3-18-2019

# Image-based Authentication

Mozhgan Azimpourkivi
*Florida International University*, mazim003@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

# IMAGE-BASED AUTHENTICATION

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Mozhgan Azimpourkivi

2019

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Mozhgan Azimpourkivi, and entitled Image-based Authentication, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Sundaraja Sitharama Iyengar

B. M. Golam Kibria

Christine Lisetti

Geoffrey Smith

Umut Topkara

Bogdan Carbunar, Major Professor

Date of Defense: March 18, 2019

The dissertation of Mozhgan Azimpourkivi is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2019

DEDICATION

This humble work is dedicated to my parents.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisors Dr. Bogdan Carbunar and Dr. Umut Topkara who never stopped offering me advice and encouragement, and without whom this road was a lonely one. Thank you for guiding me, placing opportunities in front of me, and showing me the paths that consistently resulted in my academic and personal development. Thank you for your dedication and integrity.

I would like to thank the rest of my dissertation committee members, Dr. Sundaraja S. Iyengar, Dr. Geoffrey Smith, Dr. Christine Lisetti, and Dr. B. M. Golam Kibria for their insightful comments, encouragements and supports.

I also would like to thank Kianoosh G. Boroojeni, Samira Pouyanfar and my colleagues in CaSPR Lab: Nestor Hernandez, Mahmudur Rahman, Ruben Recabarren, Sajedul Talukder and Mizanur Rahman for the great discussions we had and for their endless support.

Last but not least, I would like to thank Farhad Hemmati, Siavash Rastkar, Shahrzad Ghourchian, Reza Sheykhi, Zahra Nafar, Saman Biook, Reza Amini and Anooshe Tavakoli. You walked along side me and made my life more colorful during the past 5 years.

Above all, I would like to thank my parents and my sister, Parisa, for all the unconditional love and support they have given me throughout my life. Without them, I may have never gotten where I am today.

ABSTRACT OF THE DISSERTATION

IMAGE-BASED AUTHENTICATION

by

Mozhgan Azimpourkivi

Florida International University, 2019

Miami, Florida

Professor Bogdan Carbunar, Major Professor

Mobile and wearable devices are popular platforms for accessing online services. However, the small form factor of such devices, makes a secure and practical experience for user authentication, challenging. Further, online fraud that includes phishing attacks, has revealed the importance of conversely providing solutions for usable authentication of remote services to online users. In this thesis, we introduce image-based solutions for mutual authentication between a user and a remote service provider. First, we propose and develop Pixie, a two-factor, object-based authentication solution for camera-equipped mobile and wearable devices. We further design ai.lock, a system that reliably extracts from images, authentication credentials similar to biometrics.

Second, we introduce CEAL, a system to generate visual key fingerprint representations of arbitrary binary strings, to be used to visually authenticate online entities and their cryptographic keys. CEAL leverages deep learning to capture the target style and domain of training images, into a generator model from a large collection of sample images rather than hand curated as a collection of rules, hence provides a unique capacity for easy customizability. CEAL integrates a model of the visual discriminative ability of human perception, hence the resulting fingerprint image generator avoids mapping distinct keys to images which are not distinguishable by humans. Further, CEAL deterministically generates visually pleasing fingerprint im-

ages from an input vector where the vector components are designated to represent visual properties which are either readily perceptible to human eye, or imperceptible yet are necessary for accurately modeling the target image domain.

We show that image-based authentication using Pixie is usable and fast, while ai.lock extracts authentication credentials that exceed the entropy of biometrics. Further, we show that CEAL outperforms state-of-the-art solution in terms of efficiency, usability, and resilience to powerful adversarial attacks.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

CHAPTER 1

# INTRODUCTION

## 1.1 Motivation

Mobile and wearable devices are popular platforms for accessing sensitive online services such as e-mail, social networks and banking accounts. Traditional solutions for authentication equation are not always suitable for mobile and wearable devices. For instance, while the traditional text password or PIN are widely used, and can be securely stored, accessed and converted to cryptographic keys, the small form factor of mobile and wearable devices disqualifies them as being user-friendly. In addition, while fingerprint based protection is convenient, it requires specialized hardware that is not ubiquitous in all mobile devices.

Generally, a secure and practical experience for mobile device-based user authentication raises significant challenges: the small and limited form factor of these devices, especially for wearable devices (e.g., smartwatches [Sam17] and smartglasses [Vuz17]), complicates the input of the commonly used text-based passwords, while the memorability of these passwords poses a significant burden on users who access a multitude of services [WWB+05b]. Further, the form factor of mobile and wearable devices renders them easy targets for theft followed by attacks to recover the user's authentication credentials, keying information, or sensitive information.

Biometric authentication solutions seemingly address the issues with small form factor of mobile and wearable devices. Biometric authentication features, comprised of "something you are", provide a basis for sufficiently strong systems security, instant verification and convenience for users. Forecasts put the total biometrics software and hardware market revenue exceptions to a staggering $15 billion for 2025 [KW17], which is dominated by applications in consumer device au-

thentication, mobile banking, point of sale payments, cashpoints, and IT systems security. However, the cost-benefit analysis of biometrics does not include personal implications to users, who are the least prepared for the imminent negative outcomes, and are not often given equally convenient alternative authentication options. Users have at most 10 fingerprints for all of their accounts in their lifetime, yet unlike passwords and credit card numbers, biometrics cannot be reset and re-issued when compromised. Furthermore, biometrics such as fingerprint, face and gait cannot even be kept secret. More importantly, as surrendering biometrics may become de facto mandatory [MK16, Kee15], existing vulnerabilities [XPFM16, PLK+12, ACJP14, GRGB+12], coupled with the compromise of large scale biometrics databases [Pet15], raise significant long term security concerns, especially as transactions authenticated by biometrics across different systems are linkable and traceable back to the individual identity.

In addition, token-based authentication solutions, e.g., SecurID [RSA17], usually require extra hardware and expensive infrastructure [Sec15] (e.g. for issuing, managing, and synchronizing the token). In addition, use of these solutions can impact usability: the user needs to enter password plus an additional code to the device to authenticate.

A secret image-based authentication approach, where users authenticate using arbitrary images they capture with the device camera, may address several of the above problems. For instance, the authentication is not tied to a visual of the user's body, but that of a personal accessory, object, or scene. As illustrated in Figure 1.1, a user sets her *reference* credential to be an image of a nearby object or scene. To authenticate, the user captures a *candidate* image; the authentication succeeds only if the candidate image contains the same object or scene as the reference image. In this dissertation, we introduce Pixie and ai.lock (see Chapter 4 and Chapter 5), two

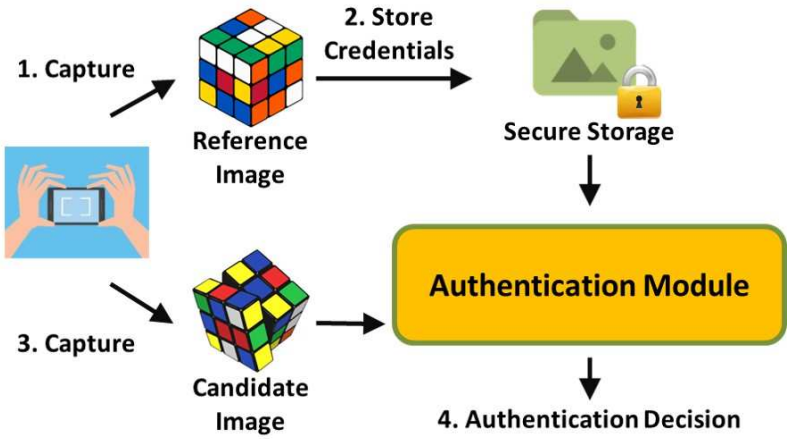



Figure 1.1: Image-based authentication scenario. The user captures the image of an object or scene with the device camera. The information about reference credentials are securely store on the mobile device or on a remote server. The user authenticates only if she can captures another image of the same object or scene.

secret image-based authentication solutions. These solutions improve on (1) biometrics, by freeing users from personal harm, providing plausible deniability, allowing multiple keys, and making revocation and change of secret simple and (2) token-based authentication, by eliminating the need for an expensive infrastructure. Visual token-based solutions (e.g., based on barcodes or QR codes) [MPR05, HPOH12] can be seen as special cases of secret image-based authentication.

Further, online fraud that includes phishing attacks [Ram10], has revealed the importance of conversely providing solutions that allow users to verify the authenticity of online services they access or the identity of other users with whom they communicate, e.g., social networking friends, e-mail contacts, etc. One approach that allows the users to perform such verification, without any reliance on trusted third parties with predefined authorities, is through the use of key fingerprints.

Key fingerprints [wae16, akw, GSS+06, Hui00, LLvG09, PS99] help humans compare arbitrary data strings (e.g., keys, addresses, and identifiers) for equality. Key fingerprints have a wide range of applications, including preventing phishing

[Ram10, RAM+18] and Bitcoin clipboard [Sub18] attacks, authentication in End-to-End Encrypted (E2EE) applications, device pairing and security indicators, see § 6.2.2 for more details.

Efficient Key Fingerprint Generation (KFG) solutions need to minimize the time taken by a human to compare the data, and minimize the success rate of "collision" attacks, where adversaries find different key fingerprints perceived as being the same by humans. Tan et al. [TBB+17] have shown that Visual Key Fingerprint Generation (VKFG) solutions and Vash [vas14] in particular, that convert input strings into images for humans to compare, outperform several text-based key fingerprint solutions (e.g. [wae16, Hui00]) in terms of both human attack detection rate and comparison time.

The attack success rate against visual key fingerprint representation solutions is however still unacceptable, exceeding 10% [TBB+17]. One reason for this is that existing solutions do not take into account the limits of human perception and how it relates to the space of images that they generate. More specifically, existing solutions cannot predict if specific changes in the input data will generate human-distinguishable changes in the generated images. Further, we show that most Vash [vas14]-generated images are vulnerable to attack (see § 6.7.4), and out of only 10,000 random-generated Vash images, we found 24 human-validated collisions.

Unlike existing solutions that only generate structured images (e.g. Vash [vas14], Unicorn [vdE17] and CLPS [OKS+13]), a straightforward approach to generate realistic and diverse set of images from input strings, is to use a Generative Adversarial Network (GAN [GPAM+14]). GANs can model the distribution of image samples, and can be used to draw previously unseen samples from the estimated distribution.

In this dissertation, we introduce, implement and evaluate CEAL, short for CrEdential Assurance Label, a human-centric VKFG solution that employs GANs [GPAM+14]

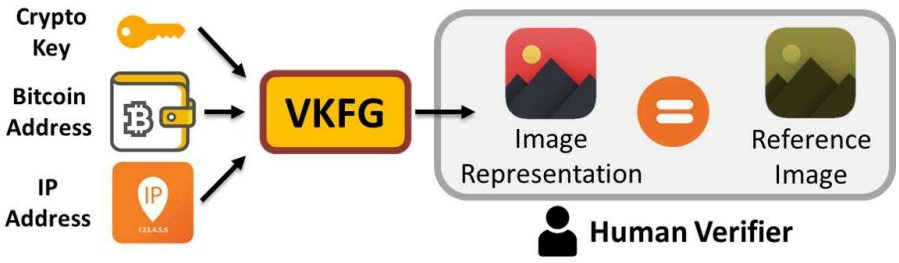



Figure 1.2: Visual key fingerprint-based authentication scenario: Given an arbitrary input string, the VKFG function generates an image fingerprint representation of the input. A human verifier compares this image against a securely acquired reference image fingerprint, e.g., from trusted site, person-to-person, etc.

to generate realistic, attack-resilient images (i.e. *ceals*), that are easy to compare by humans. As depicted in Figure 1.2, to authenticate an identity, the user needs to generate the image key fingerprint corresponding to the identity using VKFG. Then, he compares this image to a *reference* image of the key fingerprint that he has obtained through a secure channel. Unlike existing solutions that base their security on cryptographic hash functions or pseudo-random number generators, CEAL has built-in "input-spreading" properties that endow it with second pre-image and collision resistance properties that is required for key fingerprint generation solutions.

## 1.2 Contributions

In this dissertation, we design secure, efficient and usable solutions for mutual authentication between mobile device users and remote parties. Specifically, we propose three image-based authentication solutions. The first two approaches are designed for authenticating a mobile device user either locally, to a mobile device, or remotely to a remote service provider. Conversely, we present a visual key fingerprint representation solution to allow users to verify the authenticity of a remote identity, e.g., remote service provider, social network contact or a nearby device the

user is willing to connect. In the following, we briefly introduce these systems that include: Pixie, ai.lock and CEAL.

### 1.2.1 Secure image-based mobile authentication

We introduce Pixie and ai.lock, two secret image-based authentication methods for mobile and wearable devices. Pixie is an object-based authentication solution that employs traditional image processing techniques to extract image features (i.e., "keypoints") and match user captured images. Pixie has an important drawback when deployed on mobile devices: the image keypoints that it extracts need to be stored and matched in cleartext on vulnerable devices. Therefore, to eliminate this problem, we need to make sure that the image features are stored on a secure remote server. In contrast, ai.lock uses state of the art, Deep Neural Network (DNN) based image feature extraction (see § 3.1.4) along with locality sensitive hashing (see § 3.2) to extract binary *imageprints* that are robust to changes caused by image capture conditions. The ai.lock's imageprints can be securely stored and matched using secure sketches. This makes ai.lock resilient to device capture attacks. Furthermore, on larger and more complex attack datasets, the use of DNNs enabled ai.lock to achieve False Acceptance Rate (FAR) that are at least 2 orders of magnitude smaller than those of Pixie ($\leq 0.0015\%$ vs. $0.2 - 0.8\%$), for similar FRRs (4%).

In the following sections, we briefly describe the Pixie and ai.lock systems. In addition, we describe our major contributions when designing these systems.

#### Pixie

We introduce Pixie, an easy to use camera-based remote authentication solution for mobile and wearble devices, see [CaS17d] for a short demo. Pixie can establish trust

to a remote service-based on the user's ability to present to the camera a previously agreed secret physical token. We call this token, the *trinket*. We use the term trinket to signify the uniqueness and small size of the token, not its value.

Users choose their trinkets similar to setting a password, and authenticate by presenting the same trinket to the camera upon further authentication attempts. The fact that the object is the trinket, is secret to the user. Pixie extracts robust, novel features from trinket images, and leverages a supervised learning classifier to effectively address inconsistencies between images of the same trinket captured in different circumstances.

Pixie combines graphical password [BCVO12, Pas17, DMR04] and token-based authentication concepts [RSA17, VAS17], into a two factor authentication (2FA) solution based on what the user has (the trinket) and what the user knows (the trinket, the angle and section used to authenticate). Contrary to other token-based authentication methods, Pixie does not require expensive, uncommon hardware to act as the second factor; that duty is assigned to the physical trinket, and the mobile device in Pixie is the primary device through which the user authenticates. Pixie only requires the authentication device to have a camera, making authentication convenient even for wearable devices such as smartwatches and smartglasses.

Our major contributions are as follows:

- We introduce Pixie, a novel camera-based two factor authentication solution for mobile and wearable devices. Pixie leverages the ubiquitous cameras of mobile devices to snap images of trinkets carried by the users. A quick and familiar user action of snapping a photo is sufficient for Pixie to simultaneously perform a graphical password authentication and a physical token-based authentication, yet it does not require any expensive, uncommon hardware. Pixie establishes trust based on both the knowledge and possession of an arbi-

trary physical object readily accessible to the user, called *trinket*. Users choose their trinkets similar to setting a password, and authenticate by presenting the same trinket to the camera.

- We extract robust, novel features from trinket images, and leverage a supervised learning classifier to effectively address inconsistencies between images of the same trinket captured in different circumstances. Pixie classifier determines if a candidate image contains the same token (i.e. trinket) as a set of reference images.
- We develop several image-based attacks including brute force image pictionary attacks, a shoulder surfing flavor and master image attacks. We construct more than 14.3 million authentication instances and show that Pixie is resilient to these attacks.
- We manually collect a dataset of images that consist of 1400 images of 350 different objects. We only selected objects that are good candidate to be used as trinket. We have captured 4 images for each trinket, that differ in background and lighting conditions, i.e., either indoors using artificial light or outdoors in daylight conditions. We make our datasets including the Pixie attack datasets, available for download [CaS17c].
- We implement Pixie in Android and study the usability and discoverability of Pixie as a novel form of authentication. We show through a user study with 42 participants that Pixie is accurate, faster than text passwords and perceived as such by users. In addition, we show that Pixie's trinkets are memorable.
- We publish Pixie as an open source prototype, with code and the Android installation file available on GitHub [CaS17e] and Google Play Store [CaS17b]. Pixie work is published in proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT) in 2017 [ATC17a].

**ai.lock**

Image-based mobile authentication approach, as in Pixie, raises important challenge when deployed to locally authenticate a user to a mobile device (local authentication scenario): an adversary who captures or compromises the device that stores the user's reference credentials (e.g. mobile device) and has access to its storage, should not be able to learn information about the reference credentials or their features.

To address the problem of secure storage and matching of the image features, we introduce *ai.lock*, a practical, secure and efficient image-based authentication system that can be used for both local and remote authentication scenarios. Unlike Pixie, ai.lock uses state of the art, DNN-based image feature extraction (see 3.1.4) along with Locality Sensitive Hashing (LSH) [Cha02] to extract binary *imageprints* that are robust to changes in image capture conditions. We show that imageprints can be securely stored and matched using secure sketches [DRS04]. This makes ai.lock resilient to device capture attacks.

We measure the security of ai.lock against brute force attacks on more than 3.5 billion authentication instances built from more than 250,000 images of real objects, and 100,000 synthetically generated images using a GAN[RMC15] trained on object images. We show that the ai.lock estimated entropy is superior to a fingerprint-based authentication built into popular mobile devices.

Our major contributions are as follows:

- We introduce *ai.lock*, a practical, secure and efficient image-based authentication system that converts general mobile device captured images into biometric-like structures, to be used in conjunction with secure sketch constructs and provide secure authentication and storage of credentials.
- To extract invariant features for image-based authentication, ai.lock leverages (1) the ability of Deep Neural Networks (DNNs) to learn representations of

the input space (i.e., *embedding vectors* of images) that reflect the salient underlying explanatory factors of the data, (2) Principal Component Analysis (PCA) [F.R01] to identify more distinguishing components of the embedding vectors and (3) Locality Sensitive Hashing (LSH) [Cha02] to map the resulting components to binary space, while preserving similarity properties in the input space. We call the resulting binary values *imageprints*. ai.lock builds on a secure sketch variant [DRS04] to securely store reference imageprints and match them to candidate imageprints.

