Squint Hard Enough: Attacking Perceptual Hashing with Adversarial Machine Learning

Jonathan Prokos[1*], Neil Fendley[2], Matthew Green[1], Roei Schuster[3], Eran Tromer[4], Tushar M. Jois[1], 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 [17, 38]. 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 [38] or PDQ [16] 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.

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]

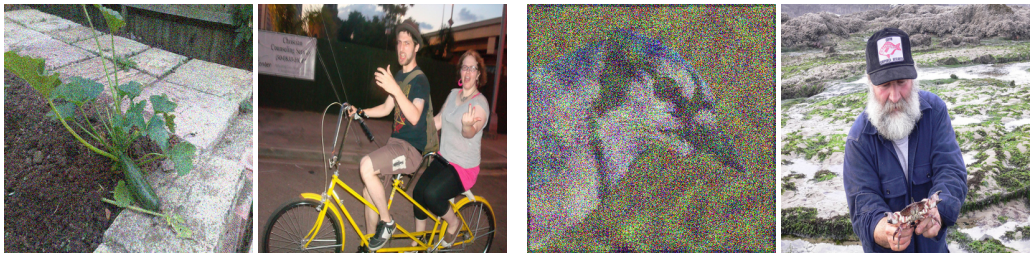
While there has been a great deal of research into the properties of perceptual hashing algorithms, surprisingly little work has considered these functions from a privacy or security perspective [14]. This reflects a common understanding that perceptual hash matching systems *are not expected to resist adversarial exploitation.* Indeed, this warning is given explicitly within the specification of the few widely-used perceptual

[*]Currently affiliated with Two Six Technologies, LLC (Arlington, VA).

[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) [1]. Providers often ban the accounts of reported users, even in cases where law enforcement determines the finding to be a false-positive [29].

(a) PhotoDNA Collision at $\Delta_d = 1720$

(b) PDQ Collision at $\Delta_d = 86$



(c) PhotoDNA Collision at $\Delta_d = 342$

(d) PDQ Collision at $\Delta_d = 38$

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.

hash algorithms [16] 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 do not further violate confidentiality. The adversarial creation of collisions, including second-preimages, 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 [32, 39].

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 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 [3].[3] This spurred providers and researchers to investigate the problem of deploying hash-based content scanning *within E2EE protocols*. Several protocols were subsequently published [2,33,53]. 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

[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 [39].

[3]More recently, the EU Commission formally presented a proposal [15] to mandate CSAM scanning capabilities in messaging applications, raising questions about whether this would apply to E2EE applications [34].

hash-based matching into the encryption mechanism itself. The security of the underlying E2EE communications system now *fundamentally depends 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.

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 counting false positives when considering a natural image corpus. However as a component of an encryption system, 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 sensitive protected speech such as a political campaign image. This attack might cause a PHM system to trigger improperly on content that is clearly permitted, 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 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 [38] and Facebook's PDQ [16]. 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 [2] and in the research literature [33, 53]. 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. 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) 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 [10] 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 [27, 28, 38] 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[4], demonstrating attack viability.

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

Targeted-second-preimage on Facebook's PDQ. PDQ [16] 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 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 [2]. 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. Jain et al. [25] evaluated PHM robustness against detection-avoidance attacks, but did not consider second preimages nor did the authors examine PhotoDNA. Finally, in a September 2021 blog post, Krawetz published a purported description of the PhotoDNA hash function [32] and claimed (without providing complete details) the ability to extract useful imagery from PhotoDNA hashes.

Ethical considerations. The analysis of perceptual hash functions raises a number of challenging ethical questions. Although these functions have many applications, they are a critical ingredient in deployed (server-side) scanning systems that are used today to detect content such as CSAM, terrorist media, and image-based sexual abuse (colloquially known as "revenge porn".) Moreover, both the design and security properties of the PhotoDNA function have for several years been carefully shielded from public analysis: the owners of the technology (NCMEC and Microsoft) require cooperating organizations to sign a non-disclosure agreement as a condition for obtaining the function implementation [39,42]. While neither organization explicitly discusses the motivation for this NDA requirement, it is reasonable to assume that these organizations are concerned about the possible impact of public analysis on the effectiveness of these systems.

On the opposing side, the research community has long argued that public and adversarial examination is essential to ensuring the correct operation of security systems, adhering to the view that "security by obscurity" is a brittle and ineffective approach to protecting high-value systems. The present work is primarily motivated by recent interest in deploying perceptual hash functions within E2EE messaging systems [2–4, 15]. As discussed in previous sections, the security guarantees in these systems may be undermined by an attacker that is able to manufacture false positives in perceptual hash functions. Since client-side scanning proposals are being actively debated for near-term deployment and these hash functions may be included in such proposals, it is our view that examining the risk posed by these systems is now a matter of public concern. Hence we believe that the benefit from conducting this analysis is sufficient to compensate for any near-term harm that might occur due to this publication.

Nonetheless, to minimize any potential harm from this work, we employ the following precautions: (1) we do not publish the design of previously-confidential algorithms: all of the software analyzed in this work is either public (PDQ [16]), or was previously extracted and published by other parties [27, 28, 32]. (2) We do not publish confidential algorithm details or parameters in this work, except where necessary to support our scientific results. (3) While we intend to provide our source code on request to reviewers and researchers, we do not intend to publish our attack code to the public. And finally, (4) we have disclosed our results to both Facebook and Microsoft.

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.[5]

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

[5] 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.

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 paradigm has been used to implement keyword-based scanning in some Xiaomi phones [51].

