

Human Distinguishable Visual Key Fingerprints

Mozhgan Azimpourkivi and Umut Topkara, *Bloomberg;* Bogdan Carbunar, *FIU*

https://www.usenix.org/conference/usenixsecurity20/presentation/azimpourkivi

This paper is included in the Proceedings of the
29th USENIX Security Symposium.

August 12–14, 2020

978-1-939133-17-5

Open access to the Proceedings of the
29th USENIX Security Symposium
is sponsored by USENIX.

Human Distinguishable Visual Key Fingerprints

Mozhgan Azimpourkivi
Bloomberg LP
mojganaz@gmail.com

Umut Topkara
Bloomberg LP
topkara@gmail.com

Bogdan Carbunar
FIU
carbunar@gmail.com

Abstract

Visual fingerprints are used in human verification of identities to improve security against impersonation attacks. The verification requires the user to confirm that the visual fingerprint image derived from the trusted source is the same as the one derived from the unknown source. We introduce CEAL, a novel mechanism to build generators for visual fingerprint representations of arbitrary public strings. CEAL stands out from existing approaches in three significant aspects: (1) eliminates the need for hand curated image generation rules by learning a generator model that imitates the style and domain of fingerprint images from a large collection of sample images, hence enabling easy customizability, (2) operates within limits of the visual discriminative ability of human perception, such that the learned fingerprint image generator avoids mapping distinct keys to images which are not distinguishable by humans, and (3) the resulting model deterministically generates realistic fingerprint images from an input vector, where the vector components are designated to control visual properties which are either readily perceptible to a human eye, or imperceptible, yet necessary for accurately modeling the target image domain.

Unlike existing visual fingerprint generators, CEAL factors in the limits of human perception, and pushes the key payload capacity of the images toward the limits of its generative model: We have built a generative network for nature landscape images which can reliably encode 123 bits of entropy in the fingerprint. We label 3,996 image pairs by 931 participants. In experiments with 402 million attack image pairs, we found that pre-image attacks performed by adversaries equipped with the human perception discriminators that we build, achieve a success rate against CEAL that is at most 2 $\times 10^{-4}$%. The CEAL generator model is small (67MB) and efficient (2.3s to generate an image fingerprint on a laptop).

1 Introduction

Key Fingerprint Generators (KFGs) [2, 7, 19, 25, 34, 41] help simplify the error-prone and cumbersome task of comparing arbitrarily complex and long strings (e.g. cryptographic keys, addresses, identifiers) received from a trusted and an untrusted source, by converting it to the equivalent yet more natural task of comparing images or shorter text (i.e. fingerprints). Applications include manual key verification in SSH, OpenPGP and end-to-end encrypted applications [51], detection of Bitcoin clipboard attacks that target millions of addresses [48], device pairing, and the development of visual CAPTCHAs.

While secure KFGs do not need to generate memorable fingerprints, they need to be resilient to *collision* attacks, i.e., make it hard to find distinct input strings whose fingerprints are perceived to be identical by humans, and simultaneously minimize the time taken by a human to compare fingerprints (see Figure 1 for example collision attacks on text-based KFGs and CEAL). Tan et al. [50] have shown that Visual KFGs (or VKFGs) [5, 17, 33, 36, 41, 52], that convert input strings into images for humans to compare, outperform several text-based KFGs (e.g. [2, 25]) in terms of both collision attack detection rate and comparison time.

However, existing VKFGs do not factor in the limits of human visual perception, and how it relates to the space of images that they generate. Instead, they convert the input string (e.g., the hash of a key) to a structured image, e.g., by mapping an input byte to a specific color or shape. The inability to determine if changes in the input string will generate human-distinguishable fingerprint images, renders VKFGs vulnerable to collision attacks. Case in point, in § 10.4 we report vulnerabilities of Vash [1], a VKFG, identified as state-of-the-art in terms of usability and attack detection ability [50].

In this paper, we address the insufficient *capacity* of existing VKFGs, i.e., the small number of human-distinguishable images that they can generate. We develop CEAL, a novel approach to build effective VKFGs, and demonstrate its use by constructing a state of the art VFKG. CEAL maximizes the VKFG input bit-length, where the VKFG converts each possible input value into a fingerprint image that is human-distinguishable from those of all other inputs.

Exploring the space of human-distinguishable images is made challenging by the wide range of human visual sys-



Figure 1: Sample key hashes, shown in hex format broken across two lines, that differ in a single bit (top and bottom) and their corresponding CEAL generated images. Unlike the textual keys, the images are easy to distinguish by humans.

tems and the number of images that need to be compared. Using humans to verify the distinguishability of generated images, has further scalability problems. This suggests the need for an automatic solution (e.g. a classifier) to predict human perception in terms of image distinguishability. While deep learning seems ideally suited for this task, deep learning networks require large training datasets (e.g. 1.2 million images for Inception.v1 [49]). Our limited ability to collect ground truth labeled data will impact the accuracy of a human-distinguishability predictor, hence our ability to generate distinguishable images.

Contributions. In this paper, we leverage Generative Adversarial Networks (GANs) [18] to address the above challenges and generate realistic, attack-resilient images that are easy to compare by humans. Our choice of realistic images is motivated by previous research that has shown that the human visual system is better at distinguishing changes in images when their content is more natural [40]. We introduce CEAL, a training approach to build a strong VKFG able to generate human-distinguishable images, even when the input strings differ in a single bit. Figure 1 shows such CEAL-generated image fingerprints, for almost human-indistinguishable 123 bit key hash pairs (shown in hex format).

To address the human-distinguishability challenges, we design and train the first Human Perception Discriminator (HPD) network, a classifier that predicts whether two input images would be perceived as distinct by human verifiers. To address the high cost of human-annotated training data, we leverage the observation that the HPD only needs high precision in detecting human distinguishable images: low recall will only make it more conservative.

To increase the capacity of VKFGs, we introduce and build CL-GAN, a Deep Convolutional Generative Adversarial Network (DCGAN) [42] that takes as input a latent vector, and is adversarially trained using the HPD, to generate not only realistic but also HPD-distinguishable images.

We seek to eliminate the aforementioned inability of conventional VKFGs (e.g., Vash [1]) to generate distinguishable images when input strings are modified in only a few bits. For this, we leverage previous results on learning disentangled representations [10, 11, 16, 28, 31] to conjecture that we can decompose the latent vector into subsets of *major* and *minor* components, where major components contribute to the image distinguishability thus the capacity of the CEAL VKFG, while minor components do not (see § 7.2). To validate this conjecture, we overcome GAN instabilities and mode collapse problems [27, 35, 44] to integrate the constraints of the major and minor components into the CL-GAN training procedure, decompose the latent vector into such components, and enhance the ability of the major components to encode human distinguishability and of the minor components to enable the image generator to produce realistic images.

We experimentally identify the minimum Hamming distance between major components provided to CL-GAN, that when changed, consistently generated human-distinguishable images (see § 9.2). We use error correction codes to encode the CEAL input vectors (key hashes) into a representation that ensures HPD-distinguishability of the generated images.

Further, we show that the likelihood of finding a collision (i.e., human-perceived similar CEAL images, generated from different inputs) decreases as the distance between input latent vectors increases. Since GANs can be unstable, we also train CEAL to generate diverse images (§ 7.2).

Results. We implemented and trained CEAL using Tensorflow [6], 557 image pairs labeled by 500 crowdsourced workers, and 26,244 synthetically-generated image pairs. We show, using 402 million preimage attack instances, that it is computationally hard even for adversaries controlling all but one of the bits of the input string hash, and equipped with the HPD classifiers that we have developed, to find a collision: only between $1.86 \times 10^{-5}\%$ and $1.62 \times 10^{-6}\%$ of attack samples were identified as successful by the adversarial HPD and were confirmed by humans.

Participants in our studies took an average of 2.73s to compare similar (attack) pairs of CEAL images, 10% shorter than for Vash [1] attack images. In summary, we provide the following contributions:

- **Strong vs. Weak VKFG**. Formalize weak and strong Visual Key Fingerprint Generator (VKFG) functions. Decompose the problem of building a strong VKFG, into building a weak VKFG function, and converting a weak VKFG into a strong VKFG function [§ 3]. Show that Vash [1], a state-of-the-art VKFG [50], is not a strong VKFG function [§ 10.4 and § 10.5].
- **CEAL**. Develop the first approach to train a neural network based, strong VKFG function with built-in hash properties, that generates realistic, human-distinguishable, and attack resistant images [§ 7].
- **Human-Perception Classifier (HPD)**. Build the first classifier to predict if two GAN-generated images will

be perceived as identical or different by humans [§ 7.1].

- **CL-GAN**. Conjecture the existence of, and enforce, major and minor latent vector components, with different impacts on the human-distinguishability of generated images. Introduce and build CL-GAN, a DCGAN that enforces a maximum number of major components with human-distinguishability impact [§ 7.2 and § 7.3]. CL-GAN is small (66.7MB) and efficient (average of 2.3s to generate an image fingerprint on a MacBook laptop and 0.3s on a GPU-equipped desktop).
- **Attack Resilience**. Evaluate CEAL using a total of 402 million attack image pairs and labeled 3,226 CEAL-generated image pairs by 319 human workers [§ 10.2]. While the HPD model achieves 84% precision [§ 9.1], CEAL produces unique images, that are quickly determined by humans to be different, even when adversarially-generated [§ 10.3].

2 Related Work

Text-based KFGs transform an input key (e.g., a hashed public key) into a shorter, human readable format. The most commonly used textual KFG encodes the key into a hexadecimal or Base32 representation. On a study with 1047 participants, Dechand et al. [14] show that the hexadecimal representation is more vulnerable to partial preimage attacks.

Visual KFG (VKFG) solutions synthetically generate images to act as visual key fingerprints. For instance, Random art [41] and its implementation Vash [1] use the input key to generate a structured image (see § 10.4 for a detailed description). Other solutions, e.g., [17, 33, 36] similarly generate visual key fingerprints using a combination of colors, patterns and shapes. Avatar representation techniques such as Unicorn [52], can also be used as visual key representations [50]. WP_MonsterID [5] generates the visual representation as a collage of randomly selected parts from an existing dataset of images. However, Hsiao et al. [24] argue that an increase of the capacity of such visual solutions, requires an increase in the number of colors, patterns and shapes used, which consequently makes the images hard to distinguish. Thus, even though such solutions use an additional source of entropy (e.g., PRNG), they have not been designed to generate human-distinguishable images.

