generate images with hashes that satisfy the matching thresholds we chose in the previous section. We attempt to generate an image whose distance from the original is above the matching threshold Δ_d while remaining perceptually similar to it.[9] We ran the attack against PhotoDNA for 300 iterations and presented the image with the lowest L_2 distance to the target image.

Attack runtime. Unlike targeted-second-preimage generation, detection avoidance was relatively quick to execute. Our attack against PhotoDNA took less than 1 hour.

Results. Figure 9 shows an image of a boat, perturbed such that the PhotoDNA hash distances from the original image are above 1800 and 4000, respectively. Both of the perturbed images remain perceptually equivalent to the original, whereas the distance in hash space increases beyond the baseline $\Delta_d = 1800$ to even remain relatively clear at $\Delta_d = 4000$.

6 Related Work

Adversarial inputs. Adversarial inputs ("adversarial examples") are human-imperceptible perturbations to inputs of machine learning (typically deep neural networks) that cause dramatic, unexpected changes to outputs [7,15,41]. Attacks rely on gradient-guided optimization of function inputs, and most attacks compute the gradients directly which is impossible for most PHMs as they are non-differentiable. Some "label-only" attacks estimate gradients from black-box queries [8,16,33], and our own sampling-based method is inspired by one such method recently suggested by HopSkipJumpAttack [8]. Note that even black-box methods do not trivially extend to PHMs, most of which are non-learned functions very different in construction than neural networks and do not necessarily exhibit neural networks' locally-linear property that underlies the success of adversarial perturbations [15].

Learned-encoding privacy. The notion of hiding secrets within obscure, poorly-understood representations produced by deep learning has gained popularity in several contexts, and has already attracted strong criticism. The local updates sent in "private" federated learning in lieu of sensitive data were found to leak the data's secret attributes [27], and "deep representations" used by neural networks, which are a popular way to represent data due to their low dimensionality and (mis)perceived privacy assurances, contain a surprising amount of extractable information, to the point where raw text can be extracted from the text's embedding [34, 40].

Instance hiding for machine learning. Another line of work proposes to encode instances such that individual-instance attributes cannot be recovered, but sets of encoded instances can still be used to train machine-learning models. Conceptually, such functionality supersedes that of PHM, assuming it allows training an accurate matching function between two encoded instances. Xiao and Devadas [43] laid a theoretical foundation to such a notion of private learning, and proposed a framework for encoding instances called Dauntless. Two additional works implemented concrete instance-hiding frameworks and also published encoded datasets as public challenges: InstaHide [17] mixed images with others, and then applied a random bit-wise mask. More recently, NeuraCrpyt [44] used a randomly-selected, secret neural network to encode instances. Both of these approaches are completely broken: in both cases, Carlini et al. [5, 6] essentially recovered all images in their publicly released challenges, and also include impossibility results for a stronger notion of instance encoding privacy [5].

The above attacks do not work on PHMs, and they recover information on encrypted instances, not compute second preimages nor images that avoid detection.

[9] This likely results in an image that does not collide with *any* image in the flagged-images database: see [19, §7].

7 Conclusion

Perceptual hash functions have become an increasingly important component of the modern communications infrastructure, generally with only limited consideration of their properties. In the past this has been an acceptable tradeoff, because hash matching was performed online and confidentiality was not guaranteed by encrypted communications systems. In the next era of increasingly E2EE communications systems, the use of hash-based matching in E2EE protocols may carry significantly higher risks. Our results demonstrate that these functions are vulnerable to machine-learning based attacks that produce both collisions and avoidant images, and that these attacks have the potential to violate the guarantees of E2EE communication participants. Going forward this observation should help guide the designers of future PHF/PHM systems, and inform the debate around the viability of deploying these systems within encrypted communication mechanisms.

Our results leave several open problems. While we focused on PhotoDNA and PDQ, there are several other hash functions that, while less widely-deployed in PHM systems, could also be analyzed using our techniques. Additionally, in this work we did not evaluate whether it was possible to extract information from PhotoDNA and PDQ hashes. Finally, there are many questions facing practitioners and policymakers surrounding the design of future scanning systems, if these systems are to be built.