- We propose the LSH-inspired notion of *locality sensitive image mapping* functions ($\delta$-LSIM), that convert images to binary strings that preserve the "similarity" relationships of the input space, for a desired similarity definition (see § 5.3). A $\delta$-LSIM function can be used to efficiently match images-based on their extracted binary imageprints.
- We develop several image-based attacks including brute force image dictionary attacks using real images as well as 100,000 synthetically generated images using a GAN trained on object images. We vaccinate ai.lock to be more resistant to synthetic image attacks. We also develop a synthetic credential attack to brute force ai.lock. Finally, we introduce an object/scene guessing attack to evaluate ai.lock in a scenario similar to shoulder surfing attack. We show that ai.lock is resilient to attacks: Its FAR on 140 million synthetic image attack samples is $0.2 \times 10^{-6}\%$. ai.lock was unbreakable when tested with 1.4 billion synthetic credential attack samples. The estimated Shannon entropy [Sha01] of ai.lock on 2 billion image pairs is 18.02 bits, comparing favorably with state-of-the-art biometric solutions. Further, we show that ai.lock is a $\delta$-LSIM function, over publicly available image datasets as well as images we collected (see § 5.6.5).

- We implement ai.lock in Android using Tensorflow [ABC+16]. We have released the code and data on ai.lock Github [CaS17a]. We show that ai.lock is fast, imposing an overhead of under 1s on a Nexus 6P device. We have published ai.lock work in proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC) in 2017 [ATC17b].

### 1.2.2 Human-distinguishable Visual Key Fingerprint Generation(VKFG)

Although recent studies by Tann et al. [TBB+17] suggests that visual key fingeprints are more usable, the attack success rate against Visual Key Fingerprint Generation (VKFG) solutions is still unacceptable, exceeding 10% [TBB+17]. One reason for this is that existing solutions do not take into account the limits of human perception and how it relates to the space of images that they generate. More specifically, existing solutions cannot predict if specific changes in the input data will generate human-distinguishable images. Further, we show that most images generated by state-of-the art VKFG (i.e., Vash [vas14] an implementation of random art [PS99]) are vulnerable to attack (§ 6.7.4), and out of only 10,000 random-generated Vash images, we found 24 human-validated collisions.

In this dissertation, we introduce, implement and evaluate CEAL (CrEdential Assurance Labeling), a user-centric contact authentication system. The CEAL system generates unique, hard to spoof images (i.e., *ceals*) for each relationship between a user and a contact, and enables users to visually verify the authenticity of information received from contacts. Ceals can be embedded in social networks, online sellers, or bank accounts, in order to authenticate the owner of a ceal to the user.

Ceals protect users from man-in-the-middle attacks, where attackers impersonate the users to contact other users (e.g., their customers, friends, and contacts).

CEAL generates its images (i.e. ceals) using a Deep Generative Models (see § 3.1.5). Unlike existing solutions that base their security on cryptographic hash functions or pseudo-random number generators, CEAL has built-in "input-spreading" properties that endow it with second pre-image and collision resistance properties.

First, we reveal important vulnerabilities of existing VKFG solutions. We then perform large-scale brute-force attacks using more than 255M pairs of (target, attack) keys, that differ only in small number of bits, to break CEAL. We then identify potential successful attacks using a classifier that can predict if a pair of images are perceived as same or different by human. We then present these identified images along with their corresponding target images to Amazon Mechanical Turk (MTurk) worker to label them. We show that even under attack of such a powerful adversary, the break ratio of CEAL images is small ($< 1.0\%$).

We provide the following contributions:

- **Vulnerabilities in State-of-the-Art Solution**. We reveal important vulnerabilities of Vash [vas14], the leading visual key fingerprint representation solution. We show that even in a small sample of 10K Vash images 24 pairs of images were identified as same by our human subjects.
- **Human Perception Discriminator (HPD)**. Unlike [YN04], we build a classifier that predicts if two images (generated by a GAN) will be perceived as being the same or different by average humans. We show that despite sparse human annotated data, an HPD with high precision and low recall can be built and still be effectively used for CEAL. We also show that the HPD has good performance even when evaluated on radically different images than those in training (Vash images).

- **Hash-Like CEAL**. We develop the first VKFG solution with built-in hash properties (second pre-image and collision resistance). We train CEAL DCGAN, a Deep Convolutional Generative Adversarial Network (DCGAN) [RMC15], that is unlike the generic model (see § 3.1.5), is trained i) using HPD to generate not only realistic, but distinguishable images, ii) with constraints that ensure separation of latent vector components into major and minor components while maximizing the capacity of major components. We publish the data and code to build CEAL DCGAN and HPD networks.
- **Datasets**. We generated more than 500 synthetic image pairs and used crowdsourcing workers to label these image pairs as either same or different. In addition, we collected labels for more than 7000 ceal image pairs, including likely attack image pairs using human workers. Finally, we collected the labels for 270 vash image pairs that we generated. We publish these image datasets along with the collected labels.

## 1.3 Outline of Dissertation

This dissertation is organized as follows:

- In Chapter 2, we review and discuss the relevant work to authentication for mobile and wearable devices. We then review relevant literature to the problem of key fingerprint representation including existing solutions. We also, review traditional and more advance methods for extracting useful image features that can be employed for image comparison and matching.
- In Chapter 3, we review the required background relevant to this work including DNNs, image generation using DNNs, locality sensitive hashing, and error correcting codes.

- In Chapter 4, we describe the system and adversary model for image-based authentication scenario. We introduce Pixie, a novel image-based authentication solution. We evaluate its performance, including under attack, and study its usability. We discuss Pixie limitations and application that goes beyond mobile-device authentication.
- In Chapter 5, we describe the system and adversary model for image-based authentication scenario that is performed locally on the mobile device. We introduce ai.lock, an image-based authentication solution that extras biometric-like features from authentication images and use them for secure storage and matching of the images on the mobile device. We evaluate ai.lock performance under adversarial attacks. We discuss ai.lock limitations and application and finally conclude this study.
- In Chapter 6, we reveal important vulnerabilities of existing VKFG solutions. We then introduce CEAL, a visual key fingerprint generation solution that incorporates visual human perception into the key fingerprint generation process to guarantee the generated images are human distinguishable. We build CEAL using Tensorflow [ABC+16] and evaluate its performance under powerful attacks. We then describe the limitations and future directions to improve the key fingerprint images generated by CEAL.

CHAPTER 2

# RELATED WORK

In this chapter, we first present the relevant work to both image-based authentication and key fingerprint representation, the two major related fields to this dissertation. Particularity, we review existing solutions for authentication in mobile and wearable devices. Then, we describe traditional methods for image feature extraction and matching. We also review modern approaches using Deep Neural Networks (DNNs) for image feature extraction. Next, we review exciting key fingerprint representation methods. Finally, we review deep generative models for image generation and differentiate our work to existing solutions.

## 2.1 Mobile Authentication

The image-based client authentication solutions that we present in this dissertation (i.e., Pixie and ai.lock) are camera based authentication solutions that combine graphical password and token based authentication concepts into a single step 2 Factor Authentication (2FA) solution. Proposed image-based authentication solutions are based on what the user has (the particular trinket among all the other objects that the user readily has access to) and what the user knows (angle and viewpoint used to register the trinket). The unique form factor of image-based authentication solution differentiates it from existing solutions based on typed, drawn, or spoken secrets. We briefly survey and distinguish our image-based authentication solution from existing solutions.

### 2.1.1 Biometric-based Authentication

Biometric based mobile authentication solutions leverage unique human characteristics, e.g., faces [DLHvZH15], fingerprints [App17b], gait [JW15], to authenticate users. In particular, the form factor of Pixie and ai.lock (the image-based authentication solutions we present in this dissertation) makes them similar to camera based biometric authentication solutions based on face [BCF+13, TSK+12, DLHvZH15] and gaze [KAH+16, LDGW15]. Consequently, Pixie and ai.lock share several limitations with these solutions, that include (i) vulnerability to shoulder surfing attacks and (ii) susceptibility to inappropriate lighting conditions, that can spoil the performance and usability of the authentication mechanism [BUI+15, MB14].

In addition, previous studies [BUI+15, DLHvZH15] report that face biometrics adoption might be problematic as participants have expressed mixed feelings toward using them. For instance, participants in the De Luca et al. [DLHvZH15] user study expressed feeling awkward, as authentication can be perceived as taking a selfie in public. While we have not evaluated this dimension, we expect trinket shots to be perceived as less awkward than the appearance of taking selfies in public.

Another concern in biometric authentication solutions that requires a camera is to verify the liveness of the authentication secret to prevent spoofing attacks. In face based authentication, liveness can be verified by requiring the users to blink or move their mouth upon capturing the image of the face [KFB08a]. Boehm et al. [BCF+13] introduced a form of challenge-response liveness verification for gaze based authentication where the user gaze at and follows a moving icon on the screen. We note that liveness verification solutions, e.g. based on consistency between the device motions and motion directions inferred from images captured by the camera [MR16], can bring advantages of liveness verification to Pixie and ai.lock.

In contrast to biometrics, Pixie and ai.lock enable users to change the authenticating physical factor, as they change accessories they wear or carry. This reduces the risks from an adversary who has acquired the authentication secret from having lifelong consequences for the victims, thereby mitigating the need for biometric traceability and revocation [PPJ03].

In addition, due to the diversity of the trinkets, we need to solve a harder problem than existing biometrics based authentication solutions: while existing biometrics solutions focus on a single, well studied human characteristic, image-based authentication trinkets can be arbitrary objects or scenes, thus lack the convenience of a set of well known features.

We note that Pixie and ai.lock can be used in conjunction with biometric solutions as an additional authentication factor. For instance in touchscreen devices, we can use the touch gesture used to shoot the trinket, as an additional authentication factor [DLHB+12].

### Biometric Protection

Our work is also related to the problem of protecting biometric templates. We summarize biometric protection solutions, that can be classified into fuzzy biometric protection and feature transformation approaches [JNN08].

**Fuzzy biometric template protection**. This approach leverages error correcting codes to verify biometric data. Techniques include secure sketch and fuzzy extractor [DRS04], fuzzy vault [JS02] and fuzzy commitment [JW99], and have been applied to different biometric data, e.g. palm and hand [LS15].

In Chapter 5, we extend the secure sketch under the Hamming distance solution from [DRS04]: reconstruct the biometric credential, then compare its hash against a stored value. We briefly describe here the password set and authentication pro-

cedures for ai.lock, using generated *imageprints*, i.e., the authentication credential (see Chapter 5). Let $ECC$ be a binary error correcting code, with the corresponding decoding function $D$, and let $H$ be a cryptographic hash function.

• **Image password set**. Let $R$ be the reference image captured by the user and let $\pi_R = \pi(R)$ be its ai.lock computed imageprint. Generate a random vector $x$, then compute and store the authentication credentials, $SS(R, x) = \langle SS_1, SS_2 \rangle$, where $SS_1 = \pi_R \oplus ECC(x)$ and $SS_2 = H(x)$.

• **Images based authentication**. Let $C$ be the user captured candidate image, and let $\pi_C = \pi(C)$ be its ai.lock computed imageprint. Retrieve the stored $SS$ value and compute $x' = D(\pi_C \oplus SS_1)$. The authentication succeeds if $H(x') = SS_2$.

Dodis et al. [DRS04] further proposed the *fuzzy extractor* concept that extracts a uniformly random string $R$ from the protected biometric in an error-tolerant way. They then show how to construct this given a secure sketch. Fuzzy extractors can be used to securely encrypt the mobile data with strong keys extracted from biometrics with sufficient entropy. In Chapter 5, we show that on our experimental datasets, ai.lock's entropy is 16-18 bits, comparing favorably with state-of-the-art biometric authentication solutions (see Table 5.1).

**Transformation based biometric template protection**. A transformation is applied both to biometric template and biometric candidate, and the matching process is performed on the transformed data. In an invertible transformation (a.k.a., *salting* [JNN08]), a key, e.g., a password, is used as a parameter to define the transformation function [TGN06]. The security of this approach depends on the ability to protect the key. In contrast, in non-invertible schemes [MCF+10, RCCB07] a one-way transformation functions is used to protect the biometric template, making the inversion of a transformed template computationally hard even when the key is revealed.

**Hybrid approaches**. Hybrid transformation and fuzzy protection approaches have also been proposed. Nandakumar et al. [NNJ07] introduced an approach to make the fingerprint fuzzy value stronger using a password as salt. Song et al. [OJN08] used discrete hashing to transform the fingerprint biometric, which is then encoded and verified using error correcting codes.

### 2.1.2 Security Tokens and 2 Factor Authentication (2FA)

The trinket concept is similar to token based authentication, such as door keys and hardware security tokens [RSA17], as authentication involves access to a physical object. Hardware tokens are electronic devices that provide periodically changing one time passwords (OTP), which the user needs to manually enter to the authentication device. Software token solutions, such as Google's 2-step verification [Goo17], require the user to retrieve a verification code sent to the mobile device (e.g., through SMS) from the mobile device and type it into the authentication device. This further requires the device to be reachable from the server, hence introduces new challenges (e.g. location tracing, delays in phone network, and poor network coverage). Moreover, such solutions provide no protection when the device is stolen. They also impact usability, as the user needs to type both a password and a verification code.

Solutions such as [CDK+12, SJSN14, KMSv15] treat the mobile device as a second factor. Further, they eliminate user interaction to retrieve a token from the mobile device to the authentication device (e.g. a desktop) by leveraging proximity based connectivity (e.g., Bluetooth, Wi-Fi). For instance, in PhoneAuth [CDK+12], where the user authenticates from a browser to a remote server, the mobile device stores a private key of the user and uses it to sign server issued challenges. In Shirvanian et al. [SJSN14], the mobile device associates with an access point created

on the authentication device. In SoundProof [KMSv15], the second factor (proximity between the mobile and authentication devices), is ensured by verifying the consistency of the ambient noise captured by the two devices' microphone sensors.

Other approaches exist that seek to transform biometrics into tokens that the user needs to carry, with important implications on biometric privacy and revocation capabilities. For instance, TAPS [TAP17] is a glove sticker with a unique fingerprint intended for TouchID.

Token-based authentication requires an expensive infrastructure [Sec15] (e.g. for issuing, managing, synchronizing the token). Pixie and ai.lock provide mechanisms that make objects usable as passwords, with the existing infrastructure. They may also provide a personalized and inexpensive alternative to such tokens. In addition, as the user action of scanning a bar-code is replaced with snapping of a photo of a personal object image-based authentication can provide a faster alternative to visual token based authentication, especially when the trinket is readily accessible to the user, e.g., tattoo, piece of jewellery worn by the user, etc.

In addition, the mobile device or a specialized device can act as a secondary device for storing a token for the second authentication factor. By contrast, in image-based mobile authentication we assign the duty of storing the token for the second factor to a physical object outside the mobile device. The mobile device is the sole device that is used to authenticate to a back-end application or to access the services on remote servers. As an added benefit, the physical factor of the trinket renders image-based authentication solution immune to the "2FA synchronization vulnerabilities" introduced by Konoth et al. [KvdVB16], that exploit the ongoing integration of apps among multiple platforms.

### 2.1.3 Graphical Passwords

Pixie's and ai.lock's visual nature is similar to graphical passwords, that include recall and recognition systems. In the following, we briefly describe these solution, see [BCVO12] for a survey.

Recall based solutions such as DAS (Draw-A-Secret) [JMM+99] and variants [DY07, GGC+08] ask the user to enter their password using a stylus, mouse or finger. YAGP [GGC+08] extends DAS with approximate matches for input passwords. Pass-Go [TA08] uses a finer grained grid than DAS and restricts the user movement to grid lines and intersections, eliminating small shape variations. GrIDsure [Gem] displays digits in a 5×5 grid and requires the user to memorize a pattern within this grid. On login, the user is displayed with a grid permutation and needs to input the digits shown within the pattern. DAS variants are popular on smartphones, including Android and Blackberry (PatternLock). However, Zhao et al. [ZAH15] propose an attack framework that is able to crack a significant portion of picture gesture passwords.

Recognition-based systems (e.g., Passfaces [Pas17, DP00]) exploit the human ability to better recognize previously seen images. To create a password, users select and memorize a set of images (e.g., faces), which they need to recognize from among other images during the authentication process. Davis et al. [DMR04] have shown that when the users are in charge of choosing their own images, such schemes may be insecure, e.g., passwords may be correlated with the race or gender of the user. Similarly, GeoPass [TMSA13] is a graphical location based authentication schema which requires the users to choose and remember a location on digital maps as their password. Passfaces [Pas17, BS00, DNO08], the most popular recognition-based system, uses images of human faces. Awase-E [TK03] allows users to use their own images, improving flexibility and memorability, and increasing the attack search

space. Similar to Passfaces, Awase-E imposes several verification steps, where the user needs to correctly select the reference images from the decoys in each step. Story [DMR04] extends the Passface approach with a "sequentiality" requirement: users need to identify their reference images in the correct order. Weinshall [Wei06] extends the basic scheme with a keyboard based navigation of displayed images, to prevent spyware and shoulder-surfing attacks.

Cued-recall systems improve password memorability by requiring users to remember and target specific locations within an image [WWB+05a, WWB+05c]. PassPoints [WWB+05a, WWB+05c], one of the first solutions, also includes the sequentiality requirement: it presents the user with an image, on which he needs to select a sequence of key points. To authenticate, the user is shown the cue (the image) and needs to identify the key points, in the correct sequence. Suo [Suo06] introduced a shoulder-surfing resistant extension that blurs random areas of the image and asks the user to decide if a key point is within the clear area.

Pixie and ai.lock (see Chapters 4 and 5) can be considered to be a recognition-based graphical password systems with dynamically generated images, where the physical world around the user represents the set of possible passwords. Since the user freely presents the candidate password through a photo of the physical world, captured in different light, background, and angle conditions, we need to implement an accurate matching of trinkets. Trinkets can be small portions of items worn by users (e.g., shirt pattern, shoe section). Thus, even if the attacker is able to see and reproduce the trinket, the attacker does not know the required section and angle of the trinket. These image-based authentication solutions accurately verify that the candidate image contains the same trinket part as a previously captured reference image(s). This process endows image-based authentication with attack resilience properties: to fraudulently authenticate, an adversary needs to capture both the

mobile device and the trinket, then guess the correct part of the trinket or angle of a scene that was used when registering the image password.

### 2.1.4 Text-based Passwords

Traditional solutions to authenticate users on a mobile phones are based either on entering a Personal Identification Number (PIN) or a password. The usability of traditional text-based passwords has been well studied in literature, see e.g., [TSK+12, MKS+16, CFS+09, SKD+14]. Several limitations are associated with text passwords on memorability and usability especially when adopted in mobile platforms.

Melicher et al. [MKS+16] found that creating and entering passwords on mobile devices take longer than desktops and laptops. In mobile devices, text-based passwords need to be entered on spatially limited keyboards on which typing a single character may require multiple touches [SWKW13], due also to typing the wrong key. Cherapau et al. [CMAB15] identified memorability and less typing as reasons that users choose PINs rather than longer and more secure passwords (or passcodes) to protect their mobile devices. Shay et al. [SKD+14] have shown through a large user study of different password-composition policies, that more than 20% of participants had problems recalling their password and 35% of the users reported that remembering a password is difficult. Trewin et al. [TSK+12] found that face biometrics can be entered faster than text based passwords. Shay et al. [SKD+14] reported that user entry time for text passwords ranges between 11.6-16.2s (see Table 4.1) in line with our evaluation (see § 4.6.2).