The user studies of Tan et al. [50] that compare multiple text and visual KFG solutions suggest that VKFGs can speed up the verification of key fingerprints. In their experiments, Vash [1] outperforms the unicorn solution [52] and several text-based KFGs (e.g. hexadecimal and numeric representations) in terms of both attack detection rate and comparison time. However, the attack success rate against Vash is fairly high, at 12%. In § 10.4, we study Vash, and confirm that despite its reliance on a PRNG, Vash is unable to satisfy the properties that we introduce in § 3. Particularly, we show that not all the images generated by Vash are human-distinguishable,

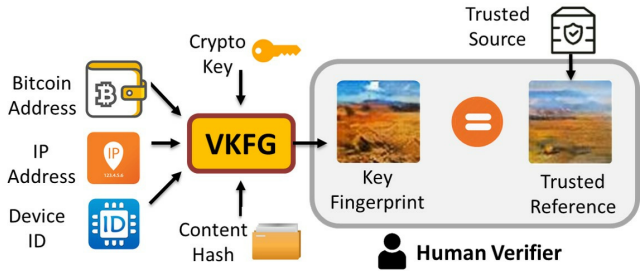


Figure 2: Visual key fingerprint generator (VKFG) model and scenario. Given an arbitrary input string, the VKFG generates an image fingerprint representation. A human verifier compares this image against a securely acquired (e.g., from a trusted site, or person-to-person) reference image fingerprint.

especially when the number of overlaid shapes and colors on the canvas increases. This is expected, as the visual sensitivity of humans to changes, diminishes with increased spatial frequency [53].

In contrast, CEAL is the first VKFG designed to ensure that the human visual system can differentiate between image fingerprints generated from different keys. This endows CEAL with resilience to adversaries that exceeds the strength assumed in state-of-the-art attacks (of Tan et al. [50] and Dechand et al. [14]). In § 10.3, we show that even adversaries who control all but one of the bits of the input string hash, achieve only a 1.7% success rate. Further, in Section 10.5, we confirm that when compared to Vash, CEAL is not only more attack resilient, but also enables faster human verifications.

3 Problem Definition

Informally, we consider the scenario depicted in Figure 2: the user is presented with two images, one acquired from a trusted source, and one generated from data received from an untrusted source (e.g., public key, shared key, Bitcoin address, IP address, domain name). The images are not necessarily available or presented on the same device. Instead, they could be displayed on different screens (e.g., of a smartphone and a laptop) or on a device and a physical medium, e.g., a printed card. To authenticate the untrusted data, the user needs to compare the two images for equality. We note that the user does not need to memorize images, but only visually compare the two images for equality.

More generally, we seek to construct a set of images, where each image can be easily and quickly distinguished from any other image in the set, by a human. Furthermore, we desire to construct a hash-like mapping function, from an input space of strings of the same size to this set of images. In the following, for simplicity, we also refer to input strings as *keys*. We represent a given key with an image, which will not be confused for another key's image representation. To prevent brute-force attacks, we require the set of images to be

large, and infeasible to store and enumerate. Therefore, we define our set through a generator, which takes an input key and outputs the corresponding element in the set. In the rest of this section, we provide a formal definition of the visual fingerprint problem, and introduce mechanisms which we have used to build our solution.

We define the set of RGB images I, and a function $HPD_{ratio} : I \times I \rightarrow [0,1]$ that captures the proportion of experiments where humans would perceive the pair of images to be distinguishable. Let $P_u^{i,j} \in \{0,1\}$, denote the result of the u^{th} human perception experiment on an image pair $I_i, I_j \in I$, $P_u^{i,j} = 1$ if and only if the human perceives the images to be different, $P_u^{i,j} = 0$ otherwise. Then, if n is the number of human experiments conducted per each image pair, $HPD_{ratio}(I_i, I_j) = \frac{\sum_{u=1}^{n} P_u^{i,j}}{n}$.

We seek to build a strong visual key fingerprint generation function $V_s : \{0,1\}^{\gamma} \rightarrow I_S$, where, $I_S \subset I$ and γ is the input string length. V_s, and thereby I_S, has the following desired property: For all binary input strings $K_i, K_j \in \{0,1\}^{\gamma}$, and their corresponding mapped images $I_i, I_j \in I_S$, where $V_s(K_i) = I_i, V_s(K_j) = I_j$,

$$K_i \neq K_j \iff HPD_{ratio}(I_i, I_j) = 1 \quad (1)$$

In practice, it is very challenging to build a generator that satisfies this strong requirement for all possible human visual systems. Instead, we propose to first build a weak visual key fingerprint generation function $V_w : \{0,1\}^{\gamma'} \rightarrow I_{\mathcal{W}}$, where $I_{\mathcal{W}} \subset I$. Let d_H denote the Hamming distance. The V_w is not able to guarantee that key pairs will be distinguishable if their d_H is within d, i.e., $E(HPD_{ratio}(I_i, I_j) \mid d_H(K_i, K_j) < d) < 1 - \varepsilon$. However, for key pairs whose d_H value is at least d, V_w is able to guarantee human distinguishability, i.e., $\forall K_i, K_j \in \{0,1\}^{\gamma'}$, and $I_i = V_w(K_i)$, and $I_j = V_w(K_j)$, we have $d_H(K_i, K_j) \geq d \iff HPD_{ratio}(I_i, I_j) = 1$.

Weak-to-strong problem decomposition. We thus decompose the problem of building a strong VKFG function into two sub-problems. First, build a weak VKFG function, and identify the minimum value d that satisfies the above requirements. Second, use the identified d to convert the weak VKFG into a strong VKFG function.

In addition to the human-distinguishability of generated fingerprints, developed solutions should also (1) have a sufficiently large *capacity* to be resistant against preimage attacks, i.e., the number of unique and human-distinguishable generated images should be large, and (2) ensure that humans are able to quickly compare any pair of generated images.

3.1 Adversary Model

We assume an adversary who attempts preimage attacks, i.e., to find input strings whose visual fingerprints will be perceived by a human verifier to be similar to the fingerprint of a specific victim. We assume that the adversary has blackbox access to the weak V_w and strong V_s functions.

We consider a (γ, d)-adversary [14], able to identify candidate strings that hash within Hamming distance $d < \gamma$ to the victim's key hash K (γ-bit long). In Section 10.3, we evaluate our solution against an adversary that controls up to 122 out of 123 bits of the input key hash, which is stronger than the 80 out of 112 bits adversary of Dechand et al. [14]. The strength of our adversary is thus more similar to that of the adversary considered by Tan et al. [50], who can perform 2^{60} brute force attempts.

4 Applications

Immediate applications of visual key fingerprints include traditional remote authentication solutions such as SSH and OpenPGP/GnuPG [9], that encode the hash of a public key into a human readable format, for manual comparison [22]. The more recent End-to-End Encrypted (E2EE) applications (e.g., secure messaging apps [51] such as WhatsApp [4], Viber [3], Facebook messenger [13]), further offer a particularly appealing use case for visual key fingerprints. To authenticate a communication peer, the user needs to visually compare the peer's public key fingerprint against a reference fingerprint that she has previously acquired through a secure channel (e.g., in person, from a trusted site, etc).

Visual key fingerprints can also be used for device pairing (e.g., Bluetooth Secure Simple Pairing using ECDH [39]), by having the user visually compare visual key fingerprint images of device keys, displayed on both paired devices.

The dependence of the HPD model performance on human annotations can be used to setup a mechanism which not only provides a non-cognitive human user detection, but also further improves the HPD. That is, the developed CEAL generator can be used to construct a matching based CAPTCHA where users are asked to mark pairs of "unmatching" images. Pairs labeled by the large number of CAPTCHA answers could then be used to build even more powerful HPD classifiers and CEAL generators, thereby setting up a self-improving mechanism. In § 11, we further discuss how adversarial interest in breaking CEAL-generated CAPTCHAs would further improve research on human visual perception.

5 Background

We now describe the architecture and training process of Generative Adversarial Networks (GANs) and error correcting codes, which we use to build CEAL.

GAN. Deep Generative Models (DGMs) are DNNs that are usually trained, using unsupervised learning, to learn and summarize key features of samples in the training data. The trained model can be used to draw samples from the modeled data distribution, i.e. generate previously unseen, but realistic and plausible instances similar to the samples in the training dataset. There are two major classes of generative

models: Variational AutoEncoder (VAE) [30] and Generative Adversarial Networks (GAN) [18].

CEAL uses a GAN model and trains a generator that takes as input a *latent vector*, i.e., a set of components randomly selected from a uniform distribution over $(-1,1)$, to produce realistic and human-distinguishable images. The conventional GAN consists of two competing neural networks: (1) a generator network (G) that transforms the input latent vector into an image, and (2) a discriminator network (D) that differentiates synthetic images, generated by G, from real images in a training dataset. G and D are trained alternately. The competition drives G to generate images that look like images from a training, real image dataset. For CEAL, we use a DCGAN [42]-like architecture to generate images that represent a key fingerprint corresponding to an input key string.

Our approach is also related to, and inspired by work on learning disentangled representations, i.e. interpretable factors of data variation [10, 11, 28, 31], that seeks to learn a representation that captures the underlying generative structure of data. For instance, InfoGAN [11] learns to disentangle visual characteristics (e.g. style, color, pose of objects, etc.) of GAN-generated images. Further, SDGAN [16] uses a supervised approach to train a GAN with latent vector components representing both identities (e.g. individual humans) and observations (e.g. specific photographs) of human faces. In addition, Grathwohl and Wilson [20] disentangle spatial and temporal features in videos, in an unsupervised fashion. We propose instead a disentanglement of major from minor components: decompose the latent vector into major and minor components, and train major components to encode information about human distinguishability, and the minor components to encode image realism properties. This captures the observation that only a subset of latent vector components are able to trigger human-distinguishable changes in generated images.

Error Correcting Codes. We use binary BCH [8, 23] codes to map a key fingerprint into the input of CL-GAN (see § 7.3 and § 9.3). A t-error correcting BCH code can correct up to t bits, and the generated code words are guaranteed to be within Hamming distance at least $d \geq 2t+1$ of each other. We represent a t-error correcting code with a message length of n and code word length of k bits as $BCH(n,k,t)$.

6 Approach

We introduce the CEAL (CrEdential Assurance Labeling) approach to build a visual key fingerprint generator that will generate realistic images and address the requirements of § 3. CEAL consists of two steps, each solving one of the sub-problems of § 3, see Figure 3 .

In the first step, we train a *generator network* to be the weak V_w function (§ 7.2). That is, the network takes as input a latent vector, and produces a realistic image, human-distinguishable from other images generated from latent vectors that are in Hamming distance at least d (see § 6). We experimentally identify d in § 9.3.

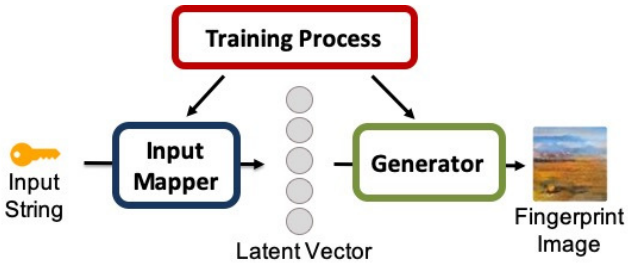


Figure 3: **The CEAL approach**: Train a generator to convert a latent vector to a realistic image, human-distinguishable from other generator produced images. Then, train an *input mapper*, that converts arbitrary input strings to latent vectors suitable for the previously trained generator.