Acknowledgements

The authors would like to acknowledge support from the NSF under awards CNS-1955172, CNS-1653110, CNS-1854000 and CNS-1801479, as well as from the Office of Naval Research under contract N00014-19-1-2292, DARPA under Contract No. HR001120C0084, a Security and Privacy research award from Google, a gift from LexisNexis Risk Solutions, and support from JPMorgan Chase & Co. Any opinions, views, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Government, DARPA, JPMorgan Chase & Co. or its affiliates, or other sponsors.

References

[1] Apple. Apple CSAM detection. https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf, 2021.

[2] William P. Barr et al. Open letter: Facebook's "privacy first" proposal. https://www.justice.gov/opa/pr/attorney-general-barr-signs-letter-facebook-us-uk-and-australian-leaders-regarding-use-end, 2019.

[3] Abhishek Bhowmick, Dan Boneh, Steve Myers, Kunal Talwar, and Karl Tarbe. The Apple PSI system.

[4] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2: new generation of memory-hard functions for password hashing and other applications. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 292–302. IEEE, 2016.

[5] Nicholas Carlini, Samuel Deng, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Shuang Song, Abhradeep Thakurta, and Florian Tramer. An attack on instahide: Is private learning possible with instance encoding? *arXiv preprint arXiv:2011.05315*, 2020.

[6] Nicholas Carlini, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Florian Tramer. Neuracrypt is not private. *arXiv preprint arXiv:2108.07256*, 2021.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[8] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. HopSkipJumpAttack: A query-efficient decision-based attack. In *2020 IEEE Symposium on Security and Privacy*, pages 1277–1294. IEEE, 2020.

[9] Department of Justice. Two Former Twitter Employees and a Saudi National Charged as Acting as Illegal Agents of Saudi Arabia. Available at https://www.justice.gov/opa/pr/two-former-twitter-employees-and-saudi-national-charged-acting-illegal-agents-saudi-arabia, November 2019.

[10] Brian Dolhansky and Christian Canton Ferrer. Adversarial collision attacks on image hashing functions. In *Workshop on Adversarial Machine Learning in Real-World Computer Vision Systems and Online Challenges (AML-CV)*, 2021.

[11] Facebook. ThreatExchange GitHub repository. https://github.com/facebook/ThreatExchange/tree/master/pdq.

[12] Facebook. Community standards enforcement report, q2 2021. https://transparency.fb.com/data/community-standards-enforcement/, 2021.

[13] Hany Farid. Reining in online abuses. *Technology & Innovation*, 19(3):593–599, 2018.

[14] Christina Garman, Matthew Green, Gabriel Kaptchuk, Ian Miers, and Michael Rushanan. Dancing on the lip of the volcano: Chosen ciphertext attacks on apple imessage. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 655–672, 2016.

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *arXiv preprint arXiv:1412.6572*, 2014.

[16] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR, 2019.

[17] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning. In *International Conference on Machine Learning*, pages 4507–4518. PMLR, 2020.

[18] INRIA. INRIA CopyDays Dataset.

[19] Shubham Jain, Ana-Maria Cretu, and Yves-Alexandre de Montjoye. Adversarial detection avoidance attacks: Evaluating the robustness of perceptual hashing-based client-side scanning. *USENIX Security Symposium*, 2022.

[20] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometry consistency for large scale image search. In *Proceedings of the 10th European conference on Computer vision*, October 2008.

[21] Jan Kaiser. jPhotoDNA GitHub repository. https://github.com/jankais3r/jPhotoDNA, 2021.

[22] Jan Kaiser. PyPhotoDNA GitHub repository. https://github.com/jankais3r/pyPhotoDNA, 2021.

[23] Neal Krawetz. Tweet dated August 19, 2021.

[24] Neal Krawetz. PhotoDNA and its limitations, 2021.

[25] Anunay Kulshrestha and Jonathan Mayer. Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation. In *30th USENIX Security Symposium (USENIX Security 21)*, Vancouver, B.C., August 2021. USENIX Association.

[26] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. CONIKS: Bringing key transparency to end users. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 383–398, Washington, D.C., August 2015. USENIX Association.

[27] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 2019.

[28] Microsoft. Microsoft PhotoDNA Match Edge Hash POST API documentation. https://developer.microsoftmoderator.com/docs/services/57c7426e2703740ec4c9f4c3/operations/596ea1487ecd9f1ba408c32f.

[29] Microsoft. PhotoDNA. https://www.microsoft.com/en-us/photodna, 2018.

[30] Microsoft. PhotoDNA Cloud Service Terms of Use, 2021.