Image-based authentication solution replaces the user action of typing a password with pointing the camera to the trinket and snapping a photo of it. In Chapter 4, we

show that Pixie is perceived as more memorable than text passwords (see § 4.6.2). We also show that Pixie entry time is faster than text based passwords on a mobile device.

### 2.1.5 Wearable Device Authentication

The special form factor and limited input method of wearable devices make the employment of conventional authentication methods, such as PIN, cumbersome for users. However, wearable devices are sometimes equipped with physiological (e.g., ECG, EEG) and kinesthetic sensors that open up a range of new possibilities for authentication solutions on these devices.

Not only the biometric information collected by wearable devices are used for user authentication [BCTPL16], wearable devices can be used as the second authentication factor: in iAuth [LL16] the smartwatch collects and sends the motion patterns of the user to authenticate to a smartphone. Wearable authentication solutions often leverage the available sensors to address the small form factor of the devices: Yoon et al. [YPL15] use the light state changes captured by the ambient light sensor as a PIN entry method. WatchMe [VVBVS15] uses the smartwatch camera to process the input (e.g. PIN) drawn by the user on a canvas. Similarly, Pixie and ai.lock can be employed as a form of authentication in wearable devices that are equipped with a camera.

## 2.2 Image Feature Extraction and Matching

Traditional feature extraction (e.g. [RRKB11, BTVG06, LCS11, Low04]) are based on finding "keypoints" on the images. These keypoints can then be used to compare against similar keypoints on another image and are usually resistant to scale

or rotation [RRKB11, BTVG06]. Particularly, these algorithms extract several features (piece of information, a.k.a descriptor) about the pixel values surrounding the keypoints. Later, these descriptors can be used to identify the patch in the image when comparing it against other patches in another image. In Chapter 4, we use SURF (Speeded Up Robust Features) [BTVG06] and ORB [RRKB11] algorithms, to extract scale and rotation invariant image keypoints and use them for image matching.

In order to match two images, one can compare all keypoint in the image to all the other keypoints in the second image to find the closest keypoints based on a distance measure (bruteforce matching). However, this is expensive and time consuming. Instead in Chapter 4, we use Fast Library for Approximate Nearest Neighbors (FLANN)-based matcher [ML09]. These are algorithms that are developed for fast nearest neighbor search based on a specific distant measure.

More advanced methods for feature extraction includes using Convolutional Neural Networks (CNNs) [LB+95]. These networks take advantage of local connectivity to extract salient image features from images. The representations learned by CNNs are shown to resemble the primate visual system [CHY+14]. In addition, they can help predict human eye fixation [KWB17], and capture information about artistic style of images that are meaningful to humans [GEB16]. Specifically, we use the ability of CNNs to capture underlying images features and use these features to train different image classifiers (see § 5.4 and 6.5.3).

## 2.3 Key Fingerprint Representation

In this section, we review the relevant key fingerprint representation solutions. This is particularly, related to the CEAL system that we introduce in Chapter 6.

### 2.3.1 Text-based Key Fingerprints

Key fingerprint representation methods transform an input key (e.g. (hashed) public key) into a shorter, human readable format. The numeric [wae16] format uses numbers (0-9) to represent a binary key fingerprint in based 10 instead of 2. However, the most commonly used textual key fingerprint representations encode the key into a hexadecimal (letters A-F and numbers 0-9) or based32 (Latin alphabet) strings. In both numeric and alphanumeric representation, the string can be grouped into chunks of equal length. Chunking can slightly affect the user verification speed [DSB$^+$].

In addition to (alpha)numeric representation, the key fingerprints can be represented as a set of pronounceable words [Hui00]. In this method, the binary key fingerprint is splitted into chunks. Each chunk is then mapped to a word that is selected from a dictionary. Other teqniques use syntactically correct English sentences [akw, GSS$^+$06] to represent key fingerprints.

The text-based schema used for key fingerprint representation impacts the efficiency and accuracy of verification process [DSB$^+$, TBB$^+$17, VWO$^+$17]. Under an targeted attack scenario, an adversary may try to generate a key whose fingerprint differs in positions which are not apparent to an inattentive human verifier (e.g. middle of the string instead of its beginning or the end [DSB$^+$, TBB$^+$17]). Dechand et al. [DSB$^+$] evaluate the performance and usability of key fingerprint representation using (alpha)numeric strings, words and sentences. Based on their findings, wildly adopted hexadecimal representation performs poorly both from verification accuracy and usability aspects. In addition, while representing the key fingerprints using a large dictionary of words can increase the verification speed, representing keys using sentences achieved the best attack detection rate.

### 2.3.2 Graphical and image-based representations

Several techniques represent the key fingerprints visually, using graphics or synthetically generated images. Visual key fingerprint representation solutions,such as Random art [PS99] and OpenSSH Visual Host Key [LLvG09], use the key to generate a structured image.

For instance, Vash [vas14], is an open source implementation for Random art [PS99], that converts input binary strings into corresponding image "fingerprints". Vash uses the input string as the seed to a Pseudo Random Number Generator (PRNG), then uses the output of this PRNG to construct an expression tree structure. The number of nodes of the tree, which we denote by $N$, is chosen randomly by PRNG. Vash then converts this tree to an image fingerprint: Each node in the tree corresponds to an *operation* function, which modifies the pixel values of the fingerprint image. Each operation is chosen randomly using the PRNG, from an existing pool of 17 operations, e.g., ellipse, flower, squircle, etc. In addition, the parameters of an operation are chosen randomly in [0, 1]. However, this value can then be mapped to a numbers in a different range.

Avatar representation techniques have also been used as key fingerprint representation. For instance, in [vdE17] a unicorn image with randomly chosen properties is drawn based on the input key and a PRNG.

Further, the text-and-image hybrid solution and the most widely used solution, OpenSSH visual host key [LLvG09] splits the 128 bits MD5 hash of the input key, into chunks. Each chunk is then processed to decide how to navigate on canvas according to a set of rules and add specific ASCII characters to the canvas. WP-MonsterID [mon] randomly selects pieces of the visual representation from a dataset of images. However, this requires a large image dataset to ensure the uniqueness and distinguishability of key fingerprints.

Tan et al. [TBB+17] performed user studies to compare several textual and graphical key fingerprint representation solutions. Their findings suggest that visual key fingerprints can speed up the verification of key fingerprints. In their experiments, Vash [vas14] outperformed the unicorn solution [vdE17] and several text-based key fingerprint representation solutions (e.g. hexadecimal and numeric representations) in terms of both attack detection rate and comparison time.

In contrast, CEAL is the first visual key fingerprint representation solution designed to ensure that the human visual system can differentiate between image fingerprints generated from different keys. In Chapter 6, we show that despite its reliance on a PRNG, Vash is unable to satisfy the requirements that we introduce for visual key fingerprint representation (see 6.3.1). Particularly, we show that not all the images generated by Vash are human-distinguishable, especially when the number of overlaid shapes and colors on the canvas increases. This is expected, as the visual sensitivity of human to changes diminishes with increased spatial frequency [YN04]. Further, we show that CEAL outperforms Vash. Over a set of 10K random vash images, we identified 24 images indistinguishable, however, using the same procedure, we did not identify any indistinguishable images for CEAL.

## 2.4 Deep Generative Models for Image Generation

Deep Generative Models (DGMs) are DNNs that are usually trained, using unsupervised learning, to summarize explanatory 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 fake and unseen but realistic and plausible instances similar to the samples in the training dataset. There are two major classes of generative mod-

els: Variational AutoEncoder (VAE) [KW13] and Generative Adversarial Networks (GAN) [GPAM$^+$14].

In this dissertation, we focus on GANs for generating images (see § 3.1.5 for a review on the conventional components and training process of a GANs). Note, GANs are an active area of research and there has been many efforts to generate realistic and high resolution images using this networks (e.g. [KALL18]).

In Chapter 4 and 5, we use a traditional GAN to model an adversary who can use these networks to draw a large number of plausible image samples to break image-based authentication solutions that we introduce.

In addition, in Chapter 6, we introduce CEAL DCGAN (see § 6.4.1), a GAN network trained to generate realistic and human distinguishable images. To train this network, we modify the process of training GANs to generate images that are not only realistic but human distinguishable. CEAL DCGAN uses a network with similar architecture as in DCGAN [RMC15] to generate images. However, it can be replaced with any state-of-the-art GAN-like network (e.g. [KALL18]).

Similar to [CDH$^+$16, DBML18], we modify the training process of GAN to generate images with particular properties. InfoGAN [CDH$^+$16] uses unsupervised learning to distangle the visual characteristics (e.g. style, color, pose of objects, etc.) of the generated images by a GAN. Similar to CEAL, SDGAN [DBML18] uses a supervised approach to train a GAN with latent vectors that include components representing both identities (e.g. individual humans) and observations (e.g. specific photographs) of human faces.

In CEAL, we decompose the latent vector components (the input to a GAN) into major and minor components (see Conjecture 6.4.2). The major components learn to carry information about human distinguishability aspect of the generate images, while minor components represent other properties of the images (e.g. realism).

CHAPTER 3

# BACKGROUND

In this chapter, we present the necessary background relevant to this research. First, we review Deep Neural Networks(DNNs), including the generative models for generating images. We review the components of these networks and how these networks are trained.

We then present the background on locality sensitive hashing for transforming real valued vectors into binary strings while preserving the similarity relationships of the input vectors. In addition, we briefly review BCH [BRC60] an error correcting code that we use in this dissertation.

## 3.1 Deep Neural Networks (DNNs)

Deep learning [Sch15] is a branch of machine learning that attempt to model and learn high level abstractions in data. A Deep Neural Networks (DNN), is a network of connected processing units (neurons/nodes) that are wired in an specific fashion to perform a task. In the simplest case, a neural network can be defined by input, hidden and output neurons. An input neuron accepts a real value as input and passes a modified version of it, by using a transformation function, to the neuron(s) in the next layer. Each neuron is connected to other neurons in the next layer through a weight, which is a model parameter. In addition, there is a value, referred to as bias, associated with each neuron in a neural network. The neuron uses its connected weights and its bias to transform the input of the neuron into its output.

### 3.1.1 Deep Neural Network Models

Neural network models are usually used for function approximation. There are three major types of neural network models: (1) discriminative models, (2) generative models, and (3) hybrid models. Here, we briefly describe these models. Consider a pattern recognition problem where we want to identify the relationship between covariant input samples to a target variable y, where y could be discrete (e.g., as in classification problems) or continuous (e.g., as in regression problems).

#### Generative Models

A generative model learns the joint probability distribution of samples and their corresponding class labels, i.e., $p(x, y)$. Once this model is estimated, we can apply Bayes rule to indirectly compute the conditional probability of $y$ given the sample $x$, i.e., $p(y|x)$. In the case of a classification problem, a generative model learns the distribution of individual classes. In addition, generative models potentially learn the natural features and properties of data. As a result, these models can be used for drawing (unseen) samples from the same distribution as in original data. In practice, generative models are widely used for unsupervised feature learning.

#### Discriminative Models

Unlike generative models, discriminative models are directly trained to estimate the conditional probability of a target value given a sample, i.e., $p(y|x)$. Discriminative models learn the boundary of samples from different classes. Many of traditional classification algorithms, such as logistic regression and SVM, can be categorized as discriminative models.

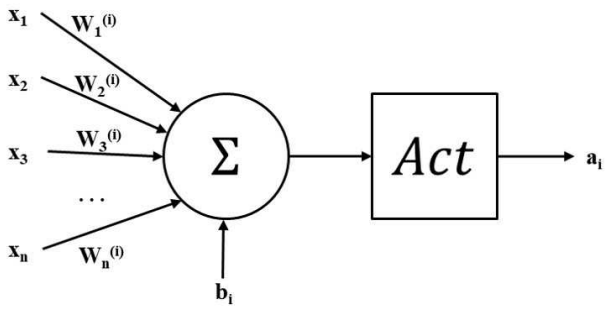



Figure 3.1: A neuron (with index i) in a feed forward neural network: the input to the neuron is scaled, summed, added to a bias and is passed to a non-linear activation function (Act) to produce the output.

#### Hybrid Models

Hybrid models combine the generative and discriminative models to perform a task. For instance, the output of a discriminative model can be used as a criteria to guarantee certain property of the samples generated by a discriminative model. Similarly, a generative model can assist with training a discriminative model by improving the optimization process or providing regularization for the model [Den12]. In CEAL (see Chapter 6), we use a discriminative model (i.e., Human Perception Discriminator) to push an image generator (a generative model) to generate only images that are human distinguishable.

### 3.1.2 Layers in Deep Neural Networks

Independent of the DNN type, i.e., discriminative, generative or hybrid, the architecture of a neural network consist of connected neurons that are stacked on top of each others in different layers. There are several types of layers in a neural network. Here, we briefly introduce some of these layers that are usually used in practice and the networks we have designed or used in the dissertation.

**Fully Connected Layer**

In a feed forward neural network with fully connected layers, the connection between the neurons of different layers do not form a cycle: the nodes in each layer are connected to all the other nodes in the next layer, while, there is no connection between the neurons in one layer. In a forward pass, the information flows in only one direction from input layer to optionally intermediate (hidden) layers and finally the output layer. A single neuron accept as input a vector $x$ with $n$ components (i.e., x is a n-dimensional vector) and produces a single output. The output of the neuron is computed using $h = W^T x + b$. In this equation, $W$ is the weights connecting the neuron to the input or the neurons in the previous layer, and $b$ is neuron's bias. This bias can increase the computational capability of the neuron. However, the relation between the input and output of a layer is linear, hence, a neuron acts only as a linear transformation function. In order to increase the computational capability of a neuron, a non-linear function is applied to the output of each node. This non-linear function is referred to as the "activation function". Therefore, the transformation of input to the output is performed through $h_{out} = Act(h)$ where Act is an activation function. In § 3.1.2, we review several activation functions that we have used in the networks designed or used in this dissertation.

The notation above can be extended to a network with a layer that has multiple neurons. Let $m$ be the number of neurons in this layer. Each node is connected to the nodes/input components from the previous layer through a set of weights i.e., $\{W^{(1)}, W^{(2)}, W^{(3)}, ..., W^{(m)}\}$. Also, for these nodes we have biases $\{b_1, b_2, b_3, ..., b_m\}$ respectively. Then, the activation of $i^{th}$ node in this layer is computed as $a_i = Act({W^{(i)}}^T x) + b_i$, where ${W^{(i)}}^T$ is the transpose of the $i^{th}$'s nodes weights. Figure 3.1 depicts this process. To summarize, in a fully connected layer, each neuron computes the dot product between the input and its weights, adds the bias to

the dot product value and optionally applies a non-linearity (i.e., activation function) to the result obtained in previous step.

### Convolutional Layer

Convolutional layer derives its name from the convolution operator in mathematics. A convolutional layer is organized in 3-dimensions: width, height, and depth. In addition, to provide weight sharing and local connectivity, the neurons in each layer are connected only to a small region of the previous layer. The spatial span of this connectivity is referred to as the "receptive field" of a neuron. The size of the receptive field along depth axis is always equal to the depth of the input. The set of weights that are used to transform an input region to output are referred to as a "filter" (a.k.a. kernel). The output of a convolutional layer is computed by sliding the filter from the top left corner of the input to the bottom left corner, $s$ step at a time. $s$ is a hyperparameter and is referred to as "stride". This process is similar to the convolution operation. To perform the convolution operation, the input is sometimes padded with zeros around the boarders to simplify calculations. Based on the size of the input (I), receptive field of a filter (f), stride (s) and the size of padding (p), we can compute the size of the output of a convolutional layer as $(I - f + 2p)/(s + 1)$.

As an example, consider an image as input to a convolutional layer. Let the size of the image be $m \times m \times d$ where m is the width and height of image and d is the number of channels in the image, e.g., an RGB image has $d = 3$. The convolutional layer itself can have $k$ filters of size $f \times f \times q$ where $f < m$ and $q = d$. In this case, each neuron in the convolutional layer has weights that are connected to a $[f \times f \times d]$ region in the input, for a total of $f \times f \times d$ weights plus

one bias parameter. With stride 1 and no padding, the output of such layer has size $(m - f + 1) \times (m - f + 1) \times (k)$.

### Pooling Layer

Commonly, convolutional layers are followed by a pooling layer. The purpose of this layer is to reduce the spatial dimensionality of the intermediate representations and the number of parameters of the network. Consequently, overfitting can be controlled by reducing the number of parameters.

Similar to a convolutional layer, a pooling layer has filters of small size, however, each pooling layer filter is applied independently on each slice of the input on the depth axis. The dimensionality of the representations are reduced by applying either a minimum or maximum function, or by taking the average over the receptive field of a filter. Depending on the function used, the pooling layer is referred to as a min pooling, max poling or average pooling layer.

### Activation Function

In order to increase the computational capacity of the neural networks, a non-linear function is applied to the output of the layers in an element wise fashion. Here, we review the activation functions we use in this dissertation.

**Sigmoid.** The sigmoid (a.k.a logistic) activation function accepts a real-valued number as input and squashes it into range of (0,1). The mathematical form of this function is $\sigma(x) = \frac{1}{1+e^{-x}}$.

**Softmax.** The softmax function is the generalized version of logistic function that squashes a k-dimensional vector of real values into a k-dimensional normalized probability values: all the output values add up to 1. The mathematical form of softmax function for the $j^{th}$ element of input is $\varphi(x)_j = \frac{e^{x_j}}{\sum_{i=1}^{k} e^{x_i}}$. In this equation, $x$ is a

k dimensional input and $1 \leq j \leq k$. Softmax function is usually used in case of multi-class classification problems where it computes normalized probabilities over $k$ different mutually exclusive classes. In contrast, sigmoid function is used in case of binary classification problem (i.e., when k = 2).

**Tanh.** Tanh activation function provides a zero-centered activation by squashing the neuron's output to range [-1, 1]. Tanh can also be considered as a scaled sigmoid function, i.e., $tanh(x) = 2\sigma(2x) - 1$.

**Rectified Linear Unit (ReLU).** The ReLU is one of the most popular activation functions. It cuts the activation of the neuron at zero by computing the function $ReLU(x) = max(0, x)$.

### 3.1.3 Training Using Gradient Decent Algorithm

In a feed-forward neural network, the connections between the nodes of different layers do not form a cycle. In other words, there is no feedback connections that connects an output of the model to itself. Therefore, the information in a feed-forward neural network flows in only one direction: from the input to the output by passing each hidden layer in the network. The purpose of a feed-forward neural network is to approximate a function $y = f(x, \theta)$, where $x$ is the input and $\theta$ is the set of model parameters (i.e., the layers' weights and biases).

In a forward pass, the output of the model is computed based on the input: each layer performs a transformation on the input data and passes the result to the next layer. When the output of the network is computed, a loss (a.k.a error) function is calculated based on the model output and expected output (e.g., actual class label associated with the input). In the next step, the parameters of the neural network are adjusted to gradually reduce the loss value. This process is performed in a

backward pass, using gradient descent algorithm or similar, from the output node to the input.