In the second step, we build an *input mapper* that converts arbitrary input strings into latent vectors that are within Hamming distance of at least d from other mapped inputs (§ 9.3). We show that it is possible to build the input mapper, thus convert the trained weak V_w into a strong V_s function, using an error correcting code encoder, e.g., [8, 23], $ECC : \{0,1\}^\gamma \rightarrow \{0,1\}^\gamma$, with minimum distance of d. Specifically, $\forall K_i, K_j \in \{0,1\}^\gamma, K_i \neq K_j \implies d_H(ECC(K_i), ECC(K_j)) \geq d$. Then, the trained system first applies ECC to the input, then applies the V_w generator to the encoded string. This ensures that the input to V_w will always result in a human-distinguishable output, by the definition of the V_w. Therefore, $V_w \circ ECC : \{0,1\}^\gamma \rightarrow I_{\mathcal{W}'}$, where $I_{\mathcal{W}'} \subset I_{\mathcal{W}}$, and $\forall I_1, I_2 \in I_{\mathcal{W}'}, HPD_{ratio}(I_1, I_2) = 1$.

How to Train Your Generator. Having access to a HPD_{ratio} function, would allow a generator training algorithm to tap into golden annotations of which images are suitable to generate. In practice, we are not able to run a large number of human perception experiments for any given pair of images. However, given a sufficient number of annotations, a regression predictor model $HPD_{predict} : I \times I \rightarrow [0,1]$ may be used to approximate the HPD_{ratio} function, $E(|HPD_{predict}(I_1, I_2) - HPD_{ratio}(I_1, I_2)|) < \varepsilon$. We show that, even with a small number of annotated data, a very limited classification model $HPD_{equal} : I \times I \rightarrow \{0,1\}$ which can detect distinguishable image pairs with high precision at the cost of low recall, $P(HPD_{ratio} > 0 \mid HPD_{equal}(I_1, I_2) = 1) < \varepsilon$, is sufficient for training a generator which satisfies the strong V_s requirement (see Equation 1 and § 3).

In the following, let K be the input key string (see Figure 3), and let $\gamma = |K|$. The input module converts K into a latent vector L, $\lambda = |L|$, $\gamma < \lambda$. The generator network converts L into a fingerprint image. Table 1 summarizes the notations that we use in this paper.

Symbol	Description
γ	Length of input string (e.g. hash of key)
λ	Length of input latent vector (M+m)
M	Number of major components
m	Number of minor components
d	Hamming distance between two input strings
CL-GAN	CEAL training network
G-CEAL	Generator network in CL-GAN
D-CEAL	Discriminator network in CL-GAN
HPD	Human Perception Discriminator

Table 1: CEAL notations.

7 The CEAL System

We now describe the CEAL training process outlined in Figure 3. Unlike existing techniques that generate images using handcrafted rules, CEAL uses GANs (see § 5) to generate realistic, human-distinguishable images from input strings. While a DCGAN [42] can model the distribution of the training data and generate previously unseen, but realistic samples from the estimated distribution, in § 8 we show that human workers recruited from Amazon Mechanical Turk (MTurk) often cannot perceive differences between images that are generated by a DCGAN, from similar inputs.

To address this problem, we introduce CL-GAN, a DCGAN-based deep generative model that we train to generate images that are not only realistic, but also human-distinguishable. We train CL-GAN's generator network G-CEAL, using two classifiers (see Figure 4): (1) the CL-GAN discriminator (D-CEAL) that is trained to differentiate synthetically generated images by G-CEAL from a dataset of real images, and (2) the HPD classifier, trained to estimate the likelihood that a human will label a pair of images as either identical or different.

In the following, we first describe the HPD employed by CL-GAN, then detail the training process of CL-GAN. Finally, we describe CEAL's input mapper module, see Figure 3.

7.1 Human Perception Discriminator (HPD)

The Human Perception Discriminator (HPD) module takes two images as input, and computes the probability that the images are perceived as being different images by humans. We build HPD using a deep neural network (DNN). The high level architecture of the HPD classifier network (see Figure 5), is similar to a Siamese network [12]. Specifically, the HPD consists of two identical, twin networks (with shared weights). Each network accepts as input one of the two input images and passes it through the layers of a trained network.

To train a DNN with millions of parameters, we need a large training dataset of labeled samples. However, collecting labeled data is a time consuming and expensive process. To

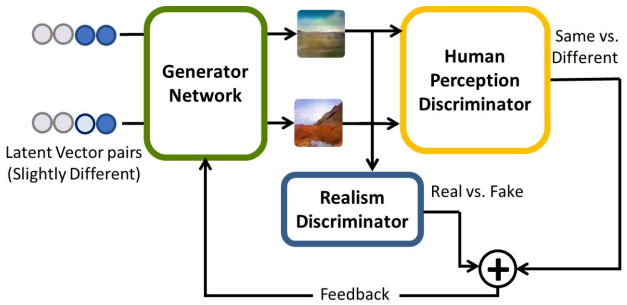


Figure 4: **CL-GAN architecture and training**. We use the combination of Discriminator loss and HPD loss to train the generator to generate distinguishable and realistic images.

address this problem, we leverage previous studies that have shown that the features that are learned by a DNN are transferable and can be used to perform similar tasks [46, 54]. This is because the representation learned by deep neural network is distributed across different layers, where shallow layers capture low level features (e.g. Gabor-like filters in the image domain), and deeper layers capture more abstract and complicated features (e.g. the face of a cat). It is common practice to use one or a combination of the representations learned by different layers of an existing network, as features for a new, related task. Experimentally investigating which representation performs best for the problem of image matching in HPD is hence a typical feature selection task in machine learning.

Specifically, we employ a transfer learning approach using the Inception.v1 model [49], trained for image classification tasks on the ImageNet dataset[15] (1.2 million images from 1,000 categories). We extract 50,176 features for each image, from the Inception.v1 network, i.e., the activations of the "Mixed_5c" layer of Inception.v1. In § 9.1, we experimentally justify the choice of this layer.

Following the Inception.v1 network, HPD consists of 3 additional fully connected layers, see Figure 5. In Section 9.1, we describe our hyper-parameter search process to find the number of layers and the number of nodes in each layer. We use these layers in order to train the HPD. This is because we cannot update the weights of the Inception.v1 layers. Instead, we optimize the weights of the 3 fully-connected layers, using weighted contrastive loss [12] with L2 regularization. The loss will enable the network to differentiate between the two images; regularization will prevent overfitting. Equation 2 shows how the weights are updated based on the weighted contrastive loss for two input samples I_1 and I_2:

$$L(\theta, Y, I_1, I_2) = \frac{1}{2}(1-r)(1-Y)(D_w{}^2) + \frac{1}{2}rY(max(0, \mu - D_w))^2 \quad (2)$$

θ denotes the model parameters (weights and biases), and Y is the actual class label of the image pair, i.e. 1 for different and 0 for identical images. D_w is the Euclidean distance between the outputs of the twin networks (O_1 and O_2 in Figure 5) for the input image pairs. $r \in [0,1]$ is the weight (importance) assigned to the positive (different) class and

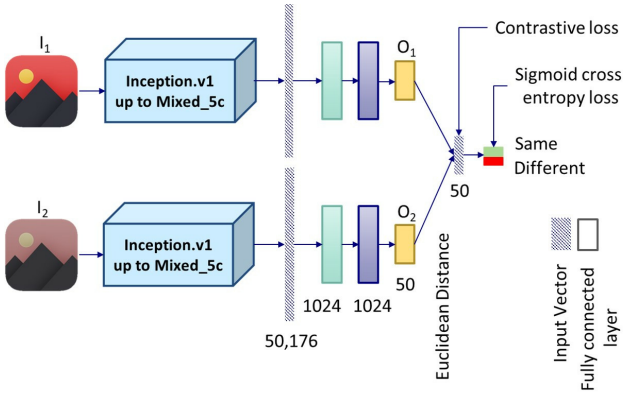


Figure 5: **Human Perception Discriminator (HPD) architecture.** HPD passes input images I_1 and I_2 through the Inception.v1 network, applies 3 fully connected layers to generate image feature vectors O_1 and O_2, computes the squared Euclidean distance between O_1 and O_2 and passes it through a fully connected layer. HPD classifies I_1 and I_2 as different or identical, based on this distance.

$\mu \in \mathbb{R}$, $\mu > 1$ is a margin.

After training the 3 additional layers in the twin Siamese network using contrastive loss, the network has learned to differentiate between the input image pairs, i.e. generate distant representations (O_1 and O_2) for dissimilar images and similar representations for similar images. We freeze the network weights and feed their derived output, i.e., the component-wise squared differences between the activations of the last layers in the twins networks, to an additional fully connected layer consisting of 1 neuron, i.e., the HPD output, with sigmoid activation function, see Figure 5. We optimize this layer's weights using a weighted cross-entropy loss and L2 regularization. The purpose of this last layer is to classify the image pairs into either the "identical" or "different" class.

7.2 Training CL-GAN

As described in § 7, we train CL-GAN's generator, G-CEAL, to generate images that are both realistic, and visually distinguishable by humans. We seek to thwart even adversaries who can generate input values K' that are at small Hamming distance from a victim K value (see § 3.1). For this, we design G-CEAL to generate fingerprint images that are visually different even when the input keys are similar. We define then the following image pair generation (IPG) process, that takes as input a seed latent vector v of length λ, and an index $i \in \{1,2,3,...,\lambda\}$, and outputs two vectors v_1 and v_2, also of length λ, that differ from v only in their i-th component:

Definition 1 *(**Image Pair Generation**: IPG(v, i)). Generate vectors v_1 and v_2, such that $v_1[i]$ = 1 and $v_2[i]$ = -1, and $v_1[j] = v_2[j] = v[j]$, $\forall j \in \{1,2,3,...,\lambda\}, j \neq i$.*

1 and -1 are extreme values of each component, which we use to maximize the component effect in generated images.

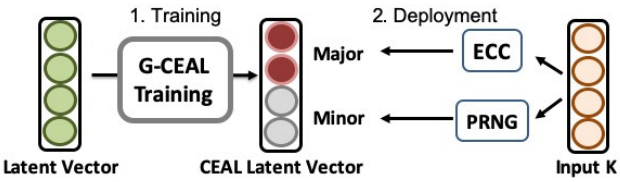


Figure 6: **Major and minor components**. The G-CEAL generator trains M components to be major components and the rest $\lambda - M$ to be minor components. Major components impact the human-distinguishability of generated images; minor components are used to generate realistic images. Input Mapper converts the input key into a latent vector.