[31] Muhammad Shahroz Nadeem, Virginia NL Franqueira, and Xiaojun Zhai. Privacy verification of photodna based on machine learning. IET Digital Library, 2019.

[32] Ellen Nakashima. Chinese hackers who breached Google gained access to sensitive data, U.S. officials say. *The Washington Post*, May 2013.

[33] Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. *arXiv preprint arXiv:1612.06299*, 2016.

[34] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1314–1331. IEEE, 2020.

[35] Paul Rosler, Christian Mainja, and Jorg Schwenk. More is less: On the end-to-end security of group chats in signal, whatsapp and threema. In *EuroS&P '18*, 2018.

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[37] SCSC under the MOND. Assessment of cybersecurity of mobile devices supporting 5g technology: Analysis of products made by Huawei, Xiaomi and OnePlus. https://www.nksc.lt/doc/en/analysis/2021-08-23_5G-CN-analysis_env3.pdf, 2021.

[38] Priyanka Singh and H. Farid. Robust homomorphic image hashing. In *CVPR Workshops*, 2019.

[39] Irwin Sobel. An isotropic 3x3 image gradient operator. Presentation at Stanford A.I. Project, 1968.

[40] Congzheng Song and Ananth Raghunathan. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 377–390, 2020.

[41] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[42] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[43] Hanshen Xiao and Srinivas Devadas. Dauntless: Data augmentation and uniform transformation for learning with scalability and security. *IACR Cryptol. ePrint Arch.*, 2021:201, 2021.

[44] Adam Yala, Homa Esfahanizadeh, Rafael GL D' Oliveira, Ken R Duffy, Manya Ghobadi, Tommi S Jaakkola, Vinod Vaikuntanathan, Regina Barzilay, and Muriel Medard. Neuracrypt: Hiding private health data via random neural networks for public training. *arXiv preprint arXiv:2106.02484*, 2021.

A Additional Examples

The following pages contain the results of our additional experimentation as described in Section 5.3.1. We ran our Targeted-Second-Preimage Generation attack on 30 randomly selected image pairs from the ImageNet Validation dataset [36]. All image pairs which converged below the baseline Δ_d (17 pairs for PhotoDNA and 30 for PDQ) are shown in Figs. 10 to 18 and Figs. 19 to 33 respectively.

(a) L_1 hash distance $\Delta_d = 367$

(b) L_1 hash distance $\Delta_d = 976$

Figure 10: PhotoDNA Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 613$

(b) L_1 hash distance $\Delta_d = 454$

Figure 11: PhotoDNA Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 1686$

(b) L_1 hash distance $\Delta_d = 1251$

Figure 12: PhotoDNA Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 1121$

(b) L_1 hash distance $\Delta_d = 1136$

Figure 13: PhotoDNA Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 866$

(b) L_1 hash distance $\Delta_d = 342$

Figure 14: PhotoDNA Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 940$

(b) L_1 hash distance $\Delta_d = 434$

Figure 15: PhotoDNA Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 1596$

(b) L_1 hash distance $\Delta_d = 758$

Figure 16: PhotoDNA Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 1798$

(b) L_1 hash distance $\Delta_d = 1478$

Figure 17: PhotoDNA Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 1523$

Figure 18: PhotoDNA Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 84$, Iterations: 700

(b) L_1 hash distance $\Delta_d = 84$, Iterations: 4.05k

Figure 19: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance and between attack image (top) and target image (bottom) is displayed below each pair, along with the number of iterations required for the attack image.

(a) L_1 hash distance $\Delta_d = 84$, Iterations: 5.1k

(b) L_1 hash distance $\Delta_d = 84$, Iterations: 4.2k

Figure 20: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 84$, Iterations: 6.35k