In Chapter 6, we use the well-known cross-entropy loss for binary classification problem, defined as $cross\_entropy(y, \hat{y}) = -ylog(\hat{y}) - (1-y)log(1-\hat{y})$, for a given input sample. Here, $y$ is the target class label of the input, while $\hat{y}$ is the predicted class.

In addition, we use weighted contrastive loss [CHL05] to train a twin network to classify images as either "same" or "different". The purpose of this loss is to enable the network to differentiate between the two images. Equation 3.1 shows how the weights are updated based on this loss for two input samples $I_1$ and $I_2$.

$$L(\theta, Y, I_1, I_2) = \frac{1}{2}(1-r)(1-Y)({D_\theta}^2) + \frac{1}{2}rY(max(0, \mu - D_\theta))^2 \quad (3.1)$$

$\theta$ 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 same images. $D_w$ is the Euclidean distance between the outputs of the twin networks for the input image pairs ($I_1$ and $I_2$). $r \in [0, 1]$ is the weight (importance) assigned to the positive (different) class and $\mu \in \mathbb{R}$, $\mu > 1$ is a margin.

### Types of Gradient Descent

Depending on the number of samples used for computing the loss function, gradient decent algorithm can be categorized into three category: (1) Batch Gradient Decent (BGD), (2) Stochastic Gradient Decent (SGD) and (3) mini-batch gradient decent. **Batch Gradient Decent (BGD) optimization.** In the BGD algorithm, the weights are incrementally updated after one pass over all the training samples. Such pass over training data is referred to as a training "epoch". In other words, in BGD

all the training samples are used to compute the loss function. The value of the loss is then used to updated the network's weights.

**Stochastic Gradient Decent (SGD) optimization.** Unlike BGD algorithm, in SGD algorithm the weights are updated after observing each sample individually.

**Mini-batch Gradient Decent optimization.** In mini-batch gradient decent algorithm, the weights are updated after processing a fixed size of samples from the training data. The size of this set is referred to as "batch-size". The batch-size is a hyper parameter and it's value is usually selected based on the memory and computational capacity of the system on which the training is performed. In this dissertation, we use mini-batch gradient decent algorithm for training the neural networks.

### Gradient-based Optimization

In this section, we review two different gradient-based optimizer that we use in this dissertation for training the Human Perception Discriminator and CEAL neural networks (see Chapters 5 and 6).

**Gradient decent optimizer.** In this method, the weights of the model are adjusted in the negative direction of gradient of loss function with respect to weights. This is because the gradient represents the direction in which the function is maximized. Particularly, the gradient of the loss with respect to weights are computed using $\frac{\partial L(\theta)}{\partial w_{ij}}$, where L is the loss value and $w_{ij}$ is the weight of $j^{th}$ node in $i^{th}$ layer of the network. Once this derivatives are computed, the weights are updated using Equation 3.2. In this equation, $\eta$ is hyperparameter known as "learning rate". Learning rate defines the step sizes we take in the opposite direction of gradient to minimize the loss function. Choosing the right learning rate is very important: if it

is too large, the model might not converge to local/global minima; while if it is too small, the learning would be so slow.

$$w_{ij} := w_{ij} - \eta \frac{\partial L(\theta)}{\partial w_{ij}} \tag{3.2}$$

**Adam optimizer.** In traditional gradient descent method, a single learning rate is used for all weight updates. In addition, the value of this learning rate is constant during training. However, Adam optimization algorithm "computes individual adaptive learning rates for different parameters from estimates of first moment (the mean) and second moment (the uncentered variance) of the gradients." [KB15]. Particularly, the decaying first moment ($m_t$) and second moment ($v_t$) are estimated as in Equation 3.3 and Equation 3.4. $m_t$ and $v_t$ are estimates of the first and the second moments of the gradients respectively.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.3}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) {g_t}^2 \tag{3.4}$$

### 3.1.4 Representation Learning Using Deep Neural Networks

Representation or feature learning is a field of machine learning that seeks to construct a suitable representation of the input vectors for the learning task. The idea is to transform the raw input vectors into a new space that captures the most important factors of the unknown generating distribution of the input data. Principal Component Analysis (PCA) [F.R01] is an ancestor of such solutions that performs a linear transformation of the input vectors to capture the directions (components) with the highest variability.

In contrast, DNNs leverage linear and non-linear transformations, performed by several processing layers, in order to learn a high level abstract feature set of the input data. A well-known method for finding a feature representation for images is to use an auto-encoder [Bal87]. An auto-encoder is a neural network with two components: an encoder that transforms the input image into a compact code; and a decoder that attempts to reconstruct the input image from the code. An auto-encoder is trained in an unsupervised fashion, to minimize the reconstruction error of the input at the output. When the network is trained, the code can be considered as the compact representation of the salient features of the input data.

Further, the representation learned by DNNs can be re-purposed for a new task. This process is referred to as "transfer learning" [YCBL14]. Empirical results have shown the effectiveness of representations learned by DNNs for different tasks including image classification [YCBL14, DJV$^+$14, PCS17], and verification of different biometric information [CHL05, MCP$^+$15, PHJ16].

Particularly, one can use the entire or part of a pre-trained model weights (source model) to build a new network for performing a different similar task (target model). In this setting, the target model weights are initialized by transferred weights from the source model. Then, these weights can be retrained for the new task. In another setting, additional new layers are stacked on the (partial) copy of the source model and the new layers are fine-tuned for the target task using a labeled dataset of samples.

Transfer learning facilitates the learning process for the new network as the extracted feature by the source model are also useful for performing the new task. For instance, let us assume we have a DNN that is trained to classify images of certain objects, e.g., cat, dog, flower, tree, house. We can use this network to train a classifier for distinguishing other classes of images, such as different types

of flowers, e.g., iris, lily, etc. For this, we can stack one of the deepest layers of the pre-trained network with several additional (fully connected) layers. Then, we retrain the new layers to optimize the network weights for the new task using labeled samples of flower images. The source model can extract useful image features (e.g., features that resemble Gabor filters, color, etc.) that are also useful for performing the target task (i.e. flower classification).

In addition, in a recent study Elsayed et al. [EGSD18] introduced reprogramming DNNs. This is a new form of transfer learning approach where instead of using pre-trained model weights, or the features extracted by this model, the input to the pre-trained model is modified so that the network performs well on the new task. This can be considered as reprogramming the network.

In contrast to image classification problem, ai.lock and CEAL image-based authentication solutions differ in their need to ensure that two object/scene images are the same or different for the purpose of authentication. In ai.lock and CEAL architectures, we exploit the ability of DNNs to learn features of the input images and capture the important underlying explanatory factors of images. We conjecture that such features will have small variations among images of the same object or scene, captured in different circumstances.

### Pre-trained Deep Neural Networks

We are interested in the task of extracting robust and salient features of images using the abilities of DNNs. Training a DNN with millions of parameters is computationally expensive and requires a large training dataset of labeled samples, rarely available in practice. Instead, we employed a "transfer learning" [SZL15] approach: obtain a trained DNN and use it for a similar task.

Particularly, we exploit different pre-trained networks for image feature extraction: when the image is input to a pre-trained network, we consider activations of specific layer, or multiple layers, as corresponding image features.

In this dissertation, we use two versions of Inception network for image feature extraction. Inception networks use a specific module a.k.a Inception modules in their architecture. The Inception module was introduces by Google engineers to enable the network to have a deeper architecture, hence pushing the performance of image classification task for computer vision. This module uses filters of different sizes in the convolutional layers. The output of convolution operations using these filters are then concatenated, enabling the network to learn from multiple sources of information when performing the task. This continuous effort has resulted in several versions of Inception. Each version provides an improvement over the prior version, however, depending on the task in hand and the dataset in use, an earlier version can have better performance over a later one.

**Pre-trained Inception.v1**. For image feature extraction in CEAL, we use Inception.v1 [ten17, SLJ+15] network pre-trained on ImageNet dataset [DDS+09], of 1.2 million images of 1,000 different object categories, for image classification (see Chapter 6).

**Pre-trained Inception.v3**. Similar to Inception.v1, Inception.v3 [SVI+16a] is a network pre-trained on ImageNet dataset [DDS+09], of 1.2 million images of 1,000 different object categories. We use this network for image feature extraction in ai.lock (see Chapter 5).

### 3.1.5 Generating Images Using Deep Neural Networks

In § 3.1.1, we described the generative models. Generative models digest a large dataset, related to some domain (e.g., images, sentences, sounds, videos, etc.), and learn to generate data samples that look similar to this dataset. In this dissertation, we focus on image data generation.

A Deep Generative Model (DGM) is usually trained, using unsupervised learning, to summarize explaining key features of images in training dataset. The trained model can be further used for drawing samples from this data distribution, i.e. they can generate fake but visually realistic and plausible images. There are two major classes of generative models: Variational AutoEncoders (VAEs) [KW13] and Generative Adversarial Networks (GANs) [GPAM$^+$14]. VAEs consist of two DNNs, i.e., a encoder part that transforms the input images to a compact hidden code (a.k.a. latent variable); and a decoder part that learns to reconstruct the hidden code back into the input image. VAEs are trained to minimize the reconstruction loss of the decoder when there is a prior on the distribution of latent variables, i.e., the encoder is constraint to generate latent variables that follow a specific distribution (e.g. normal, or uniform).

A GAN [GPAM$^+$14] is implemented by a system of two competing DNNs: (1) a generator network ($G$) that transforms the input latent vector into an image. The latent vector is usually selected from a simple distribution such normal or uniform distributions. We assume that the latent vector components are randomly drawn from $U(-1, 1)$; (2) a discriminator ($D$) network that differentiates between instances from the real images and synthetic images generated by the $G$ network. $G$ and $D$ are trained alternately: $D$ is trained using the images generated by $G$ and a set of real images from a training dataste. $G$ uses the output of $D$ for the images it generated to generate better images that can deceive $D$ into believing they are real.

The competition will drive $G$ to generate image samples that look like images from the training dataset.

In Chapters 4 and 5, we use a GAN model with the same architecture as in Deep Convolutional Generative Adversarial Networks (DCGAN) [RMC15] to generate a huge number of synthetic image samples that look similar to the real images in our training datasets. We use these synthetic datasets to attack Pixie and ai.lock image-based authentication solutions. In addition, in Chapter 6, we use a DCGAN generator in CEAL system. We modify the process of training this DCGAN to generate images that are not only realistic, but distinguishable by human visual system. The images that CEAL generates can be used to represent a key fingerprint corresponding to a public key or identity.

## 3.2 Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) seeks to reduce the dimensionality of data, while probabilistically preserving the distance properties of the input space. LSH was initially used to solve the near neighbor search problem in high dimensional spaces [IM98]. While seemingly the ideal candidate to provide the ai.lock functionality (i.e. mapping an image into a binary string), LSH does not work well on images: images of the same scene or object, captured in different conditions, e.g., angle, distance, illumination, will have dramatically different pixel values, leading to a high distance between the images and thus also between their LSH values.

We use however Charikar's [Cha02] LSH as a building block in ai.lock. Charikar's [Cha02] LSH defines a family of hash functions in the space $\mathcal{R}^d$. Specifically, the LSH function $h_r$ is based on a randomly chosen $d$-dimensional Gaussian vector with independent components $r \in \mathcal{R}^d$, where $h_r(u) = 1$ if $r \cdot u \geq 0$ and $h_r(u) = 0$ if

$r \cdot u < 0$ ($\cdot$ denotes the inner product). This function provides the property that $Pr[h_r(u) = h_r(v)] = 1 - \frac{\theta(u,v)}{\pi}$, for any vectors $u$ and $v$, where $\theta(u, v)$ denotes the angle between the input vectors.

## 3.3 Error Correcting Codes

In this dissertation, we use binary Bose-Chaudhuri-Hocquenghem BCH [BRC60, Hoc59b] codes. A t-error-correcting BCH code can correct up to t bits. In this case, the code words, generated by encoder, are guaranteed to be at least in d hamming distance of each other, where $d \geq 2t + 1$. We represent a t-error correcting code with message length n and code word length $k$ bits as $BCH(n, k, t)$.

CHAPTER 4

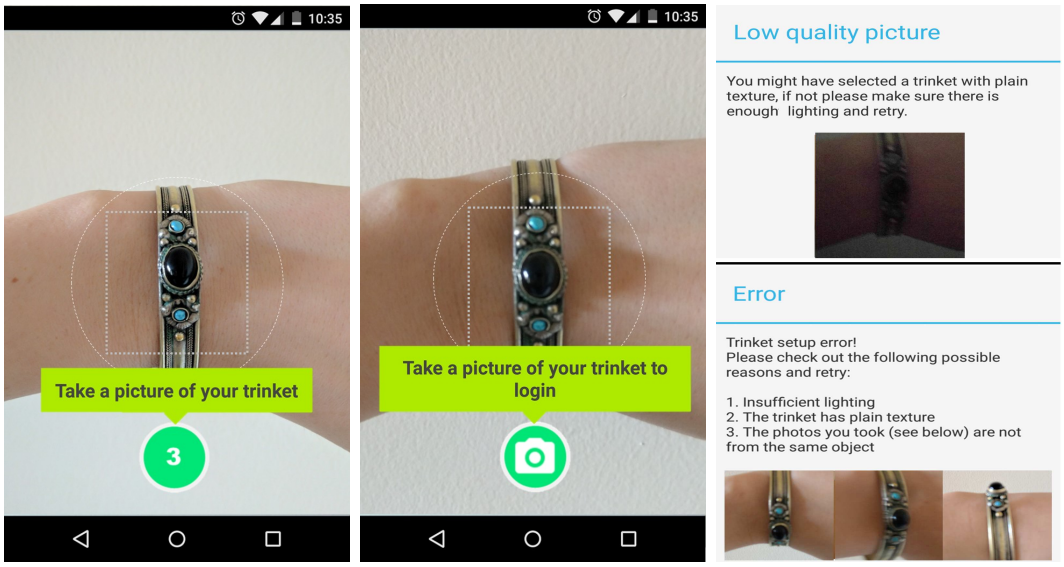
# PIXIE: IMAGE-BASED REMOTE AUTHENTICATION

## 4.1 Introduction

We introduce Pixie, a novel, camera based two factor authentication solution for mobile and wearable devices, see Figure 4.1 and [CaS17d] for a short demo. A quick and familiar user action of snapping a photo is sufficient for Pixie to simultaneously perform a graphical password authentication and a physical token based authentication, yet it does not require any expensive, uncommon hardware. Pixie establishes trust based on both the knowledge and possession of an arbitrary physical object readily accessible to the user, called *trinket*. While the trinket is essentially an authentication token, it is not (necessarily) electronic, but can be any object easily accessible to the user during the authentication process, e.g., wallet, watch, clothing pattern. Pixie also differs from authentication based on biometric, since the trinket can be changed, as users change accessories they wear or carry.

Just like setting a password, the user picks a readily accessible trinket of his preference, e.g., a clothing accessory, a book, or a desk toy, then uses the device camera to snap trinket images (a.k.a., *reference* images). All the user needs to do to authenticate is to point the camera to the trinket. If the captured *candidate* image matches the reference images, the authentication succeeds.

**Challenges.** Building a secure and usable trinket based authentication solution is difficult. Unlike biometrics based solutions, trinkets can be chosen from a more diverse space than e.g., faces, thus lack the convenience of a set of well known features. In addition, users cannot be expected to accurately replicate during login, the conditions (e.g. angle and distance, background) of the trinket as in the setup process. Thus, Pixie needs to be resilient to candidate images captured in different




(a) (b) (c)

Figure 4.1: Pixie: (a) Trinket setup. The user takes photos of the trinket placing it in the circle overlay. UI shows the number of photos left to take. (b) Login: the user snaps a photo of the trinket. (c) Trinket setup messages provide actionable guidance, when the image quality is low (top), or the reference images are inconsistent (bottom).

circumstances than the reference images. To address these challenges, Pixie requires the users to capture multiple trinket images in registration phase. In addition, we leverage robust keypoints [BTVG06, RRKB11] extracted from images of the trinket to perform image matching. Particularly, we identify a novel set of features extracted from a 1-to-1 matching of candidate to reference image keypoint descriptors. We then train a classifier on these features to decide if the candidate and reference images "match".

We identified an additional challenge in early pilot studies: Pixie users can use low quality reference trinkets, (e.g. with uniform textures, or inconsistent reference images with different viewing angles), or capture low quality images of their trinkets, (e.g., blurry, or with improper lighting conditions, see Figure 4.2(d)-(f)). In order to help the users pick high quality images of trinket, we develop features that capture

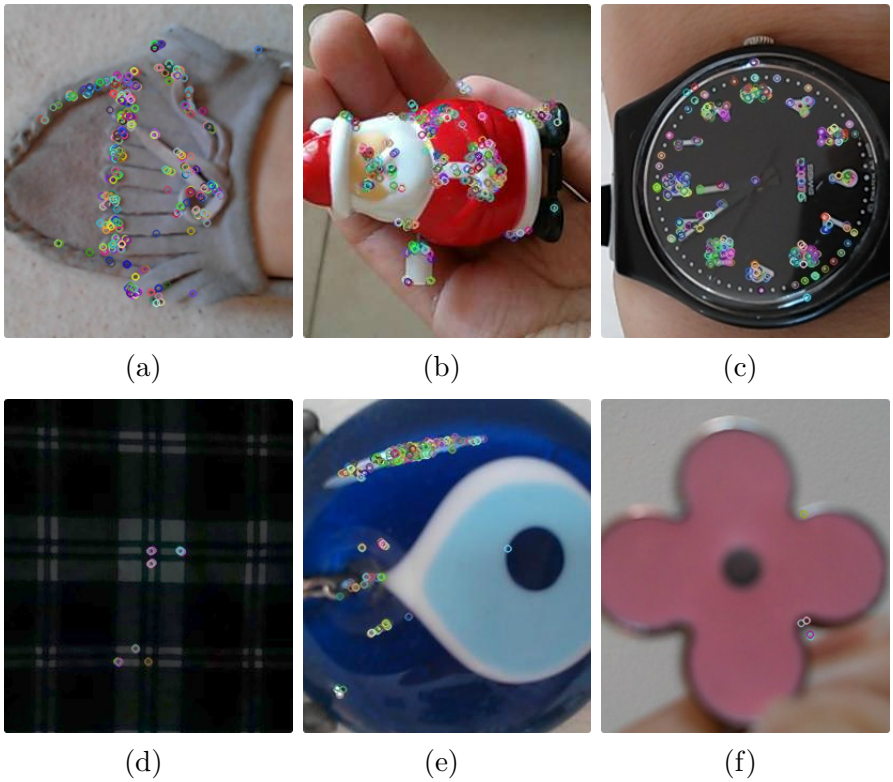



Figure 4.2: Examples of good (a-c) and low quality (d-f) trinket images. Trinkets are small (parts of) objects carried or worn by users, thus hard to steal and even reproduce by adversaries. ORB keypoints are shown as small, colored circles. Good images have a high number of keypoints on the trinket. Low quality images are due to (d) insufficient light conditions on shirt section, (e) bright light and reflection, (f) image blur, or uniform, texture-less trinket.

the quality of reference and candidate images as defined by the likelihood of causing false accepts or false rejects. We use these features to implement image filters that detect and reject low quality images before they can be used as Pixie trinkets. We introduce *two rules of Pixie filters* for rejecting trinket images: (i) images on which they predict Pixie will fail, and (ii) images with feature values for which Pixie has not been trained.