Edge hashing. An alternative approach, exemplified by Microsoft's "Edge Hash" [37], 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 [4, 18, 33, 53]. 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 [53] or on private set intersection [2, 33]. A variant of this approach was recently prototyped by Apple [2] and is scheduled to be deployed in Apple's iCloud Photos system soon.

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 if 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 [4, §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 [38] 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 [38] 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 [38], as well as more complete purported descriptions derived from reverse-engineering and/or partial descriptions published by researchers [31, 32]. In 2020 a binary implementation of PhotoDNA was extracted from forensic software [27, 28], 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 [32]. 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 [54], a type of 1-dimensional convolutional filter based on the 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 [18]) [6]; to evaluate the appropriate bound Δ_d we conduct experiments with false-positive rates in Section 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 [16].

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. It then computes 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, it emits a 1 if the corresponding element of transform is greater than the median, otherwise it emits 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 [16]. 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 treat the hash as a black box.

[6] Initial experimentation utilizing the L_2 metric for comp suggested negligible difference in performance and thus was not further evaluated.

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

Algorithmic weakness. E2EE-PHM systems are potentially vulnerable to attacks in which a surveillance adversary constructs meaningful files that have *semantically non-equivalent* content, and yet are identified as a match to the underlying PHM system.

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; however, a collision of perceptually-similar images is not useful for the surveillance attacks below. A second-preimage attack may not be useful either, if the generated preimages are "garbage" images that would not pass muster with 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.

Targeted collisions. In this special form of second-preimage attack, the attack algorithm takes as input a known *target digest* D corresponding to the hash of an unknown *target image* T, as well as a known *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: $\mathsf{comp}(H(T'), D) < \Delta_d$.

Given the ability to practically generate targeted collisions, two forms of surveillance attacks are possible.

Surveillance threat model 1. The attacker is able to send files via some communication medium which does not reveal identity of recipients (e.g., a bulletin board or an anonymous messaging system). The attacker wishes to deanonymize these recipients. The attacker is assumed to be capable of crafting and posting innocuous-looking files to that medium, such that recipients are likely to forward these via the same medium (with significant probability). The communication medium is assumed to implement E2EE-PHM, and the attacker knows the digest D of some image in the E2EE-PHM database. Moreover, the attacker is assumed to get notified of positive detec-

tion of that digest: for example, attacker may be a nefarious E2EE-PHM service provider, may have acquired access to the service provider's system via an intrusion or an insider; or may partially control an organization (such as NCMEC) which receives notifications.[7]

Surveillance attack 1. Under these assumptions, the attacker will first create a targeted collision where the starting image S is some innocuous-looking image that is likely to be circulated by the targeted users, and the target digest is D (i.e., the digest of some illicit image). The attacker then transmits the resulting image T' to the targeted users over the communication medium. Any user who takes the bait, and forwards T' via the E2EE-PHM system, will be (mis)detected as having sent the illicit file. The attacker gets notified of these false positives and can deduce who transmitted T' and when.

Surveillance threat model 2. The attacker wishes to detect any transmission of some specific image T in a communication medium which implements E2EE-PHM. The attacker is assumed to know the digest D of the image T (but not necessarily T itself). The attacker is assumed to be able to cause the addition of an illicit image of its choice to the E2EE-PHM database (e.g., by presenting it to the service provider, or to an organization such as NCMEC that is trusted to curate the database), and to be notified of positive detection of that image (as above).

Surveillance attack 2. Under these assumptions, the attacker will first create a targeted collision where the starting image S is some illicit image of their choice, and the target image is the aforementioned image T being surveilled (specified by its digest D). The resulting image T' is perceptually similar to the illicit image S, and thus the attacker can cause it to be added to the E2EE-PHM database. Subsequently, whenever a user transmits file T through the private channel, the transmission is flagged by the PHM system (as a false-positive match of T') and the attacker is notified.

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 [19, 35, 48], 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 [13, 43].[8]

3.2 User Framing and Censorship

A variant of the previous attack scenario considers an adversary who wishes to use targeted collisions to harm a specific user, or to harm anyone who distributes certain legitimate context, by causing them to be flagged by the E2EE-PHM system. We consider the same algorithmic weakness as above, *i.e.*, collisions between semantically non-equivalent images.

Framing threat model. The adversary aims to frame an unwitting user for trafficking in illicit content, using seemingly-innocuous images to do so. 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). The adversary can cause an illicit-looking 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).

Censorship threat model. The adversary desires to prevent legitimate image T from being freely communicated. The adversary knows the digest D of T. As above, the adversary can cause illicit-looking images to be added to the PHM-based content scanning database.

Framing or Censorship attack. In both cases of the above threat models, the adversary can proceed as follows. It finds a target collision, setting the source image S to some arbitrary illicit image, and the target digest to D. The resulting image T' is perceptually similar to S and can thus be added to the database; but its digest is close to the target image T. Henceforth, anyone who sends the image T will be flagged as a (false-positive) match of the illicit image T'.

Censorship attack. The adversary can use 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 to the database; but its digest is close to the target image T, causing any distribution of T to be flagged as distribution of T', and will be flagged by the system.

3.3 Illicit-Content Detection Avoidance

Algorithmic weakness. E2EE-PHM systems flag a communication if there is a match with some illicit content. In practical terms, illicit content digests are stored in some database. This is essentially a "deny list": a match with a hash in the deny list is blocked. In client-local matching, these deny lists are placed on device. Access to this deny list creates a risk that dishonest users will abuse this knowledge to evade detection.

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

[8] Most notably, in 2019 several Twitter customer service employees were prosecuted for allegedly spying on critics of the Kingdom of Saudi Arabia [13]. Content moderation employees and their infrastructure, including operations outsourced to third-party firms, represent an ideal access point for sophisticated attackers.