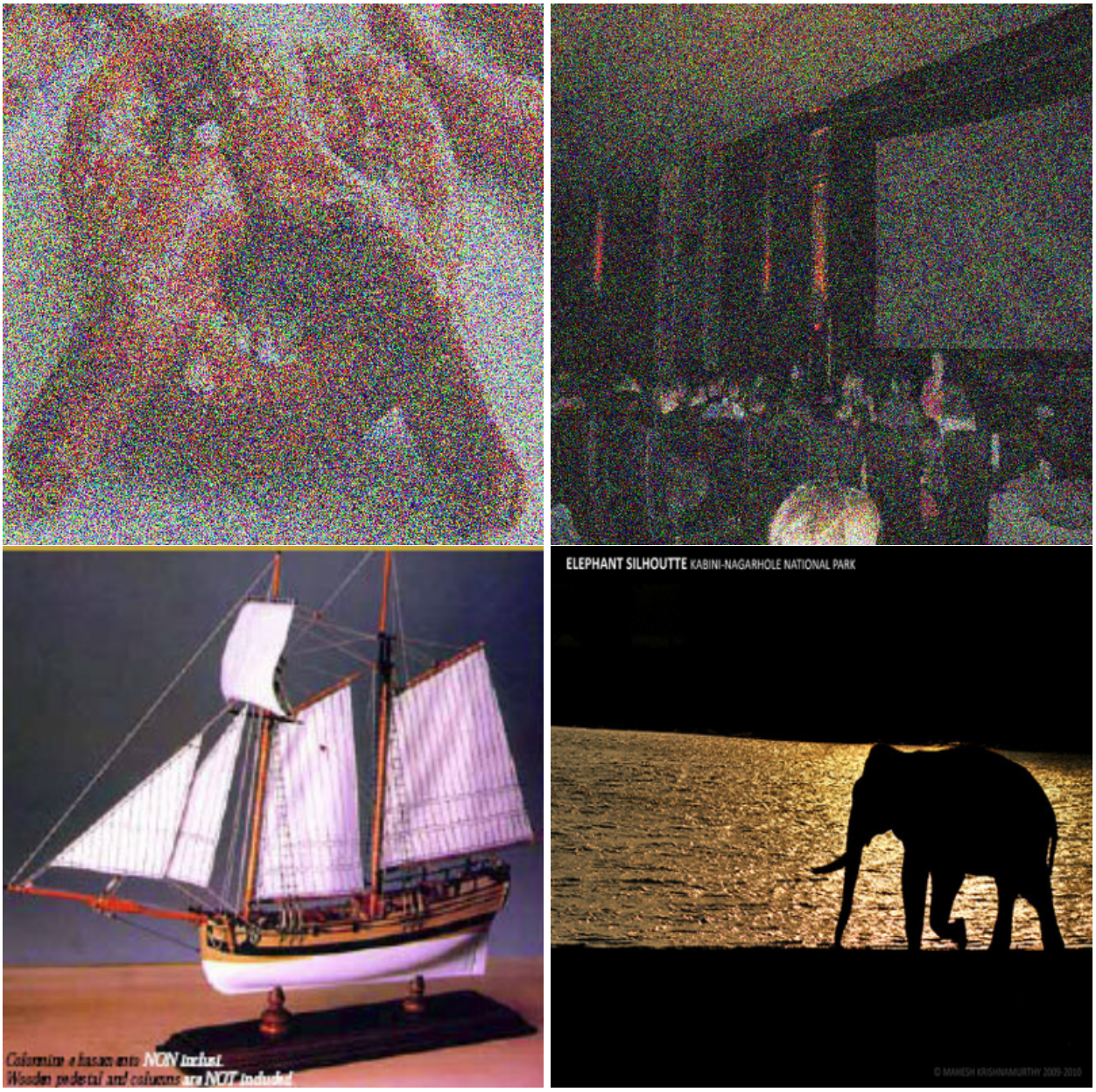
(b) L_1 hash distance $\Delta_d = 84$, Iterations: 3.05k

Figure 21: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 84$, Iterations: 1.05k

(b) L_1 hash distance $\Delta_d = 84$, Iterations: 900

Figure 22: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.



(a) L_1 hash distance $\Delta_d = 82$, Iterations: 3.85k

(b) L_1 hash distance $\Delta_d = 82$, Iterations: 800

Figure 23: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 82$, Iterations: 1.5k