Major and Minor Components. Preliminary experiments with DCGAN revealed that not all input bits have an equal impact on the human-distinguishability of generated images: when changed, some bits produce images that are not human distinguishable. To address this problem, we leverage recent successes in learning disentangled representations [10, 11, 16, 28, 31], to conjecture that we can decompose the latent vector into (1) a subset of *major components*, that when changed individually, produce human-perceptible changes in the generated images, and (2) the remaining, *minor components*, that encode relatively imperceptible characteristics of the images, see Figure 6.

We build a CL-GAN that verifies this conjecture, by training M (system parameter) latent vector components to become major components, and the rest, i.e., $m = \lambda - M$, to become minor components, see Figure 6. Consistent with the above IPG, we select extreme values for the M major components, i.e., from the set $\{-1,1\}$, to maximize the effect of each component on the visual characteristics of the generated images. However, we select the values for each of the m minor components, uniformly random from $(-1,1)$.

Train for Human Distinguishability. We leverage the input latent vector to control the visual characteristics of the images generated by G-CEAL. While major components contribute to the image distinguishability thus the capacity of the CEAL VKFG, we leverage minor components to help G-CEAL generate realistic images and maintain other visual aspects of the image. To achieve this, in each adversarial training epoch, we train G-CEAL using 3 steps, to generate (1) visually distinguishable images when the values of individual major components in the latent vectors are changed (flipped between 1 and -1) and (2) visually indistinguishable images when the values for minor components are flipped (see § 7.2).

Although in each of the 3 steps we use a different set of latent vector pairs as input to G-CEAL, the objective functions for all the steps has the general form shown in Equation 3.

$$L(\theta_{G_{CEAL}}) = \alpha \times HPD_{loss} + G_{loss} \quad (3)$$

In each step, we implicitly use this equation as the loss function to train G-CEAL. In this equation, HPD_{loss} is the HPD loss that we define exclusively for the step. This loss

is an indicator of how different the generated images are, as perceived by a human. $\alpha \in \mathbb{R}$ is a weight that determines the contribution of HPD_{loss} to the overall loss value for the step. G_{loss} is the generator loss in the conventional GAN (i.e., $G_{loss} = -log(D_{CEAL}(G_{CEAL}(z)))$, where z is a sample latent vector). This loss is an indicator of how realistic and visually similar the generated images are, compared to the images in the real image dataset used for training D-CEAL.

We now describe the 3 training steps M and $m = \lambda - M$ are the number of major and minor components in the latent vector input to G-CEAL.

• **Step 1: Train major components**. This step encourages specific (i.e., the first M) latent vector components to have a visual effect on the generated images. For this, generate M random seed latent vectors. Then, for each index $i \in \{1,2,3,...,M\}$, use the IPG(i) of Definition 1, along with the corresponding generated seed latent vector, to generate two random latent vectors v_1 and v_2. Use G-CEAL to generate images I_1 and I_2 from v_1 and v_2 respectively. Use the HPD classifier to compute $HPD_{predict}(I_1, I_2)$. To force the i^{th} component of the latent vector to be a major component, i.e., maximize the effect of the i^{th} component on the visual characteristics of the generated images, we want the HPD classifier to classify all these image pairs (I_1, I_2) as different (class 1). To achieve this, define the HPD_{loss} for the pair of latent vectors to be $HPD_{loss}(v_1, v_2) = cross_entropy(1, HPD_{predict}(I_1, I_2))$.

• **Step 2: Train minor components**. This training step encourages minor components to have minimal impact on image distinguishability. For this, generate m random seed latent vectors. Then, for each minor position $i \in \{M+1, M+2, ..., \lambda\}$, form sample latent vector pairs v_1 and v_2 as in Definition 1. Use G-CEAL on v_1 and v_2 to generate images M_1 and M_2. To force the i^{th} component of the latent vector to be a minor component, we want the HPD classifier to classify (M_1, M_2) as identical (class 0). For this, define the HPD_{loss} for this pair to be: $HPD_{loss}(v_1, v_2) = cross_entropy(0, HPD_{predict}(M_1, M_2))$.

• **Step 3: Train for variety**. During preliminary experiments we observed that using only steps 1 and 2, can lead to training a G-CEAL that generates similar images from randomly picked latent vectors (see Figure 7). Step 3 addresses this problem, by encouraging any 2 major components to impose different effects on the visual characteristics of generated images. For this, generate several batches of random seed latent vectors. Then, for each seed latent vector, pick two random major components $i, j \in_R \{1,2,3,...,M\}$ and $i \neq j$. Copy seed latent vector v into two other latent vectors v_1 and v_2, then set $v_1[i] = 1$ and $v_2[j] = 1$. Let N_1 and N_2 be the images that are generated by G-CEAL from v_1 and v_2 respectively. Define the loss of the generator as: $HPD_{loss} = cross_entropy(1, HPD_{predict}(N_1, N_2))$.

Train for Realism. As described in § 7, we train G-CEAL to also generate realistic images. We do this because the human visual system was shown to be better at distinguishing changes in images when their content is more natural [40]. To achieve this, we train G-CEAL also using the output (i.e., real vs. fake) issued by the D-CEAL discriminator for the G-CEAL-generated images of the previous epoch. Then, in each epoch, D-CEAL is also trained, similar to a conventional DCGAN, to discriminate the synthetic images generated by G-CEAL in the above 3 training steps from real images of a training dataset. Subsequently, we train G-CEAL using the classification signal provided by D-CEAL, i.e., G_{loss} in Equation 3. This process encourages G-CEAL to generate (previously unseen) images, that look like the images in the training dataset of real images, and deceive D-CEAL to classify them as real images.

7.3 Input Mapper

The CEAL approach trains the above G-CEAL generator to be a weak V_w function (§ 3). In § 9.3, we detail our experimental process to find the G-CEAL's d value (defined in § 6). We now describe the input mapper module, that solves the second sub-problem of Section 3: convert input keys into codes that are at Hamming distance at least d from each other.

Specifically, as also illustrated in Figure 6, the input mapper takes as input a key (e.g., public key, shared key, Bitcoin address, IP address, domain name) and outputs a latent vector L of length λ, i.e., a code word at Hamming distance at least d from the code word of any other input key. For this, the input mapper first computes a cryptographic hash of the input to produce K, its binary key fingerprint, of length γ. It then uses K to generate both the major and minor components of an output latent vector L as follows.

Generate the major components. To generate the major components of the latent vector L, we use an error correcting encoder ECC (see § 5 and § 6). First, we compute $ECC(K)$, then perform a one-to-one mapping between its bits and the major components of L: $L[i]$ = -1 if $ECC(K)[i] = 0$ and $L[i]$ = 1 if $ECC(K)[i] = 1$, $i = \{1,...,M\}$. If $|ECC(K)| < M$, we set $L[i]$ = -1 for the remaining $M - |ECC(K)|$ major components.

Generate the minor components. We use a PRNG to randomly generate the $m = \lambda - M$ minor components of L: $L(i) \in \mathbb{R}$ and $L(i) \in U(-1,1)$, $i \in \{M+1,...,\lambda\}$.

8 Data

To train HPD and CEAL, we use several datasets of real and synthetically generated images, described in the following.

Real Outdoor Image Dataset. We used a subset of 150,113 outdoor landscape images (mountains, ocean, forest) of 64 by 64 pixels, from the MIT Places205 dataset [38, 55] to train discriminator networks (vanilla DCGAN and CL-GAN models) to differentiate between real and synthetic images.

Ground Truth Human Perception Dataset. We used the following process to collect human-assigned labels to image pairs, which we use to train the HPD network. First, we

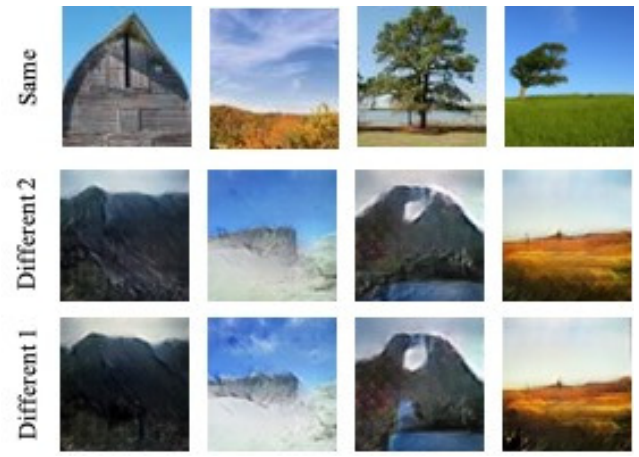


Figure 7: (top) Example real images that when shown as duplicates, were identified as different by 10% of participants in labeling study 1. (middle & bottom) Example different image pairs, generated with an early CEAL, that were identified as being identical by more than 15% of participants. This motivates the training step 3 of CEAL (§ 7.2).

trained a DCGAN network with a random uniform input latent vector of length $\lambda = 100$, using the above real outdoor image dataset. We stopped training the network when we started to observe realistic images similar to the ones in the training dataset (i.e., after 10 epochs). We refer to this trained network as the *vanilla DCGAN*. We then used the vanilla DCGAN to generate two datasets of synthetic image pairs (see below) and collected their labels using Amazon Mechanical Turk (MTurk) workers. GANs tend to generate similar images for close input vectors. Thus, since our objective is to collect labeled data to be used for identifying the boundaries of human visual distinguishability, we generate image pairs from key fingerprints that are only different in 1 component.

We followed an IRB-approved protocol to recruit 500 adult workers located in the US, to label 558 unique image pairs. We asked each worker to label each image pair as being of either "identical" or "different" images and paid them 1 cent per image comparison. We performed two image labeling studies. In the first study, we asked participants to label 35 image pairs, and in the second study 50 image pairs (see below). In each study, we randomized the order of the image pairs shown. We showed 1 image pair per screen. Both images had the same size (64x64) and resolution; we showed one image on the top left of the screen, the other on the bottom right. Across the two studies, 318 image pairs were labeled as different and 240 pairs were labeled as identical. In the following, we detail the two labeling studies that generated this dataset.

• **Labeling Study 1**. We used the vanilla DCGAN network to generate 100 synthetic different image pairs using 100 random seed latent vectors (v) and the IPG of Definition 1 for $i \in \{1,2,3,...,100\}$. Further, we generated a set of 40 identical image pairs consisting of duplicates of landscape images that we collected using Google's image search. We used proportional sampling to divide the total of 140 image pairs (100 different, 40 identical) into 4 groups of size 35 (25 assumed different, 10 assumed identical). We then recruited 400 MTurk workers and asked each to label one of the 4 groups, such that each image pair was labeled by 100 workers.

To weed out inattentive workers, we included an attention test (a.k.a. golden task [32]) at the beginning of the study. We did not collect the labels from workers who failed to answer the attention test correctly. In addition, we removed responses of "speeders" [21], i.e., workers who completed the study in less than one minute. We also removed the answers from workers who made more than 10 errors with respect to the assumed labels for the image pairs they processed. In total, we have removed the responses of 34 of the 400 workers, leaving us with labels from at least 94 workers for each image pair.