It is crucial to give the user actionable feedback about how to choose a better trinket when the Pixie filters reject a trinket image. For instance, a set of reference images can be rejected because they contain different trinkets, or because one of

| Solution | Success rate (%) | Entry Time (s) | Number of trials before success |
|---|---|---|---|
| **Pixie** | **84.00** | **7.99 (Std=2.26, Mdn=8.51)** | **1.2 (Std=0.4, Mdn=1)** |
| Text password (MyFIU) | 88.10 | 12.5 (Std=6.5, Mdn=11.5) | 1.4 (Std=1.02, Mdn=1) |
| Text password (comp8) [SKD+14]* | 75.0-80.1 | (Mdn=13.2) | 1.3 |
| Eye tracking [LDGW15] | 77.2-91.6 | ¡ 9.6 | 1.37 (Std=0.8, Mdn=1)-1.05 (Std=0.3, Mdn=1) |
| GazeTouchPass [KAH+16] | 65 | 3.13 | 1.9 (Std=1.4, Mdn=1) |
| Face biometric [TSK+12] | 96.9 | (Mdn=5.55) | N/A |
| Face & eyes [BCF+13]* | N/A | 20-40 | 1.1 |
| Face & voice [TSK+12] | 78.7 | (Mdn=7.63) | N/A |
| Voice biometric [TSK+12] | 99.5 | (Mdn=5.15) | N/A |
| Gesture (stroke) biometric [TSK+12] | 100 | (Mdn=8.10) | N/A |
| Android pattern unlock [HDLE16] | 87.92 | 0.9 (Std=0.63, Mdn=0.74) | 1.13(Std=0.06, Mdn=1.11) |
| Passpoints [CFS+09]* | 57 | 18.1 (Mdn=15.7) | 2.2 |
| Xside [DLHvZ+14] | 88 | 3.1-4.1 | N/A |
| SmudgeSafe [SSB+14] | 74 | 3.64 (Std=1.66) | N/A |

* The study device is a computer.

Table 4.1: Comparison of usability related metrics of Pixie's camera based two-factor authentication approach with text, biometric and graphical password authentication solutions. The Pixie user entry time is faster than typing text passwords. The results of text-based passwords evaluated in § 4.6.2 are consistent with those from previous work. Pixie's median of login trials until success is 1, similar to other solutions.

them is blurry. Most supervised learning classifiers are not easily interpretable, thus cannot indicate the nature of the problem. In order to provide meaningful actionable feedback, we identify feature threshold values that pinpoint problem images and naturally translate into user instructions (see Table 4.6), e.g. blurry images, and inconsistent reference sets.

**Implementation and evaluation**. We implement Pixie for Android, and show using an extensive evaluation that Pixie is secure, fast and usable, and perceived as such by users. Pixie achieves a False Accept Rate (FAR) of 0.02% and a False Reject Rate (FRR) of 4.25%, when evaluated over $122,500$ authentication instances. Pixie processes a login attempt in 0.5s on a HTC One mobile device.

Table 4.1 compares usability related metrics of Pixie's camera based two-factor authentication approach with text, biometric and graphical password authentication solutions. While Pixie takes longer than biometric authentication based on face [TSK+12], it is still faster than several authentication solutions based on

gaze [BCF+13, LDGW15]. We note that while fingerprint based authentication is fast and convenient [BUI+15], it is only applicable to devices that invest in such equipment. In contrast, cameras are ubiquitously present, including on wearable devices such as smartwatches and smartglasses.

To evaluate the security of Pixie, we introduce image based dictionary, or "pictionary" attacks, based on public trinket image datasets and images that we collected online. On pictionary attacks consisting of 14.3 million authentication attempts, Pixie achieves a FAR below 0.1%. We show that knowledge of the trinket type does not provide an adversary with a significant advantage: on our online trinket image dataset, the average number of "trials until success" exceeds $5,500$ irrespective of whether the adversary knows the type of the trinket or not. In addition, we introduce and study the concept of *master* images, whose diverse keypoints enable them to match multiple trinkets. We develop features that enable Pixie to reduce the effectiveness of master images.

A user study performed with 42 participants over 8 days in 3 sessions, reveals that Pixie is "discoverable": *without prior training* and given no external help, 86% and 78% of the participants were able to correctly set a trinket then authenticate with it, respectively. Pixie's trinkets were perceived as more memorable than text passwords, and were also easily remembered 2 and 7 days after being set.

Without any additional practice outside of the 3 sessions, participants entered their trinket progressively faster. Participants think Pixie is easier to use, more memorable and faster than text passwords. We found that preference of Pixie over text passwords correlates positively with its preference on ease of use, memorability and security dimensions and overall perception of trinket memorability and willingness to adopt Pixie. In addition, 50% of participants reported that they preferred Pixie over text passwords. 62% of the participants were willing to adopt Pixie.

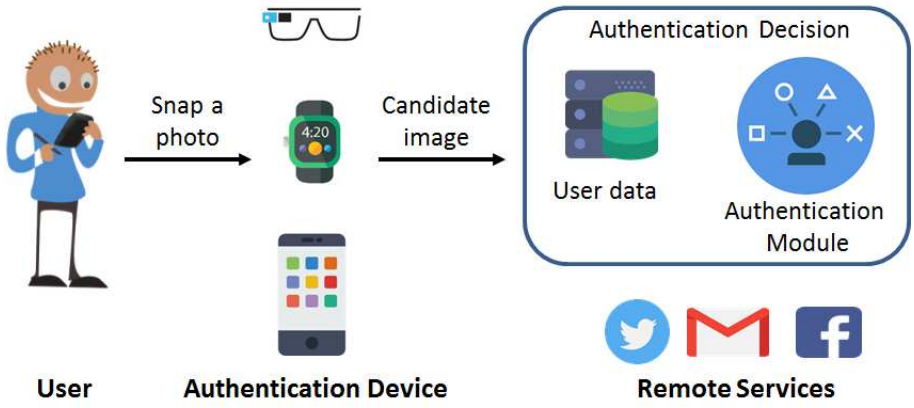



Figure 4.3: Pixie system model: the user authenticates through a camera equipped device (smartphone, smartwatch, Google Glass, car), to a remote service, e.g., e-mail, bank, social network account. The remote service stores the user credentials and performs the authentication.

## 4.2 Model and Applications

### 4.2.1 System Model

Figure 4.3 illustrates the system model. The user has a camera equipped mobile device, called the *authentication device*. Authentication devices include smartphones, tablets, resource constrained devices such as smartwatches and smartglasses, and complex cyber-physical systems such as cars. The user uses the authentication device to access remote services such as e-mail, bank and social network accounts, or cyber-physical systems, e.g., home or child monitoring systems.

We assume that the user can select and easily access a physical object, the *trinket*. The user sets the authentication secret to consist of multiple photos of the trinket, taken with the device camera. We call these "reference" images, or reference set. To authenticate, the user snaps a "candidate" image of the trinket. This image needs to match the stored, reference set. As illustrated in Figure 4.3, in this chapter, we focus on the scenario where the user is willing to authenticate to a remote

service using Pixie. The remote service will store the reference images securely and perform the image match over candidate images captured on and sent by the mobile device. However, the mobile device can also be used to store the reference images, and perform the image matching. In this case, the device associates the reference images with the user's remote authentication credentials (e.g. OAuth [DH12]). If the image match succeeds, the credentials are sent to the remote service. In § 4.7 we compare the merits and drawbacks of each approach.

It worth mentioning that, Pixie can be used both as a standalone authentication solution and as a secondary authentication solution, e.g., complementing text based passwords.

### 4.2.2 Application

While this chapter centers on a remote service authentication through a mobile device scenario, Pixie has multiple other applications such as authentication in camera equipped cyber-physical systems. For instance, cars can use Pixie to authenticate their drivers locally and to remote services [Sec17]. Pixie can also authenticate users to remote, smart house or child monitoring systems, through their wearable devices. Further, doorlocks, PIN pads [Sec17, Sch17] and fingerprint readers can be replaced with a camera through which users snap a photo of their trinket to authenticate.

Pixie can be used as an alternative to face based authentication when the users are reluctant to provide their biometric information (e.g. in home game systems where the user needs to authenticate to pick a profile before playing or to unlock certain functionalities). Pixie can also be used as an automatic access control check-

point (e.g. for accessing privileged parts of a building). The users can print a visual token and use it to pass Pixie access control checkpoints.

In addition, given the large number of people who work from home [Sed14], Pixie can provide an inexpensive 2FA alternative for organizations to authenticate employees who are connecting to the private network remotely [Gol]: replace the hardware tokens with user chosen Pixie trinkets.

We note however that as we discuss later, Pixie may be unsuitable in authentication scenarios that include (1) a high risk associated with external observers, (2) poor light conditions, (3) unpredictable movements, e.g., while walking or in public transportation, or (4) depending on the trinket object type, situations where the user cannot use both hands.

### 4.2.3 Adversary Model

We assume that the adversary can physically capture the mobile device of the victim. We also assume that the adversary can use image datasets that he captures and collects (see § 4.4) to launch brute force **pictionary attacks** against Pixie (see § 4.5.2).

Similar to PIN based authentication to an ATM, Pixie users need to make sure that onlookers are far away and cannot see the trinket and its angle. We assume thus an adversary with *incomplete surveillance* [FA12], who cannot observe or record the trinket details. However, we consider a **shoulder surfing** attack flavor where the adversary sees or guesses the user's trinket object type. The adversary can then use datasets of images of similar objects to attack Pixie (see § 4.5.2).

Further, we also consider an adversary that attempts to launch a **master image attack**, i.e., identify images that contain diverse features and match many trinkets.

Example master images include "clutter" images, with an array of shapes, colors and shadows (see § 4.5.2).

In remote authentication scenario, we assume that the reference trinket images are transferred to the remote server through a secure channel (e.g. SSH). These images are then stored on the server securely through encryption. In the scenario where the reference images have to be stored on the mobile device, we assume that the adversary cannot access the stored image password: the images could be secured through hardware-level protection, e.g., [Tru17], or through privacy preserving feature extraction and matching solutions [WHWR16, QYR+14, HLP12], see § 4.7 for a discussion. Depending on the scenario, the adversary seeks to either authenticate on the captured device, or to use the device to gain access to the victim's online service accounts.

It is important to note that, we are not assuming a naive or unmotivated adversary whose only means is to take photos of trinkets, as such limited adversaries are known to create a false sense of security [BML06]. The adversary that we assume is able to launch massive image-based attacks on Pixie using public image libraries.

## 4.3 Pixie

### 4.3.1 Pixie Requirements

Pixie is a two factor authentication solution as it requires both a possession factor and a knowledge factor to authenticate the user. The possession factor is the trinket. The knowledge factor is the knowledge of the trinket itself, its angle and section used to authenticate. In addition to being resilient against attacks (see § 4.2.3), Pixie needs to satisfy the following requirements:

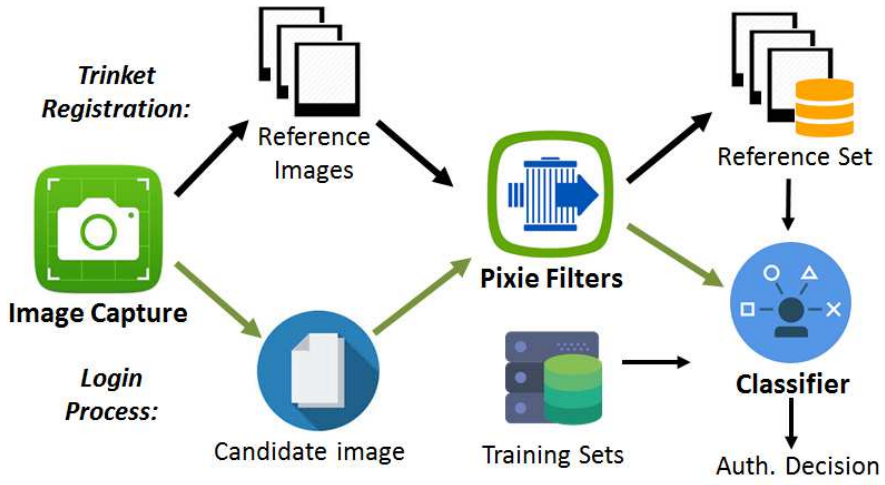



Figure 4.4: Pixie registration and login workflows: to register, the user captures "reference images" of the trinket, which are filtered for quality and consistency. To authenticate, the user needs to capture a "candidate image" of the trinket that matches the reference images.

- **Trinket image quality**. Pixie needs to ensure the quality of trinkets and images. Early pilot studies showed that not all the trinkets that the users chose, or the photos that they took, were suitable for authentication.
- **Trinket match**. Pixie needs to match images of the same trinket, even when captured with a different background, lighting, or from a slightly different distance or angle.
- **Discoverability**. New users should easily discover the functionality of Pixie.
- **Deployability**. Pixie should be easy to integrate into existing systems.

Figure 4.4 depicts the modular approach we use for Pixie to address these goals. The image capture module seeks to address part of the first requirement, by facilitating the capture of high quality trinket images. The authentication module tackles the second requirement through the use of trained classifiers to match trinket images. To simultaneously address the first and third requirements, i.e., to ensure the discoverability of Pixie while guiding new users through the capture of high quality

photos and the choice of visually complex objects as the secret, the filter module detects and eliminates low quality images and invalid reference sets. We now detail each module.

### 4.3.2 Image Capture & Feedback

We performed pilot studies to identify early problems with the Pixie user interface. For instance, during the pilot studies, some users captured trinket photos whose background provided more features than the trinkets. This revealed that the trinket needs to be the main object in captured images. To simultaneously satisfy this requirement, and the trinket quality requirement above, we design Pixie to guide the user to take larger photos of trinkets. We achieve this by overlaying a circle on the camera image: the user needs to fit the trinket impression inside the circle (see Figures 4.1(a) and 4.1(b)). Since Pixie does not allow zooming in, the user needs to bring the camera closer to the trinket, hence take a larger photo. Pixie crops the image, and keeps only the largest rectangle parallel to the sides of the device that fits the circle.

In addition, we observed that the quality of trinket images captured by the users could be low (e.g. blurry or dark, see Figure 4.2), or the users may take inconsistent trinket images in the registration phase. To ensure the quality of trinket images and the consistency of reference images, we identified common problems that occur during the image capture process (e.g., insufficient light, trinket with plain texture). Then, we mapped prefilter rejection decisions provided by Pixie's image filter (see § 4.3.4) into informative error messages (see Figure 4.1(c)). Furthermore, to facilitate the discoverability of Pixie, we designed and included a step by step in-app instruction guide on how to use Pixie (see Figure 4.12). Table 4.2 summarizes

| Requirement | Pixie Feature |
|---|---|
| Increase the size of trinket &<br>Reduce the background area in trinket images | 1. Disable camera zoom<br>2. Overlay a circle as the target area on the camera view |
| Ensure the quality of reference and candidate images<br>Ensure consistency of reference images | 1. Design prefilters for checking the quality of images<br>2. Translate the prefilter criteria into an actionable feedback to the users |
| Improve the discoverability of Pixie | 1. Show number of remaining images to take in registration screen<br>2. Show camera capture icon for login page<br>3. Add step by step in-app instruction on how to use Pixie |

Table 4.2: Summary of user interface improvements identified during pilot studies.

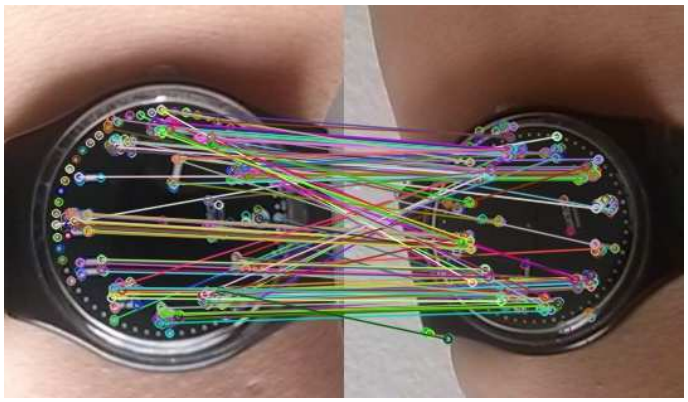


Figure 4.5: Example ORB keypoint matches between two images of the same trinket, taken in different conditions. Each line represents a match: it connects matching keypoints (shown as small colored circles) from each image.

the design improvements we made to the Pixie UI.

### 4.3.3 The Authentication Module

The authentication module is responsible for addressing Pixie's second requirement (see § 4.3.1), of matching the candidate image against the reference images. Pixie extracts robust keypoints from these images, then computes a 1-to-1 mapping between the resulting keypoint sets (see Figure 4.5 for an illustration), and filters out low quality matches. Pixie extracts a suite of features from the keypoint match process and uses supervised learning to decide if the candidate image matches the reference set. We now detail this process.

| Symbol | Description |
|---|---|
| $\overline{R}$ | The set of reference images |
| $R$ | Any of the reference images in the reference set ($\overline{R}$) |
| $C$ | The candidate image |
| $T(\overline{R})$ | Template image of reference set ($\overline{R}$) |
| $NNSim(C, \overline{R})$ | Nearest neighbor similarity of $C$ to $\overline{R}$ |
| $FNSim(C, \overline{R})$ | Furthest neighbor similarity of $C$ to $\overline{R}$ |
| $AvgRefNN(\overline{R})$ | Avg. nearest neighbor similarity of each reference image |
| $AvgRefFN(\overline{R})$ | Avg. furthest neighbor similarity of each reference image |
| $AvgRefTempl(\overline{R})$ | Avg. similarity of reference images to template image |
| KP-CNT | Number of keypoints in an image |
| DTC-KP | Avg. distance of keypoints to their centroid in an image |
| White-CNT | Number of detected edge (white) pixels of an image |
| DTC-White | Avg. distance of edge (white) pixels to their centroid |
| $MinCrossSim(\overline{R})$ | Min. similarity among all the pairs of images in $\overline{R}$ |
| $MaxCrossSim(\overline{R})$ | Max. similarity among all the pairs of images in $\overline{R}$ |
| $AvgCrossSim(\overline{R})$ | Avg. similarity among all the pairs of images in $\overline{R}$ |
| ORB [RRKB11] | ORB keypoint extraction algorithm |
| SURF [BTVG06] | Speeded Up Robust Features keypoint extraction algorithm |
| RANSAC [FB81] | Random Sample Consensus algorithm for fitting the model to data |
| FLANN [ML09] | Fast Approximate Nearest Neighbor Search |

Table 4.3: Pixie notations and algorithm acronyms.