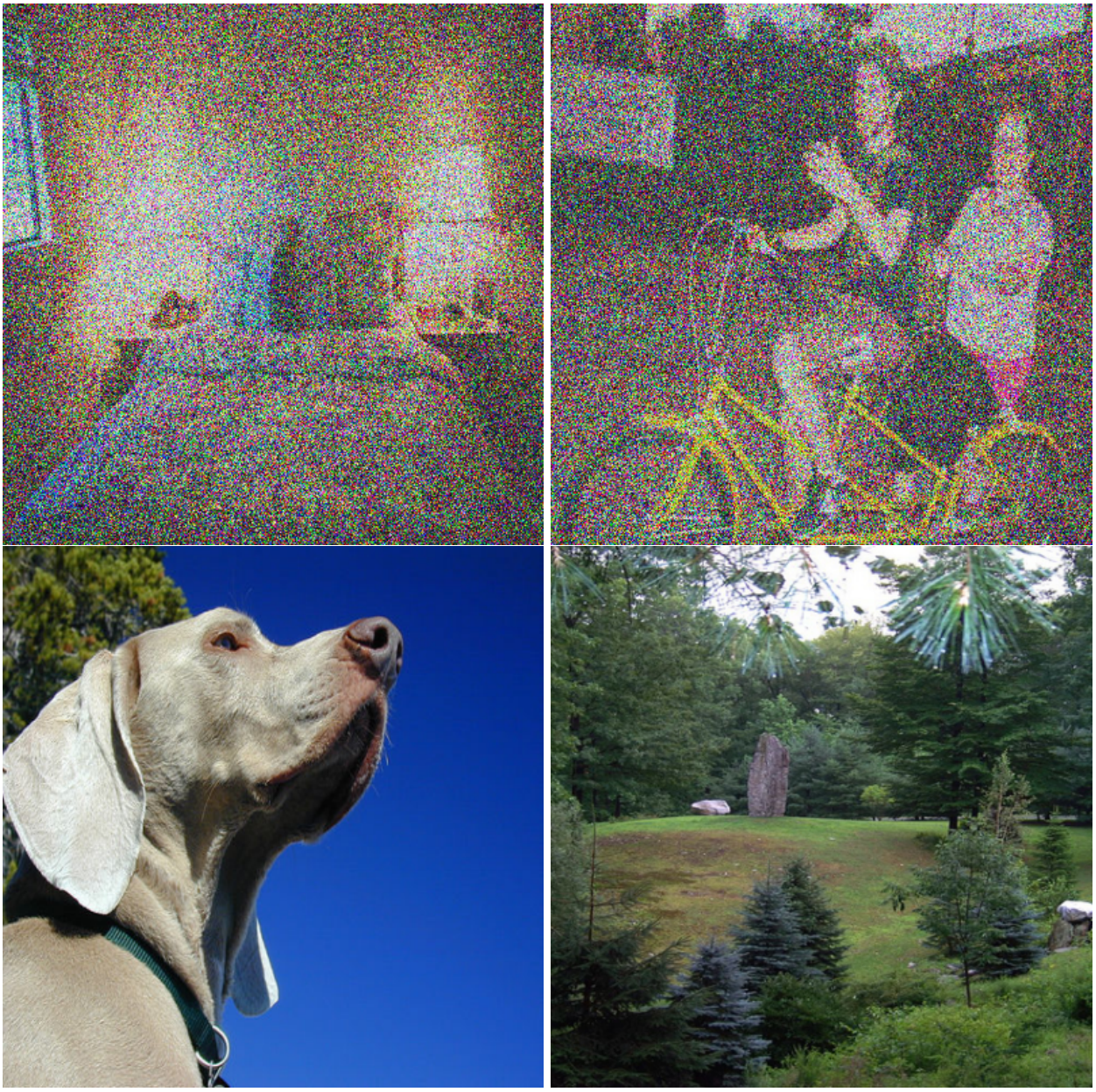
(b) L_1 hash distance $\Delta_d = 84$, Iterations: 1.2k

Figure 24: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 84$, Iterations: 2.25k

(b) L_1 hash distance $\Delta_d = 84$, Iterations: 3.75k

Figure 25: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.



(a) L_1 hash distance $\Delta_d = 84$, Iterations: 2.6k

(b) L_1 hash distance $\Delta_d = 84$, Iterations: 3.1k

Figure 26: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 84$, Iterations: 1.8k

(b) L_1 hash distance $\Delta_d = 84$, Iterations: 550

Figure 27: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 84$, Iterations: 3.6k

(b) L_1 hash distance $\Delta_d = 82$, Iterations: 650

Figure 28: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 84$, Iterations: 4.35k

(b) L_1 hash distance $\Delta_d = 84$, Iterations: 3.5k

Figure 29: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 84$, Iterations: 3.95k

(b) L_1 hash distance $\Delta_d = 84$, Iterations: 2.1k

Figure 30: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 84$, Iterations: 1.4k