Such access can cause even greater harm if the PHM scheme allows for two files with *semantically equivalent* content to have *different* digests. This vulnerability would constitute a total break in the entire E2EE-PHM system, allowing an attacker to bypass the deny list with arbitrary content.

Avoidance threat model. The attacker aims to circumvent *DB*, a perceptual hash database of illicit content, and distribute illicit content under an E2EE-PHM system. The attacker has access to *DB*, and can query a candidate image and determine if it is present in *DB*. The attacker can query *DB* locally, without generating an externally-visible trace.

Simple avoidance attack. 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*, the attacker can simply compute the digest of their illicit image and check it against *DB* before sending it.

Perceptual detection avoidance attack. A more nefarious attack is to create a different image that is perceptually similar but avoids detection. 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; moreover, further dissemination of those images by others will not be flagged either (until a different *DB* or detection system is encountered).

3.4 User Data Leakage

Algorithmic weakness. 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. We develop two attacks on PHM schemes that would facilitate such leakage from edge-hashing E2EE-PHM.

User data leakage threat model. In this attack model, the attacker wishes to recover attributes of the unknown input image, S, given its digest $D = H(S)$.

Preimage reconstruction attack. The strongest user data leakage attack is recovery of the original image S used to compute the digest $D = H(S)$. This setting is somewhat analogous to the notion of preimage attack resistance in cryptographic hash functions, but with several differences. First, even information-theoretically 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 [5]); rather, we wish to generate a meaningful "realistic" image.

Preimage attribute recovery attack. A weaker (but still harmful) attack is when the attacker recovers meaningful attributes of the image. We define this as an attribute function, $\mathtt{attr}$, which corresponds to some aspect of the image, such as identifying objects, locations, or number of people. Given $\mathtt{attr}$ and $D = H(S)$, the attacker attempts to recover the value of $\mathtt{attr}(S)$. For example, $\mathtt{attr}$ may be an image-contents label implied by a classification dataset, and in the context of E2EE-PHM, this could reveal the topic of an encrypted conversation between users from the categories of images they exchange.

3.5 Illicit-Content Data Leaks

Algorithmic weakness. 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. 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.

Illicit content leakage threat model. The attacker's goal is to deduce the attributes of, or fully reconstruct, illicit content given only their digests. We consider an attacker who has access to *DB*, a perceptual hash database of illicit content digests 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.

Illicit content leakage attack. 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.

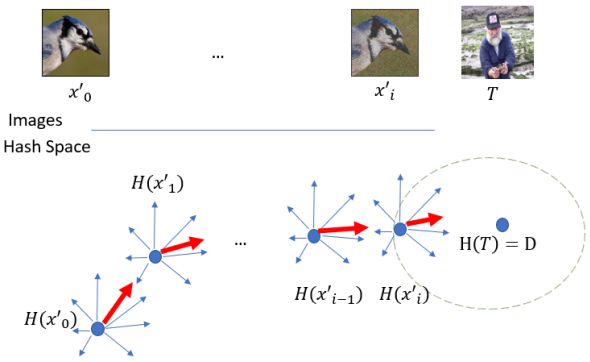


Figure 2: Visualization of the attack and gradient estimation from Section 4.1 in image and hash space. The blue arrow at $H(x'_i)$ represents query image changes to the hash and the red arrow represents the calculated gradient. Since H is non-continuous, each red arrow is a local approximation of the gradient and may not point directly at the target hash $H(T)$.

4.1 Targeted-Second-Preimage Attack

Given a digest, we adapt a gradient-based optimization technique to find second preimages. 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. There is no need to know the original image T (and in a real-world setup, an attacker likely does not, see Section 3.1). We will 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 this using indirect Monte-Carlo approximations of the hash-function gradients to minimize loss terms over hash distances and perturbation sizes. Figure 2 illustrates this.

Our attacker's optimization objective is to find a minimal δ such that: $\mathsf{comp}(H(S+\delta),H(T)) < \Delta_d$

Algorithm 1 Targeted-Second-Preimage Attack

Parameters: N : num iterations, P : num projections γ : learning rate
Input h: target hash, x_0 : starting image, Hash() : hash function, Dist() : distance function
currentDistance =Dist(h,Hash(x_0))
for $i = 1,2,\ldots N$ **do**
  δ_i =calcGrad(x_{i-1},h,P)
  $x_i \leftarrow x_{i-1} + \gamma\delta_i$
  currentDistance = h−Hash(x_i)
  if *currentDistance* $< \Delta_d$ **then**
    Hash(x_i) is a successful collision with h
  end if
end for

Gradient-based optimization. Algorithm 1 describes our optimization procedure. 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 $\delta \leftarrow x'_m - T$.

A natural choice for δ_0 is a vector proportionate to the negative gradient vector $-\frac{\partial H}{\partial x'_0}$. Let $\nabla_x(\mathsf{comp}) \equiv -\frac{\partial H}{\partial x}$ then $g_i = \gamma\nabla_{x'_i}(\mathsf{comp})/\|\nabla_{x'_i}(\mathsf{comp})\|$, i.e. g_i is the gradient in point x'_i, normalized and multiplied by a learning rate γ that modulates the change size. For $\delta_i, i > 0$, we set δ_i as a weighted average between the current gradient and a momentum term [25] given by the previous update, i.e., $\delta_i \leftarrow \rho * \delta_{i-1} + (1-\rho) * g_i$.