Table 4.3 summarizes the most important Pixie features notations. Let $C$ denote the candidate image, $\overline{R}$ be the set of reference images, and $R$ be any of the reference images (see § 4.2.1).

### Keypoint matching

We use SURF (Speeded Up Robust Features) [BTVG06] and ORB [RRKB11] algorithms, to extract scale and rotation invariant image keypoints from the candidate and reference images, e.g., shown as small colored circles on images in Figure 4.5 and 4.2. We also extract the descriptors of the keypoints, which represent their characteristics. To determine if a candidate image $C$ and a reference image $R$ contain the same trinket, we compute a 1-to-1 matching between their keypoint descriptors

| Similarity Table | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| $R_1$ | 1 | 0.9 | 0.8 |
| $R_2$ | 0.7 | 1 | 0.6 |
| $R_3$ | 0.7 | 0.8 | 1 |
| $C$ | 0.7 | 0.5 | 0.9 |

Table 4.4: Similarity table of three reference images $\overline{R} = \{R_1, R_2, R_3\}$ and a candidate image $C$. Red cells correspond to the nearest neighbor. $R_1$ is the template image. $AvgRefNN = (0.8+0.7+0.9)/3 = 0.8$, $AvgRefFN = (0.7+0.6+0.8)/3 = 0.7$ and $AvgRefTempl = (0.8 + 0.9)/2 = 0.85$. Then, $minSim(C, \overline{R}) = 0.5/0.7$, $maxSim(C, \overline{R}) = 0.9/0.8$ and $TemplSim = 0.7/0.85$.

(e.g., shown as lines in Figure 4.5). We use brute-force matching for ORB keypoints, where each keypoint of the candidate image is matched with the closest keypoint (in terms of Hamming distance) of the reference image. For SURF keypoints, we use the FLANN-based matcher [ML09].

An exhaustive matching of each keypoint in the candidate image to a keypoint in the reference image will produce low quality, *outlier* matches. We experimented with several existing filters: applying a threshold on the matched keypoint distances, cross checking the matched keypoints in both images, and RANSAC [FB81], to identify and remove outlier matches. The RANSAC based filter performed the best, hence we use it implicitly in the following.

### Image similarities and the template image

Given two images $C$ and $R$, we define their similarity $Sim(C, R)$ to be the ratio between the number of keypoint matches of $C$ and $R$, after the above filter and outlier detection steps, and the number of keypoints in $C$. We also define several concepts as bellow. These concepts are exemplified in Table 4.4.

Given $C$ and the set $\overline{R}$, we define the nearest neighbor similarity of $C$ to $\overline{R}$ as $NNSim(C, \overline{R}) = \max \{Sim(C, R) | \forall R \in \overline{R}\}$, and the farthest neighbor similarity,

| Solution | Features | Details |
|---|---|---|
| Pixie | Keypoint stats. | Statistics of ORB/SURF keypoints |
| | Keypoint nearest neighbors | Keypoint match stats. |
| | Perspective transformation | RANSAC optimal map of match keypoints |
| Pixie filters | Keypoints | Count and spread of keypoints |
| | Edge pixels | Count and spread of edge pixels |
| | Reference quality | Reference image similarities stats. |

Table 4.5: Summary of (top) Pixie features and (bottom) Pixie filter features.

$FNSim(C, \overline{R}) = \min \{Sim(C, R) | \forall R \in \overline{R}\}$.

Given a reference set $\overline{R}$, we define the average nearest neighbor similarity value of each reference image, to the other reference images in $\overline{R}$: $AvgRefNN(\overline{R}) = \frac{\Sigma_{R \in \overline{R}} NNSim(R, \overline{R}-R)}{|\overline{R}|}$. Similarly, we define the average farthest neighbor similarity value of each reference image to the other images in $\overline{R}$: $AvgRefFN(\overline{R}) = \frac{\Sigma_{R \in \overline{R}} FNSim(R, \overline{R}-R)}{|\overline{R}|}$.

In addition, given a reference set $\overline{R}$, we define its *template image*, $T(\overline{R})$, as the reference image $R$ whose value $\sum_{r \in \overline{R}-R} Sim(r, R)$ is the maximum among all reference images in $\overline{R}$. Intuitively, $T(\overline{R})$ is the reference image "closest to the center" of the reference set. We define $AvgRefTempl(\overline{R})$ as the average similarity of images in $\overline{R}$ to $T(\overline{R})$.

### Pixie matching features

We use the above concepts to extract the following features (see Table 4.5, top section for a summary). We use these features to train a supervised learning algorithm (e.g. random forest) to decide whether a candidate image matches to a reference set.

• *Keypoint counts.* The keypoint count of $C$ and $T(\overline{R})$.

• *Match based features.* The number of keypoints in $C$ and $T(\overline{R})$ that match, before the RANSAC filter. The min, max, mean and SD of the distance, size, response and angles between the matched keypoints in $C$ and $T(\overline{R})$, after RANSAC.

• *Quality of the reference set.* $AvgRefNN(\overline{R})$, $AvgRefFN(\overline{R})$ and $AvgRefTempl(\overline{R})$.

• *Similarity to template.* The similarity of $C$ to $T(\overline{R})$, normalized by the average similarity of the images in $\overline{R}$ to $T(\overline{R})$, i.e., $\frac{Sim(C,Templ(\overline{R}))}{AvgRefTempl(\overline{R})}$.

• *Similarity to reference set.* We define $minSim(C,\overline{R}) = \frac{min\{Sim(C,R)|\forall R \in \overline{R}\}}{AvgRefFN(\overline{R})}$: the ratio of the similarity between $C$ and "farthest" reference image, and the average least similarity between reference images. Similarly, we define $maxSim(C,\overline{R}) = \frac{max\{Sim(C,R)|\forall R \in \overline{R}\}}{AvgRefNN(\overline{R})}$.

• *Homography*: Output of homography between $C$ and $T(\overline{R})$: the perspective transformation between the planes of the two images (3 features).

### 4.3.4 Pixie Filters

Early pilot studies revealed that Pixie users can capture low quality images. Such images, either reference or candidate, hinder the ability of the authentication module to discern candidate images, increasing the FRR of Pixie. Furthermore, they impose gratuitous network latency in the remote authentication scenario (see § 4.2.1).

Several conditions may prevent taking high quality images Figure 4.2(d)-(f) shows example outcomes of such conditions, including (i) improper lighting or choice of a complex background that generates irrelevant keypoints, (ii) unsteady hand and (iii) choice of trinkets with constant texture. We also observed that some pilot study participants, during the reference set registration process, took photos containing different trinkets, or different areas of the same trinket. Such reference images may not only reduce the accuracy of the authentication process, but may also introduce vulnerabilities: an attacker may find it easier to capture a candidate image that is similar to one of 3 different reference images, hence, increase the chance to break the authentication mechanism.

To address these issues, we introduce a set of filters (see Figure 4.4) that reject problematic images captured by the user. We propose the *two rules of filtering*, that set out the operation space for Pixie image filters:

- **Filter Rule #1**: Pixie may not willfully fail by operating on images on which it predicts it will fail.
- **Filter Rule #2**: Pixie may not operate in a space where it has not been trained.

In the following, we detail these rules and describe the resulting filters.

### Filter Rule #1: CBFilter and RBFilter

We introduce CBFilter and RBFilter, filters that identify reference and candidate images on which they predict Pixie will fail. The filters leverage the following features, (see Table 4.5 (bottom section) for a summary).

**Filter features**. First, we define KP-CNT as the keypoint count of an image. The intuition for using this feature is that an image with a low KP-CNT (e.g., Figure 4.2(f) with only 5 keypoints) is likely to negatively impact the accuracy of Pixie's matching process. A second feature is based on the center, or *centroid* of the keypoints extracted from an image: let DTC-KP (distance to center of keypoints) denote the average distance between the keypoints of the image and their centroid. DTC-KP measures the spread of the keypoints across the image. The intuition is that a high DTC-KP may indicate that some keypoints do not belong to the trinket but to the background. Third, to detect blurry images, we use the Canny edge detector [Can86] to identify edge pixels that delimit objects in the image. Let White-CNT denote the number of detected edge ("white") pixels of an image. White-CNT is an indicator of the clarity of the image: a low White-CNT denotes a blurred image, with few trinket edges. We also introduce DTC-White (distance to center of

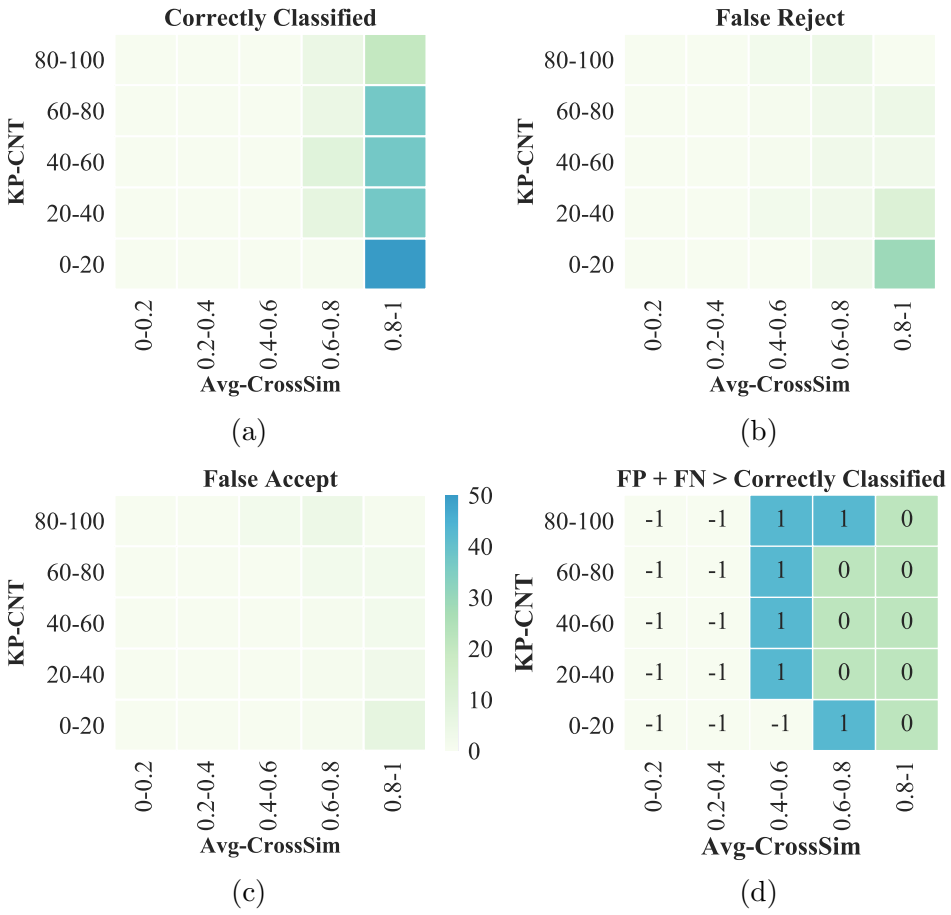



Figure 4.6: Example 2D histograms of KP-CNT of template image vs. $AvgCrossSim(\overline{R})$. (a) Correctly classified instances. (b) False reject instances. (c) False accept instances. The legend in (a)-(c) shows the color code used for the number of authentication instances. (d) Aggregated 2D histogram. The darker regions with 1 in the center have a greater proportion of misclassified than correctly classified instances. The regions with -1 in the center correspond to value ranges on which we have no template images. Conclusion: filter out reference sets with KP-CNT $< 20$ and $AvgCrossSim(\overline{R}) < 0.6$.

white pixels), the average distance of the white pixels to their centroid. DTC-White denotes the spread of the edge pixels, i.e., the size of the trinket. Finally, to detect inconsistent reference images, we define $MinCrossSim(\overline{R})$, $MaxCrossSim(\overline{R})$ and $AvgCrossSim(\overline{R})$, to be the minimum, maximum and average similarity (see § 4.3.3) among all the pairs of images in $\overline{R}$. Small cross similarity values indicate reference images of non-identical trinkets.

**CBFilter: Classifier Based Filter**. Given the reference set $\overline{R}$ and its template image $T(\overline{R})$ (see § 4.3.3), CBFilter uses a suite of features to train a supervised learning algorithm and determine if $\overline{R}$ is suitable to participate in the authentication process. The features include KP-CNT, DTC-KP, White-CNT, DTC-White of $T(\overline{R})$, the average, minimum and maximum of KP-CNT, DTC-KP, White-CNT, DTC-White over all the images in $\overline{R}$, and $MinCrossSim(\overline{R})$, $MaxCrossSim(\overline{R})$ and $AvgCrossSim(\overline{R})$.

**RBFilter: Rule Based Filter**. Pilot studies demonstrated the need to give relevant feedback to users as early as possible: early pilot study participants expressed frustration when they discovered that the photos they took were not suitable at the end of the registration, or worse, during the authentication process. The output of CBFilter cannot however be used to provide meaningful feedback.

To address this limitation, we identified common problems that occur during the image capture process, e.g., improper light, trinket with plain texture or not identical reference images. We then developed a set of rules for these filter features, that (i) predict if an image or image set will not perform well during authentication, and (ii) that can be transposed to one of the problems identified. For instance, we found that a small KP-CNT is associated with insufficient light, blur, or trinkets with a plain texture, while a small $AvgCrossSim$ value can indicate reference images containing non-identical trinkets. Figure 4.1(c) illustrates the feedback provided when the user captures a low quality trinket (top) or inconsistent reference images (bottom).

To identify such rules, we run Pixie on the Pixie dataset, a dataset of reference set and candidate image pairs that are captured in different conditions (see § 4.4 for more details). Specifically, we investigate reference sets and candidate images that contributed to misclassified instances as follows. For each pair of the above filter features, we plot the 2D histogram of instances that were correctly classified, and

| Image type | Filter Rule | Interpretation |
|---|---|---|
| Reference | KP-CNT $< 20$ | Low quality image or plain trinket |
| Reference | DTC-KP $< 30$ | Low quality image or plain trinket |
| Reference | AvgCrossSim $< 0.6$ | Non-identical trinkets in reference set |
| Candidate | KP-CNT $< 20$ | Low quality image or plain trinket |
| Candidate | DTC-KP $> 44,600$ | Out of bounds image |
| Candidate | White-CNT $> 22,400$ | Out of bounds image |
| Candidate | DTC-White $> 160$ | Out of bounds image |

Table 4.6: RBFilter and UBounds filter rules for reference and candidate images, and their real world interpretation. RBFilter (top 2 sections) filters images on which it predicts Pixie will fail. UBounds (bottom section) filters images outside the space seen by Pixie during training.

that contributed to false accepts (FA) and false rejects (FR). Figures 4.6(a)-(c) illustrates this process for the KP-CNT of template images $T(\overline{R})$ vs. $AvgCrossSim(\overline{R})$ pair of features. Then, we aggregate the results for the three 2D histograms, see Figure 4.6(d), by calculating the contribution of each type of classification result (i.e., FA, FR, True Accept (TA) and True Reject (TR)) in a cell of the 2D histogram. The dark regions have a larger proportion of misclassified than correctly classified instances. This enables us to identify "problem" regions, where the contribution of misclassified instances (FA and FR) is larger than that of correctly classified instances (TA and TR). We then define rules, i.e., threshold values, that avoid clusters of problem regions. For instance, based on the bottom area of Figure 4.6(d), we reject reference sets whose template has KP-CNT $< 20$. Similarly, we reject reference sets with $AvgCrossSim(\overline{R}) < 0.6$, as we have none with $AvgCrossSim(\overline{R}) < 0.4$ (cells with $-1$), and those in $[0.4, 0.6]$ are frequently misclassified.

Through a similar process, we have identified several other filtering rules for reference sets and candidate images, and their real world interpretation, see Table 4.6 (top 2 sections). RBFilter uses these rules to reject low quality reference and candi-

date images, and extend Pixie with informative error messages that guide users to improve the quality of captured images.

#### Filter Rule #2: UBounds

We train Pixie on a dataset of images that do not cover the entire value space of the filter features. Pixie cannot make informed decisions on candidate images whose features take values in sub-areas not seen during training. We have identified several such sub-areas for the Pixie dataset. The UBounds filter consists of the "universe boundary" rules listed in Table 4.6 (bottom section), that define these sub-areas.

By rejecting candidate images that satisfy these rules, UBounds presents a conservative performance for Pixie: Pixie would easily reject UBounds rejected candidate images, thus artificially increasing its perceived accuracy. As a result, we do not use this rules when evaluating performance of Pixie.

## 4.4 Implementation and Data

We have implemented Pixie using Android 3.2, OpenCV 2.4.10 and Weka [Wek17]. In order to evaluate the performance of the Pixie features and using several supervised learning algorithms, we have collected and generated the following datasets:

### 4.4.1 Primary Image Datasets

**Nexus image dataset**. We used a Nexus 4 device to capture $1,400$ photos of 350 unique trinkets, belonging to 33 object categories. We selected only objects that can be easily carried by users and are thus ideal candidates for image-based trinkets, e.g., watches, shoes, jewelery, shirt patterns, credit cards and logos. We have captured 4

images for each trinket, that differ in background and lighting conditions, i.e., either indoors using artificial light or outdoors in daylight conditions.

**Google Image dataset**. We used Google's image search site to retrieve at least 200 images from each of the 33 object categories of the Nexus image dataset, for a total of $7,853$ images. This dataset forms the basis of a shoulder surfing attack (see § 4.5.2).

**ALOI dataset**. We use the illumination subset of the Amsterdam Library of Object Images (ALOI) [GBS05a] dataset, that contains 24 different images for 1000 small objects (i.e., natural trinket choices) captured under various illumination conditions. We cropped these images to the size of the Nexus images ($270 \times 312$ pixels), while keeping their object centered.

**Caltech101 dataset**. We use Caltech101 [FFFP04] dataset which is a collection of $9,145$ images of small and large objects, from 101 object categories.

### 4.4.2 Evaluation Datasets

**Pixie dataset**. To evaluate Pixie, we generate authentication instances that consist of one candidate image and 3 reference images. To prevent "tainting", we need to ensure that instances used for testing do not contain reference images that have appeared in a training instance. For this, we use the $1,400$ images of the 350 trinkets, to generate 10 Pixie subsets, each containing 10 folds, as follows. To generate one of the 10 folds of one of the 10 subsets, we first randomly split the 350 trinkets into 10 sets of 35 trinkets each. For each trinket in a set, we randomly select one of its 4 images as candidate; the remaining 3 images are reference images. The trinket then contributes to the fold by one genuine instance (its candidate + its 3 reference images) and 34 "fraud" instances. Each fraud instance combines

the trinket's candidate image with the 3 reference images of one of the other 34 trinkets in the subset. Thus, each fold consists of 35 authentic and $1190 = 35 \times 34$ fraud instances. Then, one of the 10 Pixie subsets contains $12,250$ authentication instances. Thus, the Pixie dataset has a total of $122,500$ authentication instances.

**Attack datasets**. We use the Nexus dataset (§ 4.4) to build 3 *authentication attack datasets* based on the ALOI, Google Image and Caltech101 sets.