We then assigned to an image pair, an "identical" or "different" label, only if more that 90% of the worker responses concurred. 75 image pairs were labeled as different, and 65 were labeled as identical. Figure 7 (top) shows samples of identical image pairs that were labeled as different by about 10% of workers. Figure 7 (middle and bottom) shows samples of different image pairs that were labeled as identical by more than 15% of workers.

We studied the quality of responses collected from workers. 35 workers used a mobile device (smartphone or tablet) to work on our image comparison tasks. A Kruskal-Wallis test, did not show a significant difference between the number of errors made (w.r.t. the hypothetical labels) by participants in our studies using either of devices, i.e., desktop, laptop, mobile phone, or tablet, to complete the study (P-value = 0.93). We did not observe significant difference between the overall time it took for the participants using different devices to complete the studies (P-value = 0.06).

• **Labeling Study 2**. At the completion of the above study, we identified the index of components in the input latent vector whose corresponding generated images were labeled with the highest error rates by workers. We then performed a second labeling study to determine if the high error rate we observed was due to the fact that an observed "faulty" component always produces indistinguishable image pairs when its value is flipped, or this is due to other factors, e.g. the contribution of other components on the generated image.

First, for each of 3 image pairs with the highest error rate in labeling study 1, we generated 99 variant image pairs as follows: Let j be the index of the component that we flipped to generate this particular image pair in study 1 (which resulted in a high error rate). Also, let v be the seed latent vector (see Definition 1) corresponding to this image pair. For all $i \in \{1,2,3,...,100\}$ index values, where $i \neq j$, we used the IPG of Definition 1 to obtain two copies of v that only differ in the i-th component, then used the vanilla DCGAN to obtain an assumed "different" image pair. In total, we generated 297 (99×3) image pairs that are hypothetically different.

Dataset Name	# pairs	Similarity
Ground Truth Human Perception	557	Mixed
Unrealistic DCGAN Image Pairs	11,072	Same
Minor Change in Latent Vector	7,040	Same
Blob Image Pair Dataset	2,108	Different
10%-different Image Pair Dataset	1,024	Different
Enhanced Synthetic Image Pair Dataset	5000	Different

Table 2: Size of 6 generated image pair datasets, of either "identical", "different" or "mixed" image pairs, used to train the HPD classifier.

Second, for an additional set of randomly selected 7 components, plus the 3 "faulty" components above we generated 10 image pairs using a new seed latent vector randomly. We created two copies of the new seed latent vector and set the values of the j^{th} components to 1 and -1 in the first and second copy respectively. Thus, in total, we generated 100 image pairs. Further, we used another set of 49 hypothetically identical pairs, which we used to enrich our training dataset.

We have then collected labels for these 446 image pairs ($99 \times 3 + 100 + 49$) using 100 workers who labeled 50 image pairs each. As before, we eliminated the answers provided by speeders, those who failed the attention check, or made more than 10 errors with respect to the hypothetical labels of image pairs. In total, we removed responses from 13 workers. Then, for each image pair, we assigned it the assumed "different" or "identical" label, only if more than 80% of the workers agreed with it. Otherwise, we assigned the opposite of the hypothetical label as the true label of the image pair. 243 images were labeled as different; 203 image pairs were labeled as identical.

The Spearman correlation test did not reveal any significant monotonic correlation between the error rate for components in study 1, and image pairs corresponding to these components, in both experiments. This suggests that the components generating high error rates in study 1 alone, are not at fault. Therefore, we conjecture that the visual characteristics of a generated image are determined by a combination of effects of each component in the latent vector.

8.1 HPD Classifier Dataset

In order to train the HPD, we have generated 6 different datasets of synthetic image pairs, containing a total of 26,802 image pairs. Table 2 lists these datasets and their corresponding number of image pairs. One of these datasets is the above ground truth human perception dataset. In the following, we describe each of the other 5 datasets.

Unrealistic DCGAN Image Pairs. In order to train the HPD to correctly classify visually similar, but random noise images, as "identical", we generated an unrealistic image dataset of 11,072 image pairs using a poorly trained vanilla DCGAN: (1) 10,048 image pairs using a vanilla DCGAN trained for only 1400 iterations, i.e., less than an epoch, and (2) 1,024 image pairs using the same vanilla DCGAN trained for 3600 iterations (slightly more than an epoch).

We generated each of these image pairs as follows: randomly generate a latent vector, then select a random component and set its value to 1 once and -1 the other time. Convert the latent vectors to images using the poorly trained vanilla DCGAN, then label each pair as "identical". That is, we wish to train the HPD classifier to classify these image pairs as being identical, as this is how a human verifier will see them (gray images with random noise).

Minor Change in Latent Vector. We also generated synthetic "identical" image pairs, as follows. First, choose a random seed latent vector and use it to generate one image of the pair. Second, choose a random component of the seed latent vector uniformly, multiply its value by a small factor $c \in [0,1]$, then generate the other image in the pair. We generated 1024 image pairs with $c = 0.5$, 3008 pairs with $c = 0.6$ and 3008 pairs with $c = 0.7$, for a total of 7,040 image pairs. We randomly sampled 100 image pairs and 2 authors manually verified that they look identical.

Blob Image Pair Dataset. First, we generated 20 different blobs of random shapes and colors. Then, we generated 1,000 realistic images using the fully trained vanilla DCGAN model, using random input latent vectors. We formed image pairs that consist of (1) one synthetic image and (2) the same image, overlayed with one randomly chosen blob. We only accept the composite image (2) if its dominant color is dissimilar in the blob overlap position, to the color of the blob. To measure the similarity between colors we used the Delta E CIE 2000 [47] score, representing colors that are perceived to be different by humans [45]. We used the composite image if the score exceeded 50. In total, we generated 2,108 "blob" image pairs.

10%-different Image Pair Dataset. We generated 1,024 different image pairs, each as follows: generate a random seed latent vector, copy it to v_1 and v_2, select 10 random latent components (out of 100) uniformly and set the values of these components to 1 in v_1 and -1 in v_2. We then used the trained vanilla DCGAN to generate the corresponding image pair. Thus, these 1,024 image pairs are generated from latent vectors that are different in 10% of the components. We set this percentage experimentally: we generated 500 image pairs using input vector pairs that differ in $x \in [2,20]$ percent of their components, then manually compared them for visual equality. We found 10% to be the smallest percentage of difference that resulted in always distinguishable image pairs.

Enhanced Synthetic Image Pair Dataset. We generated 5,000 different image pairs as follows. For each of 1,000 random, vanilla DCGAN-generated images, we generated 5 images, by applying 5 enhancements, to change (1) image brightness, (2) contrast, (3) color, (4) add noise to the image, and (5) apply a blur filter to the image. We experimented with multiple parameters for each enhancement function and

Network	Hyper-parameters			labeled synthetic dataset				Unrealistic DCGAN image pairs (itr 1400)			Unrealistic DCGAN image pairs (itr 3600)			All other synthetic datasets		
	m	w	r	F1	FPR	FNR	Precision	F1	FPR	FNR	F1	FPR	FNR	F1	FPR	FNR
Siamese_model	1.64	0.49	0.02	0.72	0.20	0.35	0.82	-	0.06	-	-	0.32	-	0.77	0.01	0.35
HPD_model_1	-	1.57	0.24	0.82	0.24	0.21	0.84	-	0.15	-	-	0.47	-	0.83	0.02	0.29
HPD_attacker	-	2.97	0.13	0.72	0.20	0.37	0.84	-	0.05	-	-	0.31	-	0.77	0.0008	0.36

Table 3: Performance of the best HPD classifier that we trained and used to train CEAL (HPD_model_1) and the HPD model used by the attacker (HPD_attacker) and their underlying Siamese-like network, over different HPD classifier datasets.

selected them so that the generated image pairs (the original image and its enhanced version) are visually distinguishable.

9 Implementation

We have built CEAL in Python using Tensorflow 1.3.0 [6]. In this section, we describe the process we used to identify the parameters for which CEAL performs best.

9.1 HPD Training and Parameter Choice

Inception.v1 Layer Choice. We experimented with using activations of different layers of the Inception.v1, for HPD's image feature extraction effort (see 7.1). We performed 3 experiments, where we used activations from either the (1) "Mixed_5c", (2) "MaxPool_5a_2x2" or (3) "MaxPool_4a_3x3" layers of the Inception.v1. In each run, we used an identical architecture and initial weights of the fully connected layers weights in HPD. We trained the 3 networks for 1000 epochs. We repeated this process 200 times.

We compared the performance of these classifiers, using a paired t-test. We found a significant difference between the performance (over holdout datasets) of HPD classifiers trained using the "Mixed_5c" layer features, compared to the other two layers (P-Value = 0.000 when compared to "MaxPool_4a_3x3" layer, and P-Value = 0.000, when compared to "MaxPool_5a_2x2" features). In the following, we implicitly use the features extracted based on the activations of the "Mixed_5c" layer. The length of the activations vector for this layer is 50,176.

Training the HPD. We used the 6 datasets of § 8.1 to train and evaluate HPD. Specifically, we randomly split each synthetic dataset (except the ground truth human perception set), into training (80% of samples) and holdout (20%) sets: we used the training sets to train the HPD classifier, then tested its performance over the holdout sets. For the ground truth human perception dataset, we made sure that the number of image pairs that are labeled as identical and different, were distributed to training and test sets proportionally to their size.

We hyper-tuned the architecture and parameters of the HPD classifier to find a classifier which accurately identifies samples from the "different" class (has high precision). Such a classifier is necessary when training the CEAL to ensure G-CEAL stays away from generating images that are not human-distinguishable. Among the classifiers that we have trained with high precision, we chose the one with the highest F1. Figure 5 shows the best performing architecture for the HPD network. We refer to this network as HPD_model_1. Table 3 shows the performance of the Siamese network and the HPD networks that we trained and used in this paper.

In addition, we also trained an HPD model that has the same weights as HPD_model_1 in the Siamese layers, but different weights in the fully connected layer on top of the twin networks in the HPD architecture. This network, referred to as HPD_attacker, has a higher recall (lower FPR) when identifying the samples from the "identical" (negative) class on the holdout datasets. Therefore, this classifier would be preferred by an attacker, to identify potentially successful attack samples for a target CEAL image.

We note that the high FNR of our HPD models is mostly a problem for the adversary. This is because the FNR measures the ratio of the image pairs that HPD mistakenly detects to be identical. A high FNR means that the HPD is conservative: it will incorrectly identify attack image pairs, that are in fact perceived to be different by a real user. Thus, an HPD with a high FNR imposes either a lower success rate for an adversary, or more overhead on the adversary, who will have to manually verify attack images returned by the HPD.