Algorithm 2 calcGrad

Parameters: q: num projections, λ: perturbation param
Input: h: target hash, x_i : image, Hash() : hash function, Dist() : distance function
for $j = 1,2,\ldots q$ **do**
  $p_j \sim \mathcal{N}(0,1)$
  $c_j \leftarrow Dist(Hash(x_i+\lambda p_j),h) - Dist(Hash(x_i),h)$
end for
$g_i \leftarrow \frac{1}{q}\sum_1^q c_j \cdot p_j$
$\delta_i \leftarrow Norm(g_i)$

Monte Carlo gradient estimation. Algorithm 2 describes our gradient-optimization procedure. Since hashes are not necessarily differentiable, we use a gradient estimate via Monte Carlo sampling 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 *mutations* $p_1,\ldots,p_q$, each with the same dimension of the input image, via random element-wise sampling from a 0-mean distribution. For each mutation j we calculate the change in our objective $c_j \leftarrow \mathsf{comp}(H(x'_i+\lambda p_j),H(T)) - \mathsf{comp}(H(x'_i),H(T))$ where λ controls the perturbation magnitude. These terms are then used as the coefficients to calculate a weighted average of all the mutations to serve as the gradient. $g_i \leftarrow \frac{1}{q}\sum_1^q c_j \cdot p_j$

A higher number of samples q corresponds to increased approximation accuracy.

Additional variants. We experiment with two additional variants of the above optimization: first, the *grayscale variant* where perturbation pixels are constrained to have equal values across color channels. Intuitively, this might (1) reduce human-perceptible chroma-noise during the attack, and (2) reduce the optimization domain's dimensionality without impeding performance, as PhotoDNA casts images (and our perturbations) to grayscale prior to computing the hash.

Second, the *double-sample* variant, also known as antithetic sampling, tries to incorporate two samples instead of one for each mutation. Here, we only sample $q/2$ random perturbations in each round $p_1,\ldots,p_{q/2}$, and set the rest of the

q mutations to the negative values of the sampled ones, i.e., $(p_{q/2+1},\ldots,p_q) \leftarrow (-p_1,\ldots,-p_{q/2})$. As shown to be empirically effective in [23, 50], our Monte Carlo estimator will build on a consistent correspondence between a perturbation direction and the function's value. If correspondence is consistent, we expect that going in a certain direction in space, by applying a certain mutation, will have the inverse effect of going in the opposite direction. By sampling each mutation's positive-sign and negative-sign effects, we can mitigate the inclusion of mutations whose direction presents inconsistent behavior, for example when going in either the direction or its inverse, both leads to a decrease in the objective. This will lead to the mutation's coefficient terms in g_i's computation to partially cancel each other out, thus mitigating their effect.

Difference from black-box adversarial ML. Using Monte Carlo to estimate gradients is a known black-box optimization technique [61], recently used by HopSkipJumpAttack [10] and others [23] for finding adversarial examples. Our setting is somewhat different from the standard black-box adversarial ML setting, where the adversary is trying to fool a classifier to cause it to make a misprediction. Attacks like HopSkipJumpAttack use a binary value indicating whether impersonation to a target class is successful as a basis for their Monte-Carlo estimation – assuming the attacker can trivially find the decision boundary. In contrast, we try to fool a hash-distance computation instead of a classifier, and cannot make such an assumption. However, our attack can gain comparatively finer-grained success measures by utilizing hash distances. We thus adapt our gradient estimator to use raw distances instead of a binary decision. We can adapt other black-box optimization approaches in this fashion; in Section 5.4, we do this and compare them to our chosen method.

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. 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 [27, 28], 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 [27, 32] to a high-level description of PhotoDNA by one of the algorithm's authors [18]. 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 [16].

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

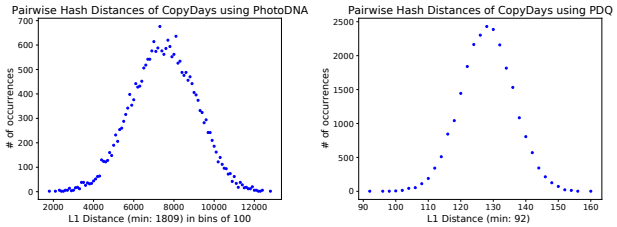


Figure 3: Frequency of pairwise hash distances derived from 157 perceptually distinct images' pairs from CopyDays dataset. PhotoDNA shown in bins of 100. Used to determine threshold for a hash match.

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 [24].

	PhotoDNA Precision	PhotoDNA Recall	PDQ Precision	PDQ Recall
Original	1	1	1	1
JPEG/5,8	1	0.43	1	0.54
JPEG/50,75	1	0.34	1	0.34
Crops/10,15	1	0.71	1	1
Crops/70,80	1	0.43	1	0.99
Strong	1	0.51	1	0.53

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 [16]. These involve computing the pairwise distances of a set of the hashed CopyDays dataset, which consists of 157 perceptually-distinct images [24, 26]. The resulting 24,492 pairwise results are plotted in Fig. 3 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. 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 [60], 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 [32].

5.3 Attack Results

We implement the following two attacks utilizing images from the ImageNet Validation dataset [49][9].