We generate the authentication attack instances for each attack dataset, and group them into 10 folds, as follows. We randomly split the 350 unique trinkets of the Nexus dataset into 10 subsets of 35 trinkets each. For each trinket in a subset, we randomly select 3 out of its 4 images, to form a reference set. We then combine this set with each of the images from ALOI, Google Image, and Caltech101 datasets, respectively. We repeat this process for all the 35 reference sets in a fold. Thus, in the ALOI attack dataset, a fold contains $840\text{K} = 35 \times 24\text{K}$ attack instances, for a total of 8.4M ALOI based attack instances. Similarly, the Google Image attack dataset contains 2.7M+ attack instances, while the Caltech101 attack dataset contains 3.2M+ instances.

## 4.5 Evaluation

We evaluate the performance of Pixie's optimal configuration under the attacks introduced in § 4.2.3. We report the performance of Pixie through its False Accept Rate (FAR), False Reject Rate (FRR), Equal Error Rate (EER), and F-measure. FAR can be defined as a the ratio of number of times an authentication system allows an unauthorized access divided by the number of identification attempts. FPR can be defined as the ratio of the number of times an authentication system incorrectly rejects the access of an authorized user divided by the total number of identification

| Keypoint Detector | FAR(%) | FRR(%) | F-measure(%) | EER(%) |
|---|---|---|---|---|
| ORB | 0.10 | 9.83 | 93.08 | 4.87 |
| SURF | 0.07 | 4.80 | 96.40 | 2.80 |

Table 4.7: ORB vs. SURF based Pixie (MLP classifier, no filter) performance, on the Pixie dataset. SURF has lower FAR and FRR compared to ORB.

attempts. EER defined the rate at which both acceptance and rejection are equal. Finally, F-measure is the harmonic mean of precision (i.e. ratio of truly accepted attempts to all accepted attempts) and recall (i.e., $1 - FNR$).

**Experimental setup.** Throughout this section, we have applied 10-fold cross-validation tests [Koh95] to assess how the results of the statistical analysis will generalize to an independent data set. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.

For our experiments, we have used a Mac OS X (2.9 GHz Intel Core i7 CPU, and 8GB DDR3 RAM) and a Nexus 4 smartphone (Quad-core 1.5 GHz Krait, and 2GB RAM; 8MP camera sensor, f/2.4 aperture).

We have used the Weka version 3.7.9 data mining suite [Wek17] to perform the experiments, with default settings: For the back-propagation algorithm of the Multi Layer Perceptron (MLP) classifier, we set the learning rate to 0.3 and the momentum rate to 0.2.

### 4.5.1 Parameter Choice for Pixie

We first identify the parameters for which Pixie performs best.

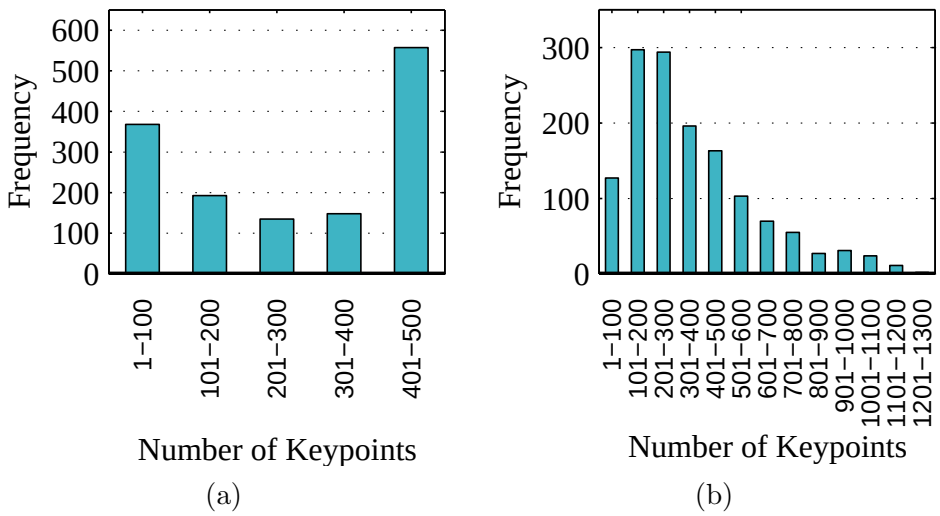



Figure 4.7: Keypoint count distribution extracted from Nexus image set by (a) ORB and (b) SURF.

**ORB vs. SURF for image keypoint extraction**

We compare the performance of Pixie when using two popular keypoint extraction algorithms, ORB [RRKB11] and SURF [BTVG06]. We use a MLP classifier for the Pixie classifier, and no filter. We perform the evaluation through 10-fold cross validation on each of the 10 subsets of the Pixie dataset (see § 4.4). Table 4.7 reports the performance of ORB and SURF: SURF has lower FAR and FRR, leading to an EER that is smaller by 2% than that of ORB.

Figure 4.7 shows the distribution of the per image number of keypoints extracted by ORB and SURF from the Nexus dataset (see § 4.4). We extract at most 500 keypoints with ORB, while with the extended descriptors, SURF discovers up to $1,289$ keypoints. We have extracted $385,361$ ORB keypoint descriptors from the $1,400$ Nexus dataset images, each being 256 bits long.

Figure 4.8(a) compares the average time to extract ORB and SURF keypoints on the Mac and Nexus 4 devices, over 100 Nexus dataset images (that include images with low, medium and high number of keypoints). On both devices, SURF is

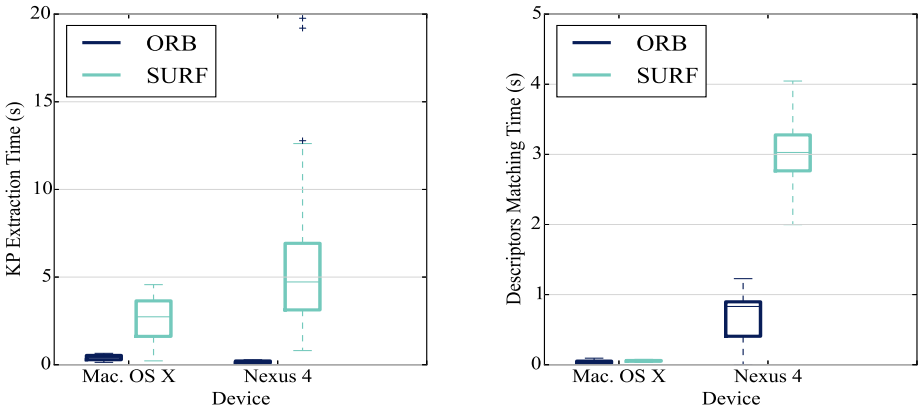



Figure 4.8: ORB vs. SURF: Pixie speed on Mac and Nexus 4. (a) Average time to extract keypoints. ORB takes an average 160ms on Nexus 4. (b) Average time to match keypoint descriptors of two images. Only ORB is viable on the Nexus 4.

significantly slower than ORB. On a Nexus 4, ORB takes 0.15s to extract keypoints, while SURF takes on average more than 2.5s on the Mac and almost 5s on the Nexus 4.

Figure 4.8(b) compares the speed of the matching step, when using ORB and SURF descriptors. These values are computed using the comparison of $10,000$ image pairs: $100 \times 100$, for the aforementioned subset of 100 Nexus images. SURF is consistently slower than ORB: On Nexus 4, SURF takes an average of 2.72s to match the descriptors of a pair of images, and may exceed 3s for images with many keypoints. ORB takes only 0.66s on average to match the descriptors.

Given the trade-off between speed and accuracy, SURF is more suitable when the image processing and matching tasks can be performed on a server. The faster ORB should be preferred in the mobile authentication scenario, when these tasks have to be performed by a mobile device. In the following experiments, we set Pixie's keypoint extraction algorithm to be ORB.

| Pixie Classifier | FAR(%) | FRR(%) | F-measure(%) | EER(%) |
|---|---|---|---|---|
| MLP | 0.10 | 9.83 | 93.08 | 4.87 |
| RF | 0.02 | 10.74 | 93.90 | 3.82 |
| SVM | 0.00 | 12.57 | 93.04 | 10.74 |
| Decision Tree (C4.5) | 0.17 | 11.54 | 91.01 | 7.66 |

Table 4.8: Classifier performance on Pixie dataset using ORB keypoint extractor and no filter. Random Forest and MLP achieve the lowest EER, thus we only use them in the following.

| Images | Filtering Rule | FAR(%) | FRR(%) | F-measure(%) |
|---|---|---|---|---|
| Reference | KP-CNT <20 | 0.09 | 6.60 | 95.06 |
| Reference | DTC-KP <30 | 0.10 | 9.12 | 93.46 |
| Reference | AvgCrossSim <0.6 | 0.07 | 8.10 | 94.53 |
| Reference | All 3 Filters | 0.06 | 4.46 | 96.75 |
| Candidate | KP-CNT <20 | 0.27 | 12.39 | 89.33 |
| Candidate | White-CNT <2000 | 0.25 | 10.55 | 90.61 |
| Candidate | Both Filters | 0.25 | 9.86 | 91.04 |
| Ref. & cand. | All RBFilter Rules | 0.04 | 5.25 | 96.58 |

Table 4.9: Performance of Pixie MLP classifier with RBFilter on the Pixie dataset. The disjunction of all the RBFilters on the reference images reduced the FAR and FRR by more than 40%.

### Classifier Choice

We use the Pixie dataset to identify the best performing classifier for Pixie's authentication module. Table 4.8 shows the results: Random Forest (RF) and MLP outperform Support Vector Machine (SVM) and Decision Tree (DT) through lower FAR and FRR. In the following, we use only RF and MLP as Pixie's classifiers.

### Pixie with filters

In this section, we evaluate the effect of Pixie filters that we have introduced in § 4.3.4 separately and in combination.

**Pixie with RBFilter**. We evaluate the effects of the RBFilter rules of Table 4.6 on the performance of Pixie. For this, in each of the 100 classification experiments (10 folds cross validation over each of the 10 subsets of the Pixie dataset), we remove from the Pixie test fold all the authentication instances that satisfy the rules. Specifically, we have removed the instances whose (i) reference images have fewer than 20 keypoints, a distance to the centroid of keypoints of less than 30, or an average cross similarity of under 0.6, and (ii) whose candidate image has fewer than 20 keypoints. We then run Pixie (with MLP) on this filtered dataset.

Table 4.9 shows that almost all the rules are effective and increases Pixie's F-measure. The disjunction of all the reference set filter rules is the most effective, for an F-measure of 96.75% (3.8% improvement from the unfiltered 93.08% of Table 4.7). The 3 reference set filter rules remove an average of 6.68 reference sets from a testing fold. When also using the candidate image filter, that removes an average of 82.23 authentication instances per testing fold, Pixie's F-measure drops to 96.58%. This is because we count the "valid" instances removed by the candidate filter as part of FRR, even though they are likely of low quality and can mislead Pixie.

**Pixie with CBFilter**. Figure 4.9 illustrates the process we employed to evaluate the impact of CBFilter on the performance of Pixie. To provide a large training set for CBFilter, we first build a Reference Set Bank (RSB), that contains all the reference sets that appear in the 10 subsets of the Pixie dataset. For each such reference set, the RSB also stores its "class", according to the outcome of Pixie (see step 1 of Figure 4.9). If the reference set has been part of any authentication instance (in the Pixie dataset) that was incorrectly classified by Pixie (i.e., either as

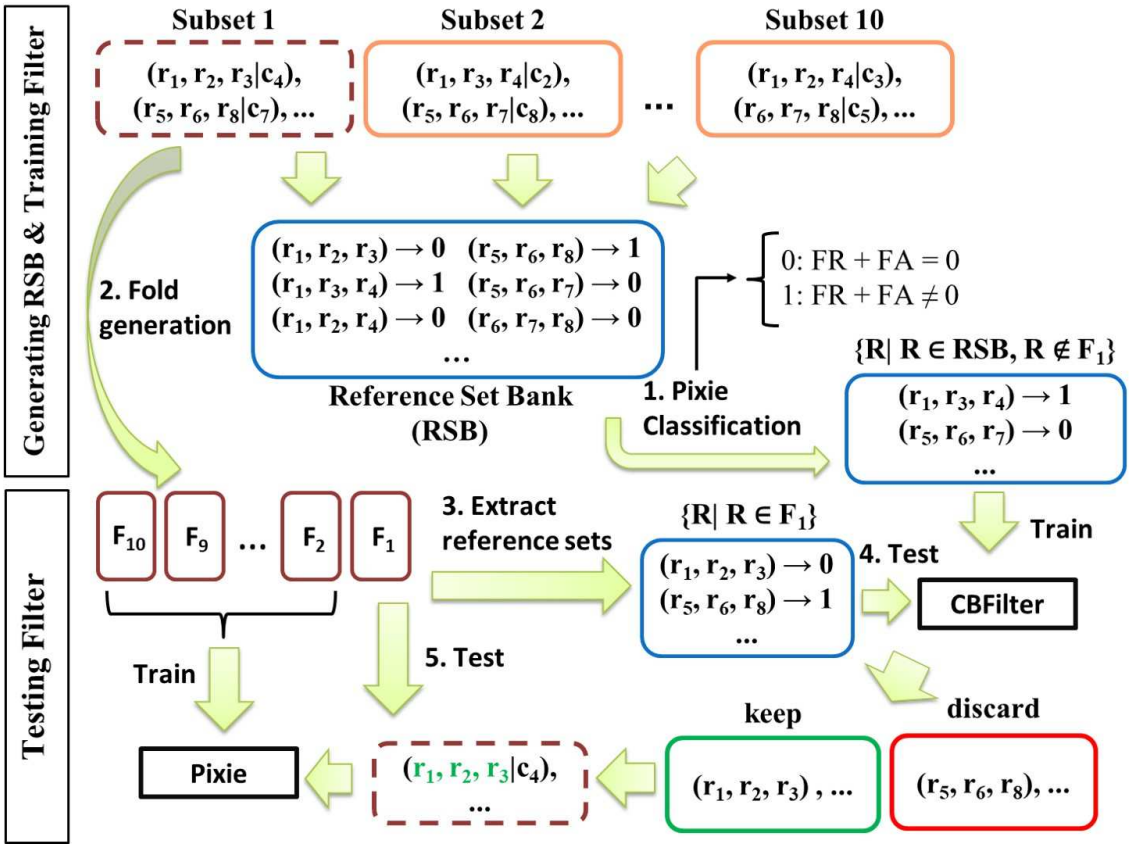



Figure 4.9: CBFilter test methodology: create a large training set (RSB) that does not contain reference images from the fold on which we later run Pixie ($F_1$). Run CBFilter on the reference sets from fold $F_1$, filter the reference sets that fail, then run Pixie on the filtered $F_1$.

FR or FA), its class is 1, otherwise it is 0.

We use the RSB set for the following evaluation process, performed separately for each subset of the Pixie dataset. Each of the subset's 10 folds, (step 2 in Figure 4.9) is used once for testing. Given one such fold, e.g., $F_1$ in Figure 4.9, we extract its reference sets. We train CBFilter on all the reference sets of RSB, that are different from the reference sets of fold $F_1$, then test CBFilter on the reference sets of $F_1$ (step 3 and 4 in Figure 4.9). We filter from $F_1$ all the reference sets that are labeled as 1 by CBFilter, i.e., predicted to be a likely culprit of a future false rejection or false acceptance. Finally, we train Pixie on the 9 other folds ($F_2$, ..., $F_{10}$) and test it

| Pixie | CBFilter | FAR(%) | FRR(%) | F-measure(%) | EER(%) |
|---|---|---|---|---|---|
| MLP | MLP | 0.07 | 6.34 | 95.54 | 2.97 |
| MLP | RF | 0.06 | 4.70 | 96.52 | 1.87 |
| MLP | C4.5 | 0.02 | 7.19 | 95.92 | 2.35 |
| MLP | SVM | 0.10 | 9.83 | 93.08 | 4.87 |
| RF | MLP | 0.02 | 7.64 | 95.63 | 2.72 |
| RF | RF | 0.01 | 5.74 | 96.77 | 1.96 |
| RF | C4.5 | 0.02 | 7.19 | 95.92 | 2.35 |
| RF | SVM | 0.02 | 10.74 | 93.90 | 3.82 |

Table 4.10: Pixie + CBFilter performance, for various combinations of supervised learning algorithms. CBFilter is effective: when using RF, it reduces the EER of Pixie (with MLP) to 1.87%.

| Algo | FAR(%) | FRR(%) | F-measure(%) |
|---|---|---|---|
| Pixie | 0.10 | 9.83 | 93.08 |
| Pixie & RBFilter | 0.04 | 5.25 | 96.58 |
| Pixie & CBFilter | 0.06 | 4.70 | 96.52 |
| Pixie & RBFilter & CBFilter | 0.02 | 4.25 | 97.52 |

Table 4.11: Filters effects on Pixie performance. The combination of RBFilter and CBFilter (RF) has the best performance.

on the filtered $F_1$. We repeat this process 100 times (for the 10 folds of each of the 10 subsets of the Pixie dataset).

Table 4.10 compares the performance of various classifiers for both Pixie and CBFilter. It shows that CBFilter is effective: when using RF classifier, it reduces the EER of Pixie to 1.87% (from 4.87%), and removes 3.45 reference sets on average from a testing fold.

**Pixie, RBFilter and CBFilter**. When used in combination with RBFilter, CBFilter removes an additional 0.9 reference sets on average from a testing fold. RBFilter's candidate rule also removes 79.59 instances. Table 4.11 compares the performance

| Attack Dataset | FAR(%) |
|---|---|
| Google | 0.054 |
| ALOI | 0.087 |
| Caltech101 | 0.042 |

Table 4.12: Performance of Pixie (with RBFilter and CBFilter) on the ALOI, Caltech101 and Google attack datasets: On more than 14M attack authentication samples, the FRR of Pixie is less than 0.09%.

of the combined Pixie, RBFilter and CBFilter against the performance of the unfiltered Pixie, as well as Pixie's combination with only one of the filters. When used together, the filters reduce the FAR of the basic Pixie by 80% and its FRR by 56%.

**Comparison to other authentication methods**. The performance of Pixie (EER=1.87) compares favorably with the performance of other biometric based authentication solutions. For instance, Meng et al. [MWFZ15] report EERs of 2-4% and 2-6% for authentication solutions based on face and fingerprint. Samangouei et al. [SPC15] report EERs of 13-30% for attribute based face authentication, and Taigman et al. [TYRW14] report an EER of 8.6% for face recognition using features extracted by deep neural networks. The gaze-challenge authentication solution of Sluganovic et al. [SRRM16a] has an EER of 6.3%, while Zhao et al. [ZFSK14] report EERs between 4.1-9.6% for touch gesture based authentication.

### 4.5.2 Pixie Under Attack

We use the Google Image based attack dataset (see § 4.4) for a shoulder surfing attack, and, along with the ALOI and Caltech101 datasets, to evaluate brute force pictionary attacks.