9.2 CL-GAN Training and Parameter Choice

We trained CL-GAN using the above HPD models. Further, we also experimented using CL-GAN variants with different architectures (e.g., different number of neurons in the first layer of G-CEAL, 8,192 and 16,384) and values for hyper parameters, including λ and the number of major and minor components.

We also performed a grid search in the parameters of the CL-GAN including (1) the input size ($\lambda \in 64, 128, 256, 512$), (2) the number of major and minor components ($\frac{\lambda}{2}$, and (3) the $\alpha \in [25, 75]$ with step size 5, in the loss functions of the CL-GAN generator (see Equation 3). For best performing parameters, we also tested with different weight initialization for the networks weights.

We trained the CL-GAN using the process described in § 7.2, for 5 epochs, with batch size 64, and the Adam optimizer [29] to minimize Equation 3 for each step. We completed an epoch when all the images in the outdoor image

dataset were shown to the discriminator. To make the training process more stable, we trained the generator 3 times for every time we train the discriminator, but using the same inputs.

We observed that, when α is increased, the HPD_{loss} decreases faster (see Equation 3). However, the quality of the images is reduced for large values of α. We also observed that it is harder to train networks with larger values of λs: the quality of images generated by CEAL and their distinguishability decreases as we increase λ. We also observed that when the size of the nodes in the first layer of G-CEAL is increased, it generates smoother or lower quality (blurred) images.

We also experimented with the number of times that the generator network is trained using the three steps described in § 7.2, in each training epoch of G-CEAL. We observed that when the minor components are trained using Step 2 twice, there is a better balance between G_{loss} and HPD_{loss} of the trained network. Therefore in the following, we implicitly train G-CEAL twice using Step 2.

We have manually evaluated the quality of the images generated by the networks we trained. We built two CL-GAN model. The parameters for the best performing network using HPD_model_1 (i.e. alpha-CL-GAN) are $\alpha = 35$, λ =512, and $M = m$ = 256. We also built and evaluated an earlier CEAL model (i.e. alpha-CL-GAN) using $\alpha = 40$, that has $\lambda = 256$ and $M = m = 128$. In the following, we describe the process to identify the input mapper parameters for both models.

9.3 Choice of Input Mapper Parameters

To determine the value of d, i.e., the minimum number of major components that need to be modified to achieve consistent human distinguishability (see § 6), we used the following procedure. For each possible value of $d \in \{1,..M = 256\}$, we generated 1 million random target keys. For each target key, we generated an attack key by randomly flipping (i.e., 1 vs. -1) the values of d major components, then generated the CEAL images corresponding to the (target, attack) pairs. We used HPD_model_1 and HPD_attacker to find pairs likely perceived as identical by humans.

We manually checked the distinguishability of the image pairs flagged by the HPD models and observed that when $d > 30$, the image pairs are consistently distinguishable. To validate our observation, we showed the images identified as identical by HPD models for d=39 and d=43 to human subjects. We selected these values to be conservative and also to ensure the availability of BCH codes with corresponding minimum Hamming distances. When d = 39, HPD_model_1 identified 17 image pairs as identical, and the HPD_attacker identified 194. When d = 43, HPD_model_1 identified 7 identical image pairs, while the HPD_attacker identified 124 image pairs. All the pairs identified by HPD_model_1 were also identified by the HPD_attacker.

We used the procedure of § 10.2 to label these pairs using 34 human workers. None of the image pairs were confirmed as being identical by human workers. Therefore, we conclude that despite training limitations and the presence of local optima, when enough number of major components (i.e., ≥ 39) are flipped, the generated images are human-distinguishable. Thus, in the remaining experiments we set d to 39.

To ensure that major components of input vector to G-CEAL are at least in d Hamming distance of each other, we use a BCH (n=255, k=123, t=19) encoder to transform a key of length $\gamma = 123$ into the values for major components. This ECC has a minimum Hamming distance of 39 bits that transforms a message of length 123 into a code word of length 255. Thus, the major components in the latent vector of any CEAL images are at least 39 Hamming distance apart.

Based on the above setting, CEAL accepts binary key fingerprints of length $\gamma = 123$ bits. Therefore, the maximum capacity of CEAL is 2^{123}, i.e., it can generate up to 2^{123} unique, distinguishable images to represent binary key fingerprints.

9.4 alpha-CEAL

In addition to the above CEAL model that uses a CL-GAN with λ =512, we have built and evaluated a preliminary model, named alpha-CEAL, that uses its own CL-GAN network, named alpha-CL-GAN, with parameters λ =256, and $M = m$ = 128 (vs. 256 in CL-GAN). We followed a similar process to the one described above, using $\alpha = 40$ to determine the best parameters for alpha_CL-GAN.

alpha-CL-GAN Input Mapper Parameters. To identify the minimum number of major components that need to be modified to achieve consistent human distinguishability for alpha-CL-GAN, we generated 1,000 image pairs by flipping (i.e., 1 vs. -1) 5, 10, 15, 20 and 30 randomly chosen major components in each of 200 latent vectors respectively. We then used a procedure of § 10.2, to label these images using 69 MTurk worker.

In these experiments, the smallest number of different major components for which participants labeled as different, all the generated alpha-CL-GAN samples, was d = 15. Therefore, for the Input Mapper module, we used a BCH(n=127, k=78, t=7), i.e., an ECC with minimum Hamming distance of 15 bits that transforms a message of length 78 into a code word of length 127. Thus, alpha-CEAL accepts binary key fingerprints of length $\gamma = 78$ bits.

10 Empirical Evaluation

In this section, we evaluate the CEAL system with parameters identified in § 9.2 and § 9.3. First, we present the memory and computation overhead of the CEAL model (§ 10.1). We then describe the procedure we employed to run the user studies that we used for evaluation (§ 10.2). We evaluate the resilience of CEAL against the adversary described in § 3.1 (§ 10.3). We investigate Vash [1], identified as a state-of-the-art VKFG in terms of usability and attack detection [50], and

report vulnerabilities that we identified (§ 10.4). We then compare CEAL against Vash, in terms of their capacity and human verification speed (§ 10.5).

10.1 CEAL Overhead

The size of the CEAL generator model is 66.7MB. To determine the overhead of CEAL in generating image fingerprints, we performed experiments using a MacBook laptop with a 2.2 GHz Intel Core i7 CPU and 16GB memory, and a desktop equipped with a GeForce GTX TITAN X GPU. Over 1 million images, the average time to generate a CEAL image on the laptop was 2.3s (SD = 0.1s), while on the desktop it was 0.3s (SD=0.005s). This reveals that even without a GPU, CEAL can efficiently generate image fingerprints.

10.2 User Study Procedure

We followed an IRB-approved protocol to recruit MTurk workers to evaluate the performance of CEAL and Vash [1]. Specifically, we have recruited a total of 374 adult , US-based participants, 132 female and 242 male, with an age range of 18 to 64 (M=35.01, SD=9.23). 90.23% of our participants had college education or higher. 50%, 49% and 1% participants used a desktop, laptop and mobile device in our studies, respectively.

We asked the participants to compare a total of 3,496 image pairs: 1,918 CL-GAN-generated image pairs (219 workers), 1,308 alpha-CL-GAN-generated image pairs (100 workers) and 270 Vash-generated image pairs (55 workers).

We informed each participant on the purpose of the image comparison tasks, explaining their relationship to a security check that requires their full attention. We showed 1 image pair per screen, both with the same size (64x64) and resolution. One image was shown on the top left of the screen, the other on the bottom right.

We conducted several studies, each needing labels for a different number of image pairs (e.g., the CEAL resilience to attacks, or the study comparing CEAL and Vash, see below). For flexibility reasons, we asked participants in these studies to compare between 35 to 50 pairs of images. However, in each study, all participants labeled the same number of image pairs. We paid participants 10 cent per image pair comparison.

To avoid collecting low quality labels from inattentive workers, we have included an attention test at the beginning of the surveys and did not collect the labels from workers who failed to answer this question correctly. Further, we have included 5 attention check questions in each study: 3 obviously different pairs of images, and 2 pairs of identical (duplicated) images. In order to keep the type of images shown to the user consistent throughout these studies, we selected attention check questions using the same visual fingerprint generator that were used to generate the other images in the study: For CL-GAN, we generated obviously different attention check image pairs from a random seed latent vector where we flipped (1 vs. -1) a random set of 200 major components (out of 256). For Vash, we generated obviously different attention check image pairs randomly. We manually verified that all these image pairs look indeed different. We randomly selected the image pairs and the order in which they were shown, for each participant, however, we did not mix Vash an CEAL images in any experiment. We removed the answers from 12 participants who had incorrectly answered 3 or more (out of 5) attention check questions in the study.

Overall, for each image pair, we collected annotations from at least 3 workers. In the Vash studies of § 10.4, we collected at least 10 labels for each image pair. We used majority voting [32] to aggregate the labels assigned by the workers to each image pair, and produce the final human-assigned label.

Attack Dataset	Attack dataset size	# attacks found by HPD_attacker	Verified attacks
$(123,1)$-adversary	123M	121	2 (1.7%)
$(123,d)$-adversary	123M	1,473	23 (1.6%)

Table 4: Attack image datasets generated to break CEAL. We show the dataset size, the portion of the (target, attack) samples that were identified by HPD_attacker and the number of attack images validated by human workers.

10.3 Resilience of CEAL to Preimage Attacks

We evaluate CEAL under preimage attacks perpetrated by the adversary defined in § 3.1.

Resilience to $(\gamma,1)$-Attack. We first consider a powerful, $(\gamma,1)$-adversary, who can find input keys that are within 1-Hamming distance of the victim's, and uses them to generate attack CEAL images. Specifically, for γ=123, we generated 1 million target inputs randomly. Then, for each such input, we considered all γ "attack" strings that are within 1-Hamming distance of target, and used CEAL to generate images corresponding to the target and attack strings. Thus, in total we generated 123 million CEAL image pairs.

To avoid the high cost and delay of using humans to compare these image pairs, we used the HPD_model_1 and HPD_attacker to determine if they would be perceived as being identical by a human verifier. Out of 1 million target CEAL images, 121 of them were broken (only) once according to the HPD_attacker (see Table 4 top). Only 1 of these images was also identified by HPD_model_1. We then presented the 121 candidates to human verifiers (see § 10.2). Only 2 (1.65%) of these images were labeled as being identical by the recruited workers. The success rate of this $(\gamma,1)$ attack can thus be either interpreted as 2×10^{-4}% (2 target inputs broken out of the 1,000,000 attempted), or 1.62×10^{-6}% (2 successes out of 123 million trials).

Resilience to (γ,d)-Attack. Second, we consider the more general, (γ,d)-adversary (§ 3.1), where $1\leq d\leq\gamma$, γ=123.