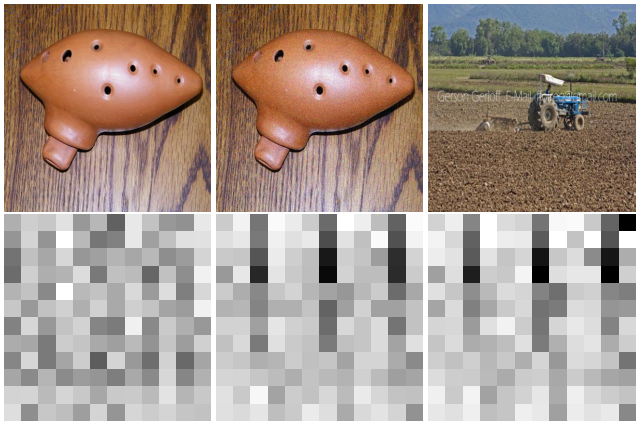
5.3.1 Targeted-Second-Preimage Generation

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 4 and 5 show two such example images, with total hash distances $1,788$ and 88 respectively[10]. We ran our attack on 30 randomly selected image pairs from ImageNet [49], producing hash collisions significantly below our baseline Δ_d. Of these pairs, we are able to reach the baseline $\Delta_d$for 17/30 runs within 20k iterations under PhotoDNA along with a minimum collision distance of 342 as shown in Fig. 6. For PDQ we are

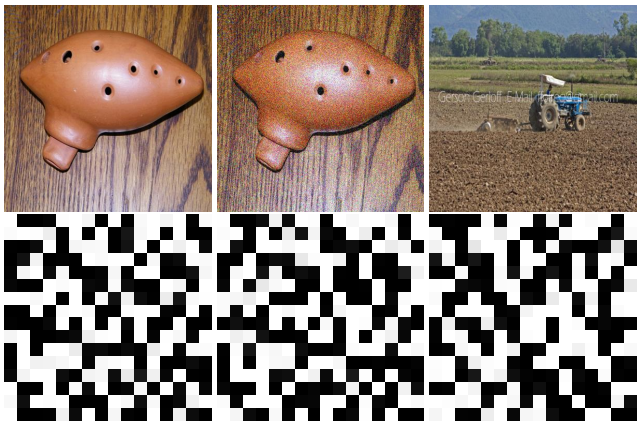
[9]This is a common image dataset, and has been used in prior perceptual hashing work [25, 42].

[10]All image pairs which reached the baseline Δ_d will be available to view at perceptualhashing.lol.



(a) Starting Image Δ_d: 6363, L2: 0 (b) Baseline Collision Δ_d: 1788, L2 37.92 (c) Target Δ_d: 0

Figure 4: Example targeted second preimage attack on PhotoDNA. The leftmost image shows a source image, while the rightmost image shows the chosen target image. The center image contains a near-collision with $\Delta_d < 1800$.



(a) Starting Image Δ_d: 112, L2: 0 (b) Baseline Collision Δ_d: 88, L2: 90.54 (c) Target Δ_d: 0

Figure 5: Example targeted second preimage attack on PDQ. The leftmost image shows a source image, while the rightmost image shows the chosen target image. The center image contains a near-collision with $\Delta_d < 90$.

Table 2: Comparison of Targeted-Second-Preimage Generation attack variants as mentioned in Section 4.1. Success rate is the percentage of runs for which the hash distance was below Δ_d. "Image distance" corresponds to the source images' average perturbation size (L2 norm size).

(a) PhotoDNA Ablation Study.

PhotoDNA	Success Rate	Image Distance
Baseline	0.833	47.66
Grayscale	0.866	59.33
ρ 0.5 Grayscale	0.900	42.81
ρ 0.5 Grayscale Double-Sample	0.833	37.78

(b) PDQ Ablation Study.

PDQ	Success Rate	Image Distance
Baseline	1	106.48
Grayscale	1	112.16
ρ 0.5	1	77.62
ρ 0.1 Grayscale	1	108.02
ρ 0.25 Grayscale	1	102.16
ρ 0.5 Grayscale	1	88.14
Grayscale Double-Sample	1	124.98

able to reach the target $\Delta_d$for all 30 image pairs within 7k iterations. Allowing the attack to run for higher iterations is able to produce a perfect hash collision for all tested images but with significant visual noise as shown in Fig. 7, even at comparatively fewer iterations. Attack progression is shown in Fig. 8 along with an evaluation of visual noise added to the starting image shown in Fig. 9.

Ablation study. We run a parameter sweep on the attack and its variants discussed in Section 4.1. We evaluate the effect of grayscale updates, momentum ρ at various levels, and *double-sample* variant. Table 2 reports these results. PhotoDNA succeeds most frequently utilizing $\rho = 0.5$ with grayscale updates and produces the clearest collision utilizing $\rho = 0.5$ with grayscale updates and double-sample. Since all runs are able to achieve perfect collisions within 20,000 iterations, we also measure perurbation sizes to evaluate their quality. $\rho = 0.5$ with grayscale updates performs the best. The reader can visually appreciate the small-perturbation-size collisions in Figs. 10 and 11.

Attack Configuration. 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 shown in Fig. 8. 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. Due to these changes, additional learning on PDQ past the baseline Δ_d led to noisy images, thus runs were stopped early – although all baseline runs were able to reach a perfect collision with enough iterations.

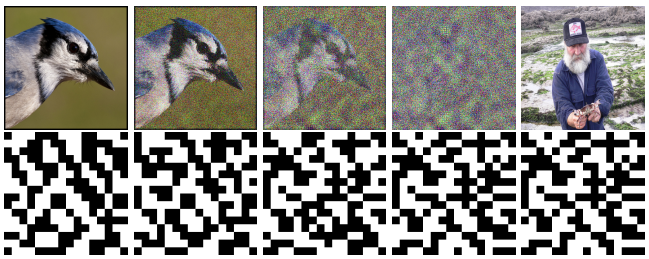
Attack Runtime. We aimed for a 4 hour completion time for each of our targeted-second-preimage attacks. Our PhotoDNA attacks took approximately 4 hours to complete 20,000 iterations while our PDQ attacks achieved convergence (*i.e.*, $\Delta_d < 90$) for all 30 image pairs in approximately 3 hours, corresponding to 600–6,000 iterations. Neither attack time is not optimal: the need to execute PhotoDNA as a binary and calculate PDQ utilizing pure Python operations on CPU no doubt increased the attack time significantly, and further optimization and parallelization will likely improve runtime.