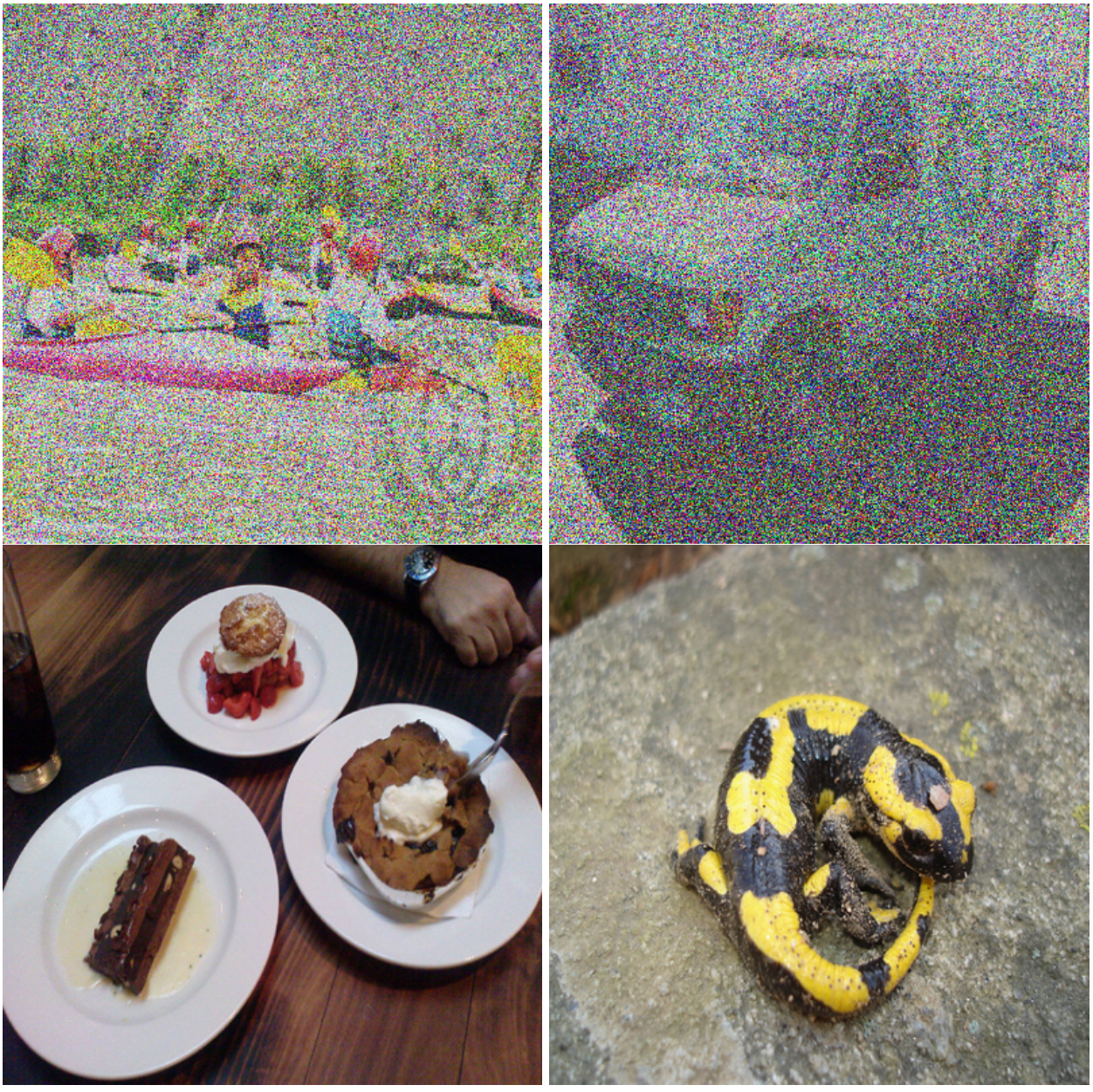
(b) L_1 hash distance $\Delta_d = 84$, Iterations: 2.6k

Figure 31: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.

(a) L_1 hash distance $\Delta_d = 84$, Iterations: 3.8k

(b) L_1 hash distance $\Delta_d = 84$, Iterations: 2.3k

Figure 32: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.



(a) Hash Distance: 84, Iterations: 2.1k

(b) L_1 hash distance $\Delta_d = 84$, Iterations: 4.25k

Figure 33: PDQ Targeted-Second-Preimage Collision pairs. L_1 hash distance between attack image (top) and target image (bottom) is displayed below each pair.