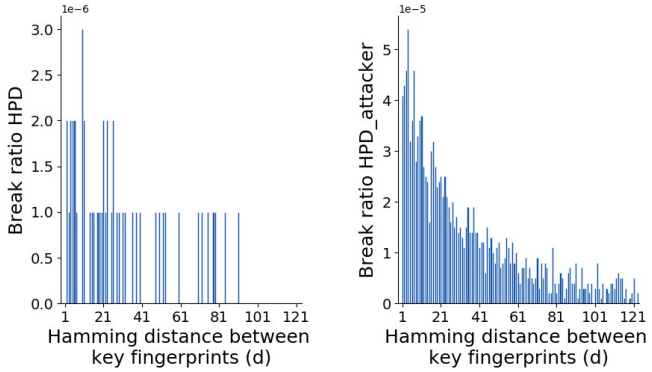


Figure 8: $(\gamma = 123, d)$**-adversary**: The break ratio of 1 million target CEAL images for each value of d, the Hamming distance between the attack and the target binary fingerprints, according to (left) HPD_model_1 and (right) HPD_attacker.

The goal is to see if such an adversary can find successful attack images generated from attack keys more different than the target. Specifically, for each value of $d \in \{1,2,3,...,\gamma\}$, we have built an attack dataset as follows: We generated 1 million random "target" inputs, and, for each target input, we randomly selected an "attack" string that is within Hamming distance d from the target. We then generated the CEAL images corresponding to each target and attack strings pair. Thus, in total we generated 123 million CEAL image pairs, organized into 123 datasets, each containing 1 million (target, attack) image pairs.

We have used both the HPD_attacker and the HPD_model_-1 to compare the 123 million (target, attack) image pairs that we generated. The HPD_attacker predicted 1,473 of the image pairs to be indistinguishable. HPD_model_1 only identified a subset of these samples (48 samples) as similar. When we presented the 1,473 candidate image pairs to human workers (§ 10.2), 23 (1.6%) of them were confirmed to be identical, one of which were identified by both HPD models (see Table 4 bottom). This suggests a success rate of $1.86 \times 10^{-5}\%$ (23 successes out of the 123 million trials) for this (γ, d) attack, performed by an adversary equipped with our HPD classifiers.

Figure 8 shows the portion of broken CEAL-generated images in each of the 123 datasets of (γ, d)-Attack (see § 10.3) according to (left) HPD_model_1 and (right) HPD_attacker. As expected, the number of broken CEAL images decreases as the Hamming distance between the target and attack binary key fingerprints increases.

We note that the KFG evaluation performed by Tan et al. [50], assumed an adversary able to perform 2^{60} brute force attempts, which is similar to the effort required to control 122 of 123 bits of the input hash key, required by a $(\gamma, 1)$-attack. Under such an adversary, Tan et al. [50] report a false accept rate of 12% for Vash [1] and 10% for the OpenSSH Visual Host Key [34]). This is significantly larger than CEAL's $2 \times 10^{-4}\%$ for the $(123, 1)$ attack and $1.86 \times 10^{-5}\%$ for the $(123, d)$ attack or even the 1.7% human error rate we observed in our user studies.

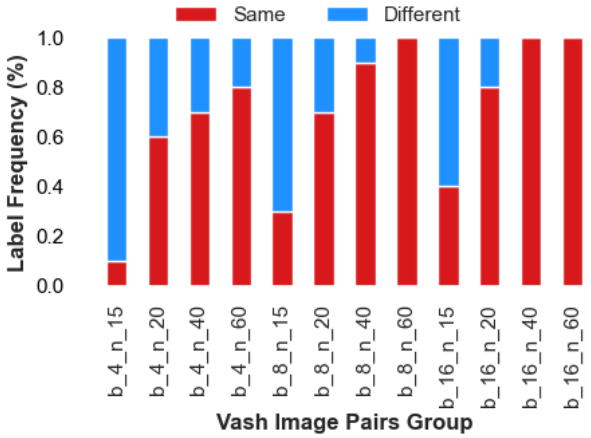


Figure 9: Distribution of "different" and "identical" labels as annotated by human workers for Vash image pairs. The number of image pairs that are identified as identical decreases as the number of buckets (b) and number of nodes (n) in the tree are decreased.

alpha-CL-GAN under $(\gamma, 1)$ and (γ, d) Attacks. We now report the performance of a $(\gamma, 1)$ and (γ, d)-adversary when breaking alpha-CEAL with γ of 78. Similar to (γ, d) attack performed for CL-GAN, we generate $\gamma = 78$ million pairs of (target, attack) samples. We observe that only a 295 images were broken according to HPD_model_1. In addition, we launched a $(\gamma, 1)$ attack on alpha-CL-GAN. Particularly, we generate 1M target keys. For each target key, we consider all 78 attack keys that are in 1-Hamming distance of the target key. We used the HPD_model_1 to decide if the generated image pairs would be perceived as being the same by a human verifier. This model identified 13 image pairs as identical.

We used the procedure described in §10.2 to label the 295 pairs of images that were identified by HPD_model_1 for (γ, d) attack and 13 pairs of images identified for $(\gamma, 1)$ attack using 31 MTurk workers. Our workers identified 3 of 295 and none of the 13 images to be identical.

10.4 Human-Distinguishability of Vash

Vash [1] is an open source implementation for random art [41], that converts input binary strings into structured, visual fingerprints. Vash seeds a PRNG with the input string, then uses the PRNG to construct an expression tree structure. The number of nodes of the tree, N, is chosen randomly using the PRNG. Vash then converts this tree to an image: Each node in the tree corresponds to an *operation* function, which modifies pixel values of the image. Each operation is chosen randomly (using the PRNG), from an existing pool of 17 operations, e.g., ellipse, flower, "squircle", etc. The operation parameters (e.g., the two foci of an ellipse or the size of a flower) are chosen randomly using the PRNG.

To study the ability of Vash [1] to generate human-distinguishable images, we generated 120 different Vash im-

age pairs, as follows. We first quantized the random values used to select each Vash tree operation, into 32 buckets, and quantized the operation parameter values into b buckets ($b \in \{4,8,16\}$) of the same size. We then generated random trees until we had 30 trees of each size $N \in \{15,20,40,60\}$, and corralled these trees into groups of 10. For each tree, we selected a random node (i.e., operation) and changed the value of one of its parameters by each of $q \in \{0.25,0.125,0.0625\}$, i.e., for each value of b. When selecting the operations, we made sure that each operation type appears in almost the same number of trees in each group. We generated thus 10 image pairs for each of the 12 combinations of q and n.

We used the procedure of Section 10.2 to label these pairs using 40 human workers. Each image pair was labeled by 10 workers. Figure 9 shows the portion of image pairs in each category that were labeled as either identical or different. We observe that human workers were able to consistently label image pairs correctly as different, only when the number of nodes N in the tree was 15, and the number of quantization buckets was 4 (i.e., a parameter needed to be changed by at least 0.25). Thus, Vash images are human-distinguishable only when the generating tree is small. However, when we generated 10,000 random Vash images (see the experiment in § 10.5), 99.98% of them were constructed from trees of more than 15 nodes. This suggests that most of Vash-generated images are vulnerable to attack.

10.5 CEAL vs. Vash

We compared CEAL and Vash [1] in terms of their capacity and the human speed of verification. Tan et al. [50] compare multiple text and visual KFG solutions, including Vash, though against a weaker adversary. Our results for Vash are consistent with those reported by Tan et al. [50].

For our comparison, we have generated 10,000 images randomly (from random keys) using CEAL, and 10,000 images using Vash. Then, separately for these datasets, we used the HPD_model_1 to predict if all pairwise images are human-distinguishable, i.e., using a total of 49,995,000 comparisons per dataset.

Since HPD_model_1 was not trained on Vash images, we sought to estimate its performance on Vash images. For this, we evaluated HPD_model_1 on the 120 Vash image pairs and their human-assigned labels, from § 10.4. HPD_model_1 achieved a FAR of 0.21, FNR of 0.14 and F1 of 0.76. Thus, HPD_model_1 achieves decent performance on Vash images, even though it was trained on dramatically different images (of nature not geometric shapes).

To estimate the number of distinguishable images for CEAL and Vash, we use the formula $\hat{k}(N_r,r) = \frac{N_r{}^2}{2r}$, where N_r is the number of samples until observing r repetition, i.e., human indistinguishable images (see [37]). This method provides only a lower bound estimate for the capacity of a visual key fingerprint generator function, as any estimation method fails when $k >> s^2$, where k is the real population size and s is the sample size used for the estimate.

Key fingerprint representation	Attack dataset size	# attacks found by HPD_model_1	Verified attacks
CEAL	∼50M	0	0 (0%)
VASH	∼50M	150	24 (16%)

Table 5: Attack datasets generated using 10K random images for each key fingerprint representation and the result of user study to label identified attacks by HPD_model_1.

Among the almost 50 million Vash image pairs compared, HPD_model_1 labeled 150 (3×10^{-4}%) pairs as being human-indistinguishable. In contrast, for the same number of CEAL image pairs, HPD_model_1 did not label any pair as human-indistinguishable. To build confidence that we did not miss relevant images, we also used the more conservative HPD_attacker model, to identify potentially indistinguishable CEAL image pairs. We found 6 such pairs.

We then used the human workers and process described in § 10.2 to confirm these 150 Vash and 6 CEAL image pairs, i.e., with each image pair being labeled by 3 humans. 24 of the 150 (16%) Vash image pairs were confirmed as being identical by the workers. Therefore, we estimate the number of perceptually different images generated by Vash as $\hat{k}(N_r,r) = \frac{10K^2}{2\times 24} = 2^{20.99}$. This result is consistent with the findings of Hsiao et al. [24]. We note that the 24 collisions occurred among 10,000 images chosen at random, and not images engineered for an attack. Section 10.4 shows that an adversary can engineer a collision for 99.98% of these images.

In contrast, none of the 6 CEAL image pair predicted to be perceived as identical by HPD_attacker, was found identical by the workers (see Table 5). Thus, we found no indistinguishable pairs among the 10,000 CEAL images.

Human Comparison Time. We studied the response times of human participants when asked to compare the above 150 Vash image pairs and 48 CEAL image pairs of § 10.3, i.e., identified as potential attacks by HPD_model_1. We measured the comparison time to be the interval between the moment when the image pair is shown to the worker, and the moment when the worker selected the response (different vs. identical). The workers were not allowed to change their selection. The average comparison time over Vash attack images was 3.03s (M=1.4s, SD=5.42s), and for CEAL it was 2.73s (M=1.83s, SD=2.33s).

11 Discussion and Limitations

Increasing Entropy. To increase the entropy of the CEAL key fingerprint generator, one could design and train multiple generators (see § 9.2), then use the input key to decide which generator to use (e.g., the value of the key's first two bits to pick one out of 4 generators). However, this approach imposes an exponential increase on computation and storage:

to achieve k bits of entropy, we need to train and access 2^k generators. Instead, in the proposed CEAL approach, we use careful training to achieve its entropy.