5.3.2 Detection Avoidance

Jain et al. [25] 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, we are able to generate

(a) Initial $\Delta_d : 6162$ (b) $i = 8.2k$ $\Delta_d : 1797$ (c) $i = 12k$ $\Delta_d : 963$ (d) $i = 20k$ $\Delta_d : 342$ (e) Target

Figure 6: A set of intermediate collisions generated at step i between the initial (left) and target (right) image pair shown at varying Δ_d thresholds using the PhotoDNA hash function. Each image (top) is shown with visualizations of their respective PhotoDNA hashes (bottom). Image quality is only slightly degraded even for very low hash distances.



(a) Initial $\Delta_d : 120$ (b) $i = 300$ $\Delta_d : 88$ (c) $i = 800$ $\Delta_d : 38$ (d) $i = 1600$ $\Delta_d : 0$ (e) Target

Figure 7: A set of intermediate collisions generated at step i between the initial (left) and target (right) image pair shown at varying Δ_d thresholds using the PDQ hash function. Each image (top) is shown with visualizations of their respective PDQ hashes (bottom). Image quality is greatly reduced as hash distance decreases.

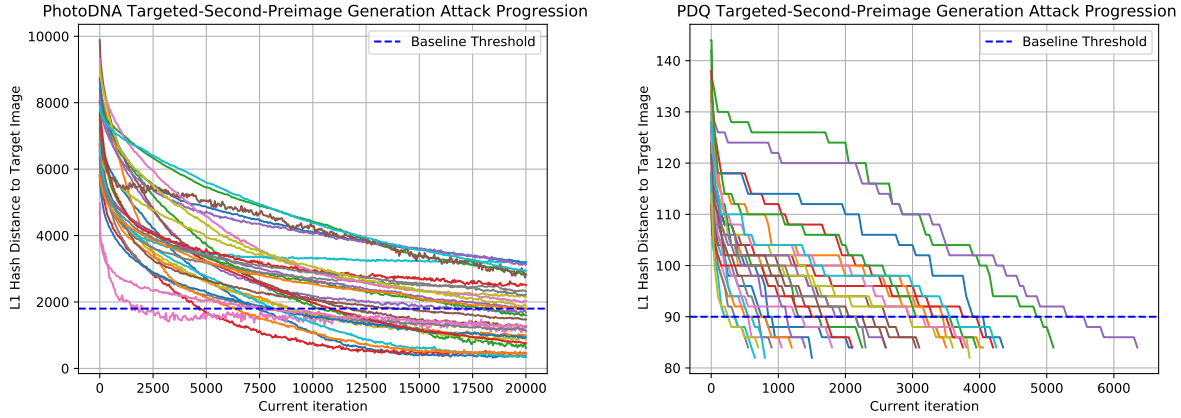


Figure 8: Targeted Second-Preimage-Generation Attack progression for 30 image pairs selected at random from ImageNet validation set [49] using PhotoDNA (left) and PDQ (right) – baseline Δ_d shown with dashed blue line.

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.[11] We ran the attack against PhotoDNA for 300 iterations and presented the image with the lowest L_2 distance to the target image. Our attack against PhotoDNA took less than 1 hour.

Results. Figure 12 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 $\Delta_d = 4000$.

[11] This likely results in an image that does not collide with *any* image in the flagged-images database: see [25, §7].

5.4 Comparison to Prior Work on Adversarial ML

Prior black-box adversarial-ML work includes a plethora of variants that optimize a black-box function by perturbing its input. We identify 3 prominent groups of such methods, implement a strong candidate variant for each, and compare our method against it. We note that our choice of baselines is not meant as a comparison against an exhaustive set of attack families, as this is a huge space (e.g. we do not evaluate finite-difference gradients [11], "hybrid" approaches that use models to guide querying [12, 58], rejection sampling [6], and many others). We aimed to choose a set of representative baselines that show that not all methods even work, and that our approach defeats natural baselines.

Particle Swarm Optimization (PSO). These evolutionary algorithms maintain a "swarm" of perturbations and iteratively move them in randomized directions, guided by the

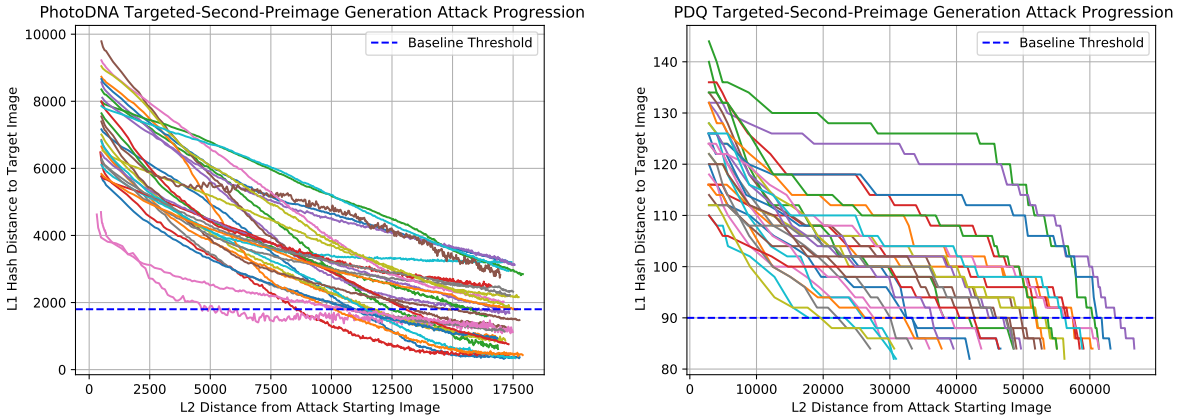


Figure 9: Targeted Second-Preimage-Generation Attack for the same 30 image pairs as above showing the relationship between L_1 hash distance and L_2 attack image distance for PhotoDNA (left) and PDQ (right).

quality ("fitness") of previously-observed perturbations. This is a classic technique in evolutionary algorithms [30] that has been popular for black-box adversarial ML [41, 52]. We use the PySwarms package [40] to implement this method in our setting. As in Section 4.1, we adapt this method to minimize hash distances by setting the fitness method as the distance, i.e., $-\mathsf{comp}(H(\cdot), H(T))$.

Natural Evolution Strategies (NES). These methods use mutations to attain *estimated gradients* and then apply standard gradient-based optimization [10, 23]. Again, the general approach was developed regardless of adversarial ML, due to Wierstra et al. [61]. Our own method (Sec. 4.1) belongs in this category. As a comparison candidate, we considered [61]'s multinomial-distribution search algorithm, but this is computationally infeasible as the covariate matrix would be of size $(400 \times 400 \times 3)^2$. Instead, we used a second algorithm from [61], based on estimating the inverse Fisher information matrix at each step and applying it as a coefficient to the gradient update.

Surrogate-based approaches. It is interesting to examine if attacking a proxy model trained by the adversary to imitate the behavior of the black-box one, which can be done in a white-box fashion, would produce second-preimages that transfer to the black-box model. This is also a well known technique [46, 47, 59]. To train our own surrogate model, we chose a neural architecture designed to compute perceptual hashes, namely NeuralHash [56]. We modify it to produce hashes of the same shape as our target hash algorithm (144 and 256 for PhotoDNA & PDQ respectively). We then train fit this network to minimize the Mean Squared Error between its output and outputs of our target algorithm's hashes on ImageNet images [49]. Once the model has been trained, we construct a fully-differentiable version of the hash-distance computation, and run a white-box attack on this model.

Table 3: Comparing our approach with prior work. Success rate is the percentage of runs for which the hash distance was below Δ_d. "Image distance" corresponds to the source image's perturbation size (L2 norm size).

(a) PhotoDNA Prior Work Study.

PhotoDNA	Success Rate	Image Distance
Our Method	0.900	42.81
NES	0.633	38.59
PSO	0.033	281.71
Surrogate	0	N/A

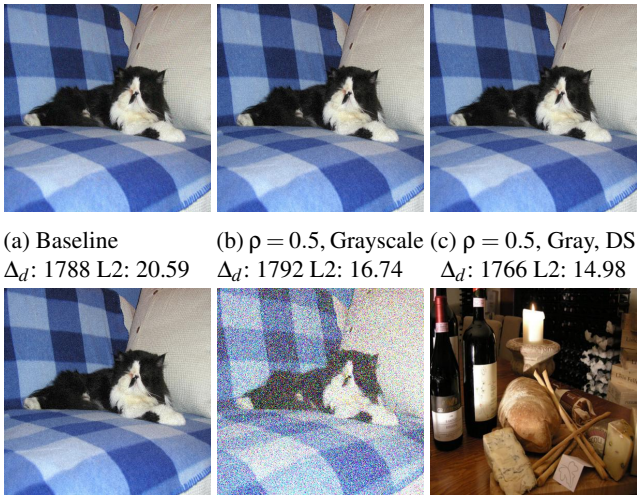
(b) PDQ Prior Work Study.

PDQ	Success Rate	Image Distance
Our Method	1	77.62
NES	1	123.23
PSO	0.366	351.21
Surrogate	0	N/A

Evaluation and results. We run a preimage attack against the same 30 images as mentioned above Section 5.3.1. Table 3 reports our results and compares against our own method. The surrogate-based attack fails completely, due to the surrogate model poorly fitting to the hash function. This is somewhat expected, since learning complex functions with ample nonlinear behavior and open output domains is highly nontrivial. PSO sometimes succeeds in evasion, but rarely so. NES succeeds at a rate more comparable to our attack, and even defeating some of our variants (compare to figures in Table 2). The reader can also find a visual comparison in Figs. 10 & 11.

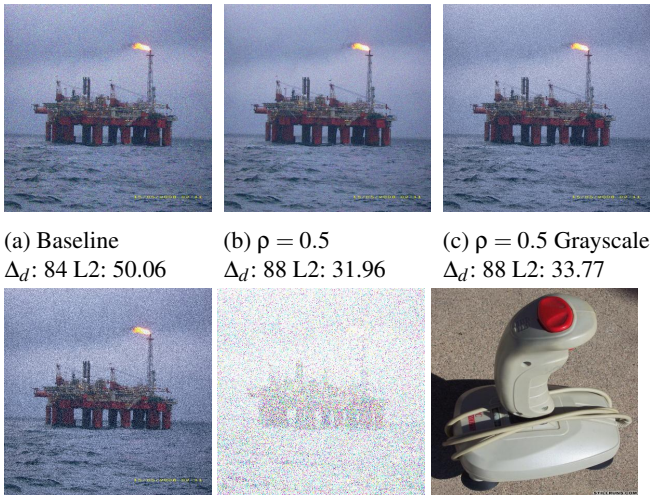
6 Related Work

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 [9, 20, 57]. Most attacks compute the gradients directly which is impossible for most PHMs as they are non-differentiable. Black-box attacks either use surrogate models [46, 47, 59] or run an evolutionary search algorithm [10, 21, 44] which often includes estimating the gradients. We explain how black-box attacks against ML classifiers can be adapted to be applied against PHMs (see Section 4.1), but prior to this work it was not



(a) Baseline Δ_d: 1788 L2: 20.59
(b) $\rho = 0.5$, Grayscale Δ_d: 1792 L2: 16.74
(c) $\rho = 0.5$, Gray, DS Δ_d: 1766 L2: 14.98
(d) NES Δ_d: 1774 L2: 18.05
(e) PSO Δ_d: 2575 L2: 203.44
(f) Target Image

Figure 10: Comparison between the best PhotoDNA Targeted-Second-Preimage collisions generated using various attack variants. Results shown in Table 2.