Improving HPD. Due to false positives, our evaluation is bound to have missed attack images. This is hard to avoid, given the need to evaluate millions of image pairs, a task that is infeasible with humans. We note that when modeling attacker capabilities, we used an HPD with higher recall than the one used to train the generator. This suggests that an adversary who has a better HPD, thus a higher chance of identifying potentially successful attacks, does not have substantial advantage against a CEAL image generator trained with a simpler HPD (though we cannot generalize this result).

However, both the attacker and the adopters of CEAL have the incentive to build a powerful HPD classifier. The attackers seek to find key fingerprint images most likely to be confused by users. The organizations adopting CEAL to protect their users would like to train a CEAL generator that uses the most of the key bandwidth available through human perception and the image generation algorithm. Thus, we expect adoption of CEAL (e.g., the CAPTCHA application of § 4) would increase interest in research of models for the limits of human visual perception.

Generalization of Results. The results of our studies do not generalize to the entire population, as we performed them on only a subset of Mechanical Turk workers. Such workers are generally better educated and more tech saavy than the broader population [26], thus are not representative of the entire population. For instance, we conjecture that workers who work on visualization tasks are less likely than the general population, to suffer from vision problems. Mechanical Turk workers also have different goals (minimize their time investment, maximize financial gains) which may differ from those of regular key fingerprint based authentication users, i.e., not only minimize time investment, but also correctly detect attacks. However, Redmiles et al. [43] have shown that in terms of security and privacy experiences, Mechanical Turk worker responses are more representative of the U.S. population than responses from a census-representative panel.

Our experiments are not equivalent to a lab study, since we do not know the circumstances or experience of the annotators. In addition, since a majority of the image pairs that we show to participants are attack images (i.e., believed to be visually similar), our studies also differ from the simulated attack format of Tan et al. [50], where the attack images form a small minority. The evaluation of Tan et al. is indeed more suitable for evaluating how humans react to attack images, in realistic settings. However, the goal of our experiments was different: we needed humans to validate image pairs predicted by HPD to be successful attacks. For instance, in one experiment we had to "human-validate" 1,473 image pairs. While our experiments are seemingly biased toward labeling image pairs as attack images, in this particular experiment, only 23 (1.6%) of the 1,473 image pairs were confirmed to be successful attacks. Further studies are needed to evaluate CEAL under realistic attack conditions such as the one proposed by Tan et al. [50].

In addition, we have trained CEAL to only generate nature landscape images. Our results do not generalize to other types of images.

More experiments are needed to verify that results of comparisons are consistent in scenarios where key fingerprints are displayed on devices with different screen properties (e.g., size and resolution), or even when printed on paper, to be compared against an image shown on a screen. Our experiments showed no difference between the responses from users comparing the key fingerprint on different devices. However, an extensive study is required to properly evaluate this aspect.

Finally, we have explored only (γ, d) attacks, for various values of d, for an adversary equipped with the HPD networks that we have developed. Future endeavors may investigate other types of attacks, including e.g., ones that attempt to find collisions for input latent vectors that are not similar.

Resistance to Adversarial Machine Learning. An attacker who has gained access to the CEAL network weights can leverage adversarial machine learning (e.g. gradient based) techniques to infer the input string from a target output CEAL image. While this problem is outside the scope of this work (e.g., CEAL images are often computed from input strings whose values are public) we note that in cases where this input is sensitive, one can apply CEAL to a hash of the input. This would force the adversary to further invert the hash to recover sensitive information.

12 Conclusions

In this paper, we proposed and have built the first human perception discriminator, a classifier able to predict whether humans will perceive input images as identical or different. We have used HPD to introduce CEAL, a new approach to train visual key fingerprint generation solutions, that provide input-distribution properties. We have shown that a CEAL-trained VKFG is substantially superior to state-of-the-art solutions, in terms of entropy, human accuracy and speed of evaluation.

References

[1] The Vash: Visually pleasing and distinct abstract art, generated uniquely for any input data. https://github.com/thevash, 2014.

[2] White paper: Whatsapp encryption overview. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf, 2016.

[3] Viber Encryption Overview. https://www.viber.com/security-overview, Last accessed, 2019.

[4] WhatsApp: End-to-end encryption. https://faq.whatsapp.com/en/general/28030015?lang=en, Last accessed, 2019.

[5] WP_MonsterID. http://scott.sherrillmix.com/blog/blogger/wp_monsterid/, Last accessed, 2019.

[6] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

[7] akwizgran. Encode random bitstrings as pseudo-random poems. https://github.com/akwizgran/basic-english, Last accessed 2019.

[8] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

[9] Jon Callas, Lutz Donnerhacke, Hal Finney, and Rodney Thayer. Openpgp message format. Technical report, RFC 2440, November, 1998.

[10] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2610–2620. 2018.

[11] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2180–2188, 2016.

[12] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[13] Kate Conger. TechCrunch: Facebook Messenger adds end-to-end encryption in a bid to become your primary messaging app. https://tinyurl.com/uetk9b5, 2016.

[14] Sergej Dechand, Dominik Schürmann, Karoline Busse, Yasemin Acar, Sascha Fahl, and Matthew Smith. An empirical study of textual key-fingerprint representations. In *Proceedings of the USENIX Security Symposium*, pages 193–208, 2016.

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

[16] Chris Donahue, Akshay Balsubramani, Julian McAuley, and Zachary C. Lipton. Semantically decomposing the latent spaces of generative adversarial networks. In *Proceedings of the International Conference on Learning Representations*, 2018.

[17] Carl Ellison and Steve Dohrmann. Public-key support for group collaboration. *ACM Transactions on Information and System Security (TISSEC)*, 6(4):547–565, 2003.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. 2014.

[19] Michael T Goodrich, Michael Sirivianos, John Solis, Gene Tsudik, and Ersin Uzun. Loud and clear: Human-verifiable authentication based on audio. In *Proceedings of the EEE International Conference on Distributed Computing Systems*, 2006.

[20] Will Grathwohl and Aaron Wilson. Disentangling space and time in video with hierarchical variational auto-encoders. *arXiv preprint arXiv:1612.04440*, 2016.

[21] Robert Greszki, Marco Meyer, and Harald Schoen. Exploring the effects of removing "too fast" responses and respondents from web surveys. *Public Opinion Quarterly*, 79(2):471–503, 2015.

[22] Peter Gutmann. Do users verify ssh keys. *Login*, 36:35–36, 2011.

[23] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2(2):147–56, 1959.

[24] Hsu-Chun Hsiao, Yue-Hsun Lin, Ahren Studer, Cassandra Studer, King-Hang Wang, Hiroaki Kikuchi, Adrian Perrig, Hung-Min Sun, and Bo-Yin Yang. A study of user-friendly hash comparison schemes. In *Proceedings of the Annual Computer Security Applications Conference*, pages 105–114, 2009.

[25] Huima. The bubble babble binary data encoding. https://tinyurl.com/phra64b, 2000.

[26] Ruogu Kang, Stephanie Brown, Laura Dabbish, and Sara Kiesler. Privacy attitudes of mechanical turk workers and the us public. In *Proceedings of the 10th Symposium On Usable Privacy and Security*, pages 37–49, 2014.

[27] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*, 2018.

[28] Hyunjik Kim and Andriy Mnih. Disentangling by Factorising. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2654–2663, 2018.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR 2015)*, 2015.

[30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *preprint arXiv:1312.6114*, 2013.

[31] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational Inference of Disentangled Latent Concepts from Unlabeled Observations. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[32] G. Li, J. Wang, Y. Zheng, and M. Franklin. Crowdsourced data management: A survey. In *Proceedings of the IEEE 33rd International Conference on Data Engineering*, pages 39–40, 2017.

[33] Yue-Hsun Lin, Ahren Studer, Yao-Hsin Chen, Hsu-Chun Hsiao, Li-Hsiang Kuo, Jonathan M McCune, King-Hang Wang, Maxwell Krohn, Adrian Perrig, Bo-Yin Yang, et al. Spate: small-group pki-less authenticated trust establishment. *IEEE Transactions on Mobile Computing*, 9(12):1666–1681, 2010.

[34] Dirk Loss, Tobias Limmer, and Alexander von Gernler. The drunken bishop: An analysis of the openssh fingerprint visualization algorithm, 2009.

[35] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *Proceedings of the Intl. Conference on Machine Learning*, pages 3481–3490, 2018.

[36] Maina M. Olembo, Timo Kilian, Simon Stockhardt, Andreas Hülsing, and Melanie Volkamer. Developing and testing a visual hash scheme. In *Proceedings of the European Information Security Multi-Conference (EISMC*, pages 91–100, 2013.

[37] A. Orlitsky, N. P. Santhanam, and K. Viswanathan. Population estimation with performance guarantees. In *Proceedings of the IEEE International Symposium on Information Theory*, pages 2026–2030, 2007.

[38] Outdoor 64 image dataset. https://github.com/junyanz/iGAN/tree/master/train_dcgan.

[39] John Padgette. Guide to bluetooth security. *NIST Special Publication*, 800:121, 2017.

[40] C Alejandro Parraga, Tom Troscianko, and David J Tolhurst. The human visual system is optimised for processing the spatial information in natural visual images. *Current Biology*, 10(1):35–38, 2000.

[41] Adrian Perrig and Dawn Song. Hash visualization: A new technique to improve real-world security. In *Proceedings of the Intl. Workshop on Cryptographic Techniques and E-Commerce*, pages 131–138, 1999.

[42] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the Intl. Conference on Learning Representations*, 2016.

[43] Elissa M Redmiles, Sean Kross, and Michelle L Mazurek. How well do my results generalize? comparing security and privacy survey results from mturk, web, and telephone samples. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2019.

[44] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. 2016.

[45] Zachary Schuessler. Delta E 101. http://zschuessler.github.io/DeltaE/learn, 2011.

[46] Ling Shao, Fan Zhu, and Xuelong Li. Transfer learning for visual categorization: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 26(5), 2015.

[47] Gaurav Sharma, Wencheng Wu, and Edul N Dalal. The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*, 30(1), 2005.

[48] William Suberg. Report: 2.3 Million Bitcoin Addresses Targeted by Malware That 'Hijacks' Windows Clipboard. https://www.viber.com/security-overview, 2018.

[49] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[50] Joshua Tan, Lujo Bauer, Joseph Bonneau, Lorrie Faith Cranor, Jeremy Thomas, and Blase Ur. Can unicorns help users compare crypto key fingerprints? In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 3787–3798, 2017.

[51] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith. Sok: Secure messaging. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 232–249, 2015.

[52] Benjamin Dumke von der Ehe. go-unicornify overview, November 2017.

[53] Yangli Hector Yee and Anna Newman. A perceptual metric for production testing. In *Proceedings of the ACM SIGGRAPH 2004 Sketches*, 2004.

[54] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3320–3328, 2014.

[55] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 487–495, 2014.