(a) Baseline Δ_d: 84 L2: 50.06
(b) $\rho = 0.5$ Δ_d: 88 L2: 31.96
(c) $\rho = 0.5$ Grayscale Δ_d: 88 L2: 33.77
(d) NES Δ_d: 80 L2: 61.34
(e) PSO Δ_d: 78 L2: 299.56
(f) Target Image

Figure 11: Comparison between the best PDQ Targeted-Second-Preimage collisions generated using various attack variants. Results shown in Table 2

(a) Starting Image L_2 Dist: 0
(b) Δ_d = 1800 (BL) L_2 Dist: 15.2
(c) $\Delta_d = 4000$ L_2 Dist: 40.2

Figure 12: 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.

clear that they will actually work, because PHMs are very different from ML classifiers. Specifically, ML adversarial attacks often assume you can randomly query to find an input for a target class. PHMs have highly non-smooth surfaces and input-obfuscating transformations inspired by those of cryptographic hashes, so it not easy to find a preimage input for a target output. This causes most black-box ML attacks to fail. Our comparison with prior work suggests that some approaches indeed fail completely, but that others like gradient estimation work well, and that our own approach significantly outperforms baselines. See Section 5.4.

Private representations (and extraction attacks). There are several other recent examples where obscure, poorly-understood representations were used with the assumption that they provide some guarantee, usually privacy, but without any rigorous analysis supporting this, and attracted widespread criticism. This includes for example the local updates sent in "private" federated learning that leak sensitive attributes [36], "deep representations" that were misperceived as private but found to contain a surprising amount of extractable information [45, 55], or "private-learning instance encodings" [22, 62, 63] that were completely broken by instance-recovery attacks [7, 8]. These recovery attacks do not work on PHMs, and they recover information on encrypted instances, not compute second preimages nor images that avoid detection.

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] 18 U.S. Code §2258A - Reporting requirements of providers. https://www.law.cornell.edu/uscode/text/18/2258A.

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

[3] 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.

[4] Abhishek Bhowmick, Dan Boneh, Steve Myers, Kunal Talwar, and Karl Tarbe. The Apple PSI system. https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf.

[5] 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.

[6] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[7] 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.

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

[9] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (Oakland '17)*, pages 39–57. IEEE, 2017.

[10] 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.

[11] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[12] Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Improving black-box adversarial attacks with a transfer-based prior. *Advances in neural information processing systems*, 32, 2019.

[13] 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.

[14] 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.

[15] European Commission. Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL laying down rules to prevent and combat child sexual abuse. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52022PC0209&from=EN, 2022.

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

[17] Facebook. Community standards enforcement report, Q2 2021. https://transparency.fb.com/data/community-standards-enforcement/, 2021.

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

[19] 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.

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

[21] 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.

[22] 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.

[23] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pages 2137–2146. PMLR, 2018.

[24] INRIA. INRIA CopyDays Dataset. https://lear.inrialpes.fr/~jegou/data.php#copydays.

[25] 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.

[26] 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.

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

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

[29] Kashmir Hill. A Dad Took Photos of His Naked Toddler for the Doctor. Google Flagged Him as a Criminal. *The New York Times*, August 2022.

[30] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[31] Neal Krawetz. Tweet dated August 19, 2021. https://twitter.com/hackerfactor/status/1428395744705142785?s=20.

[32] Neal Krawetz. PhotoDNA and its limitations. https://www.hackerfactor.com/blog/index.php?/archives/931-PhotoDNA-and-Limitations.html, 2021.

[33] 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.

[34] Matt Burgess. The EU Wants Big Tech to Scan Your Private Chats for Child Abuse. *WIRED*, May 2022.

[35] 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.

[36] 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 (Oakland '19)*, pages 691–706. IEEE, 2019.

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

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

[39] Microsoft. PhotoDNA Cloud Service Terms of Use. https://www.microsoft.com/en-us/PhotoDNA/TermsofUse, 2021.

[40] Lester James V. Miranda. PySwarms, a research-toolkit for Particle Swarm Optimization in Python. *Journal of Open Source Software*, 3, 2018.

[41] Rayan Mosli, Matthew Wright, Bo Yuan, and Yin Pan. They might not be giants: crafting black-box adversarial examples with fewer queries using particle swarm optimization. *arXiv preprint arXiv:1909.07490*, 2019.

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

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

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

[45] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy (Oakland '20)*, pages 1314–1331. IEEE, 2020.

[46] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[47] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[48] 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 *2018 IEEE European Symposium on Security and Privacy (EuroS&P '18)*, 2018.

[49] 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.

[50] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[51] 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.

[52] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1528–1540, 2016.

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

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

[55] 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.

[56] Lukas Struppek, Dominik Hintersdorf, Daniel Neider, and Kristian Kersting. Learning to break deep perceptual hashing: The use case NeuralHash. *arXiv preprint arXiv:2111.06628*, 2021.

[57] 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.

[58] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 742–749, 2019.

[59] Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. In *ICML*, 2018.

[60] Gregory K Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[61] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

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

[63] 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.