We investigate the performance of Pixie, trained on one of the 10 Pixie dataset subsets, under the attacks of § 4.2.3. We use the previously identified parameters:

the ORB keypoint extractor, MLP for the Pixie classifier, RF for the CBFilter classifier, and all the rules for RBFilter. We do not cosider UBounds filter as by using UBounds filter we obtain a conservative performance of Pixie: with UBounds, Pixie would easily reject out of bounds images, artificially boosting its accuracy.

### Pictionary Attack

Under the Google Image attack dataset, Pixie achieved a FAR of 0.054%, see Table 4.12. 216 of the 350 trinkets were not broken. However, we counted each such trinket as success at $7,853$ trials (i.e. the total number of attack images). Then, the average number of Google dataset based "trials until success", over the 350 trinkets is $5,766.12$. For the ALOI based attack, when using both RBFilter and CBFilter, Pixie achieved a FAR of 0.087%. Under the Caltech101 attack, Pixie's FAR is 0.042%. The higher FAR of the ALOI pictionary attack dataset may be due to the similarity of its images of small objects to images in the Pixie datase. Pixie filters about 10 reference sets from each attack dataset. In addition, it filters a small number of candidate images (82 and 5) from the Google and Caltech101 datasets, but 1,449 candidate images from the ALOI dataset.

### Restricted Shoulder Surfing Attack

We use the Pixie and Google Image datasets to evaluate the "guessing entropy" [DMR04] of the restricted shoulder surfing attack. The attack proceeds as follows: for each reference set of a Pixie dataset trinket, we re-order the Google dataset images to start the brute force attack with images of the same type as the trinket. We then use each image in the re-ordered Google dataset as candidate, and count the number of trials before a match (false accept) occurs. Thus, this experiment evaluates the scenario where the adversary exploits his knowledge of the trinket type.


Figure 4.10: Example master images for Pixie: each of these images matches multiple reference sets of the Pixie dataset. Master images tend to have a rich combination of shapes, shadows, colors and letters.

As in the pictionary attack above, we counted each of the 216 unbreakable trinkets as "success" at $7,853$ (the size of the attack dataset) trials. Then, the average number of "trials until success", over the 350 Pixie dataset trinkets was $5,639.53$. This result is similar to the above pictionary attack: in fact, an unpaired t-test did not find a statistically significant difference in the number of trials to break a reference set between the two scenarios ($p - value = 0.44$, for $\alpha = 0.05$). Thus, in our experiments, knowledge of the trinket type does not provide the adversary with a significant guessing advantage.

### The Master Image Attack and Defense

We identified 788 master images in the ALOI dataset, 75 in the Caltech Image dataset, and 127 in the Google dataset. Master images match multiple Pixie reference sets. Upon manual inspection, we observed that master images are not of the same type of trinket as the reference set that they match. Instead, they contain an array of shapes, shadows, letters and colors, that translate into a diverse sets of keypoints, see Figure 4.10 for examples. Less than half of the master images in the ALOI (224), Caltech101 (34) and Google (30) datasets match at least 5 reference

sets. 1 master image in the Caltech101 dataset matches 51 reference sets.

**Defense**. The shape formed by the matched keypoints in a master image is likely to be inconsistent with that of the "victim" reference set. We leverage this observation to introduce several new features: the distance to the centroid of the matched keypoints (DTC-MKP) in the candidate and template images, and the minimum, maximum and mean of the DTC-MKP over all pairs of candidate and reference images. We train the Pixie classifier using this enhanced feature set, and test it on the ALOI attack dataset. The enhanced Pixie reduces the number of effective ALOI master images (matching at least 5 reference sets) by 60%, i.e., from 224 to 88.

To evaluate the effect of the new features on the FRR, we run Pixie with both RBFilters and CBFilters on the 10 Pixie data subsets (see § 4.4) in a 10-fold cross validation experiment similar to that of § 4.5.1. We observed that when new features are included in the classification task, the FRR of Pixie decreases slightly from 4.25% (last row in Table 4.11) to 4.01%, while its FAR remained unchanged (0.02%). We conclude that the newly added features do not increase Pixie's FRR.

## 4.6 User Study

We have used a lab study to evaluate the usability of Pixie's trinket based authentication and compared it against text-based passwords. In this section, we describe the methodology and results.

### 4.6.1 Design and Procedure

We performed a within-subjects study, where all the participants were exposed to every conditions considered. Specifically, the conditions were to authenticate from a smartphone to the Florida International University Portal Website (MyFIU), using

(i) their username and text-based password and (ii) their Pixie trinket. MyFIU is a site that provides students with information about class schedules and administrative functionality.

We have recruited participants from the university campus over e-mail lists, bulletin boards and personal communications. All the participants were students enrolled at the university. The reason for selecting students for the study was to ensure a consistent and familiar login procedure to MYFIU remote service. The participants in our study achieved text password authentication times on par with previously reported results (see § 4.6.2). Considering the ubiquity of mobile devices, we believe that the participants had no unfair advantage when compared to other social groups of similar age, with respect to their ability to perform the basic action of snapping a picture with a smartphone.

In the following, we first present some demographic information about the (n=42) participants in this study, then describe the procedure we used to perform the user study.

**Demographics.** We have recruited 42 participants for our lab study. Table 4.13 shows the demographics of the participants, obtained through the study questionnaires. In addition, 41 (98%) participants said they use their phones to login to their online accounts.

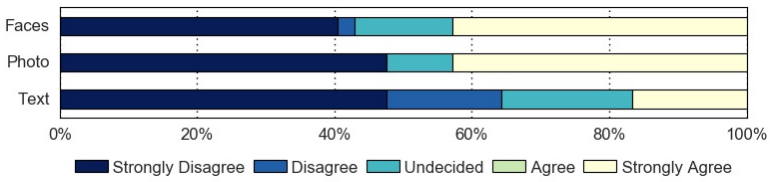
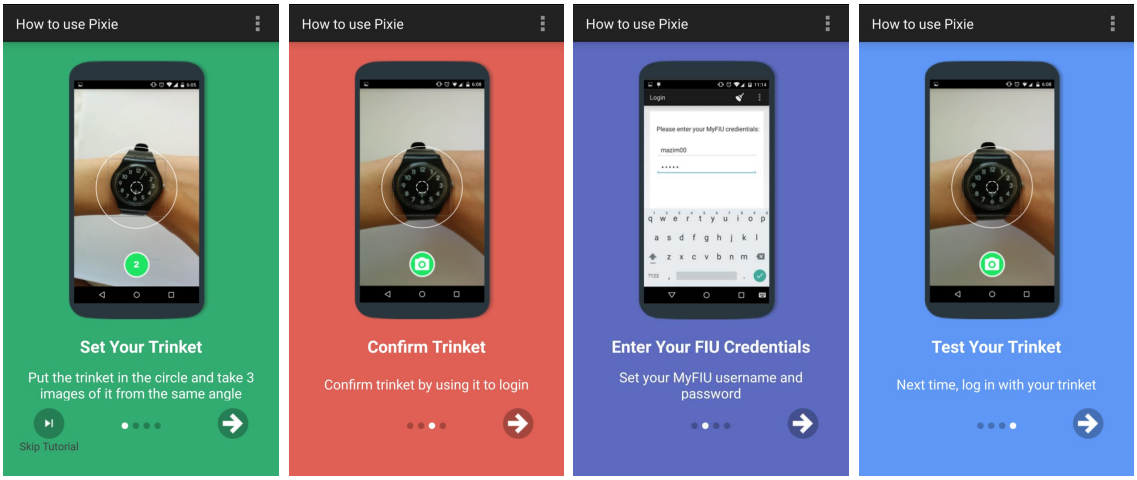



Figure 4.11: Pre-study level of agreement of the participants with ease of remembering faces, photos and text. 42% of the participants strongly agree to their ease of remembering photos and faces vs. only 16% who agreed it is easy for them to remember text.

| Demographic | Number | Proportion (%) |
|---|---|---|
| **Gender** | | |
| Female | 11 | 26 |
| Male | 31 | 74 |
| **Age** | | |
| Min | 18 | |
| Max | 50 | |
| Median | 28 | |
| Android | 20 | 48 |
| iPhone | 21 | 50 |
| Windows phone | 1 | 2 |
| Undergraduate | 16 | 38 |
| Graduate | 26 | 62 |
| CS/IT | 38 | 90 |
| Other majors | 4 | 10 |
| Use phone to login to remote services? | 41 | 98 |

Table 4.13: Participant demographics. We chose only students in order to have a consistent experience for remote authentication (on the university portal website, MyFIU).

Prior to the study, we also asked the participants to express their level of agreement using a 5-point Likert scale (from 1-strongly disagree to 5-strongly agree) with how easy it is for them to remember text, photos and faces. Figure 4.11 shows the summary of the participants responses. More than 42% of the participants strongly agreed that it is easy for them to remember faces and photos. However, only 16% of the participants strongly agreed it is easy for them to remember text. While 64.29% of the participants said it is not easy for them to remember text, a lower 47.62% and 42.86% of the participants said it is not easy for them to remember photos and faces respectively. A pairwise non-parametric Wilcoxon-Mann-Whitney test revealed no significant difference between the perceived memorability for different items. Based on this analysis and given the picture superiority effect [NRW76],




(a) (b) (c) (d)

Figure 4.12: Pixie in-app instructions (best viewed in color), showing how to (a) setup a trinket, (b) confirm the trinket, (c) enter credentials for the MyFIU account the first time the app is used, and (d) login using the trinket.

we posit that memorizing trinkets and their secret angles could be perceived to be as memorable as faces and text. We compare the perceived memorability of trinkets and text passwords in § 4.6.2.

**The study procedure.** We have conducted the study in an indoor lab using the existing artificial lighting. For the authentication device, we have used an HTC One M7 smartphone (1.7GHz CPU, 2.1 MP camera with f/2.0 aperture, 4.7 inch display with 1920 × 1080 resolution, and 137.4 × 68.2 × 9.33mm overall size).

The study consisted of 3 sessions, taking place on day 1, day 3 and day 8 of the experiment. From the total of 42 participants, 31 participants returned for and completed session 2 (7 female). Due to scheduling constraints, 3 participants returned for session 2 on day 4 or 5. 21 participants returned for and completed session 3 (4 female). The lab sessions proceeded as follows.

In the first session, we briefed participants about the purpose of the study: to explore the usability and the user interface design of a mobile device application. Then, we asked them to use Pixie to login to their MyFIU account, using their

credentials (username and password). Pixie associates the text credentials with the trinket's reference images. During subsequent login sessions, the users only needed to correctly capture the image of their trinket in order to access their account. Our goal was to let the participants experience Pixie for authentication, thus we did not ask them to enter their text password in subsequent sessions. As a result, the comparison of Pixie with text passwords is based only on the data collected in session 1.

Subsequently, the first session consisted of 3 steps. In the *discoverability step*, we gave no verbal instructions to participants. Instead, we asked each participant to try to figure out how to use Pixie, given only the in-app instructions, that show a watch as a trinket example. Figure 4.12(a-d) shows snapshots of Pixie app instructions for setting up a trinket, verifying the trinket, setting up the MyFIU account when the app is used for the first time and login step using trinket.

In the *training step*, we explained Pixie's purpose and walked the participant through the process of setting and testing a trinket using a gum pack. However, we neither justified why we chose this trinket, nor specified what other objects can be used as trinkets. We then asked the participants to set a trinket for the rest of the study.

In the third, *repeatability step* we asked the participant to repeat the login part of the process. To avoid input based on muscle memory, we distracted the participant's attention between the second and third step by playing a game for 5 minutes.

In session 2 and 3, the participants were asked to login to their MyFIU account with the trinket they chose in session 1. At the end of each session, the participants filled out questionnaires that use Likert scales (ranging from 1-strongly disagree to 5-strongly agree). The questionnaires evaluate Pixie and compare it against text-based passwords on perceived security, ease of use, memorability and speed dimensions.

In addition, at the end of session 3, we have used "emocards" [DOT01] to evaluate the emotional responses of users toward Pixie and text password authentication. Emocards are 16 cartoon faces, each representing one of 8 distinct recognizable facial expression (1 per gender). Emocards assist users to non-verbally express their emotions about products, in terms of pleasantness (pleasant, neutral, unpleasant) and arousal (calm, average, excited), two commonly accepted dimensions of emotion responses [Rus80].

**Participant dropout**. The participant drop from session 1 to session 3 is not due to a dislike of Pixie. To conclude this, we have compared the distributions of the answers of the 21 participants who dropped and of the 21 participants who stayed until session 3, on their overall impression of Pixie and their willingness to adopt it. Both questions were rated on a Likert scale. The Mann-Whitney test shows that the difference between the two populations is not statistically significant ($p = 0.7532$ for the first question, and $p = 0.0701$ for the second question at $\alpha = 0.05$). The participant drop can be due to the difficulty of scheduling 3 sessions across 8 days, at the end of the semester.

**Ethical considerations**. We have worked with our university Institutional Review Board to ensure an ethical interaction with the participants during the user study. We have asked the participants to avoid choosing sensitive trinkets. The entire experiments took around 40 minutes per participant. We compensated each participant with a $5 gift card.

### 4.6.2 User Study Results

Pixie is a novel authentication solution. Thus, we first present insights from its use across the 3 sessions, with a focus on discoverability. We then detail Pixie's observed

memorability and performance, as well as the participant perception and emotional responses. All the statistical tests performed in this section used a significance level of $\alpha = 0.05$.

### User Experience

We now detail the user experience across the 3 sessions.

**Session 1: discoverability**. Without previous knowledge of Pixie, 86% of the participants (36) were able to correctly set up their trinkets. Therefore, Pixie's Failure to Enroll (FTE) rate is 14%. From the 14% (6) participants who failed to enroll, 3 did not notice that the 3 trinket photos had to be of the same object, captured from similar angles. While Pixie provides a tooltip on the trinket capture button that guides the user to take another picture of the trinket when the app is used for the first time (see Figure 4.1(a)), these 3 participants took random pictures from different objects in the lab. These participants also did not understand the meaning of several words, as English was their second language:

> [P20]: *"Include one page saying what the trinket is. Like [sic], you can say that trinket is an object that you will be using to sign in to your account"*.
>
> [P21]: *"I don't understand what plain texture means"*.

In all 3 cases, the Pixie prefilters identified the issue correctly. The other 3 unsuccessful participants chose trinkets with a plain texture (e.g., palm of hand, pencil, objects with plain black surface) that generated errors. They either dismissed

error messages quickly or were not sure what to choose as a trinket to eliminate the errors.

Subsequently, 3 other participants were unable to perform the trinket verification step within 3 trials. This occurred due to (i) bad lighting conditions around the trinket, (ii) the participant forgetting the trinket angle, or (iii) a texture-less (plain) trinket. While the Chi-square test did not identify significant differences in the error rates caused by any of the aforementioned circumstances ($p > 0.05$), this could be because of the limited number of samples.

**Session 1: Training**. All the participants were able to set up a trinket successfully, reducing the FTE rate of Pixie from 14% in the discoverability step to 0%. All the participants then tested their trinkets within 4 trials (M = 1.29 trials, Std = 0.6): 76% of the participants were able to login from the first trial. The other 24% had lighting related difficulties (e.g., the trinket reflected the light, or was in the shadow). Only one participant required 4 trials.

**Session 1: Repeatability**. All the participants except one, were able to successfully complete this step within 3 trails (M = 1.29 trials, Std = 0.6). One participant required 4 trials.

**Sessions 2 and 3**. In session 2, 84% of the participants were able to login from the first trial, 13% logged in within 2-3 trials and only one participant needed 6 trials (M = 1.35 trials, Std = 1.02). 2 participants did not carry their trinkets and had to reset them. In session 3, 81% of the participants were able to login from the first trial and all the other participants were able to login within 2-3 trials (M = 1.20 trials, Std = 0.40).

### Participant Performance

To measure participant performance we use *success rate* [CFS+09], defined as the number of successful attempts to the total number of attempts. In order to compare the success rate of participants for text-based passwords and Pixie, we analyzed the data from either of the login recalls of each session. We only consider successful Pixie authentication sessions within 3 trials (see § 4.6.2). This is similar to MyFIU, where the participants need to reset their passwords after 3 unsuccessful trials. The success rate of Pixie improves from session 1 (82.00%) to session 2 (83.33%) and session 3 (84.00%). Throughout all the 3 sessions, the Pixie success rate for successful authentication sessions is slightly lower than the success rate for the text-based password in session 1 (88.10%). This is not surprising, given the significantly lower number of practice opportunities for Pixie, compared to the ubiquitous text passwords. However, the Chi-square test revealed no significant difference between the success rate for Pixie and text password in session 1 ($\chi^2(1) = 0.506, p = 0.48$). Similarly, the Wilcoxon-Mann-Whitney test found no significant difference in terms of the number of attempts for a successful login for Pixie within different sessions, and between Pixie and text-based password in session 1.

### Memorability

During session 2, 96% of the participants (all except one) were able to remember their trinkets. 2 participants did not immediately recall the part of the trinket they used to authenticate, but they figured it out in the 3rd attempt. These 2 participants were able to login in the first attempt in the 3rd session. 2 participants did not carry their trinkets and had to reset them in session 2. During session 3, all the participants were able to remember their trinkets. We contrast these results with the memorability of text passwords: 5 participants did not remember their

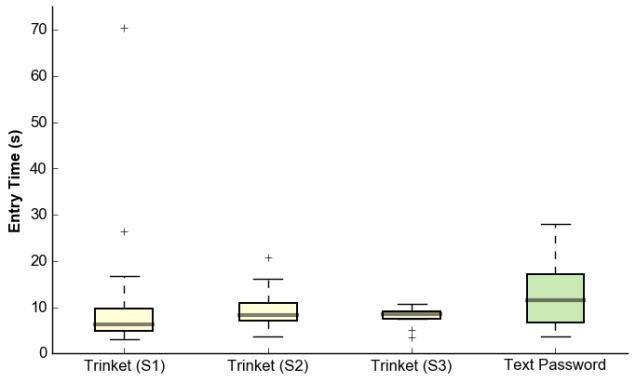



Figure 4.13: Box plot for entry time of Pixie across 3 sessions vs. text password in session 1. The Wilcoxon-Mann-Whitney test revealed that Pixie's entry time in each session was significantly less than the entry time for text passwords. For a single participant, the Pixie entry time was 70.51s during session 1.

MyFIU password and had to reset it in the first session. This is consistent with previous findings: Wiedenbeck et al. [WWB+05b]) report that more than 17% of text-based passwords are forgotten in one week.

### User Entry Time

We have measured the *user entry time*, the interval from the moment when a user starts Pixie and when Pixie submits the captured photo to the authentication module. Figure 4.13 shows the box plot of the user entry time for Pixie in different sessions vs. the time for text passwords, during session 1. The shortest authentication session was 3.01$s$ and the longest session was 70.51$s$ for Pixie. The average entry time improves from session 1 (M=9.71s, Std=11.42s, Mdn=6.24s), to session 2 (M=9.71s, Std=4.66s, Mdn=8.32s) and session 3 (M=7.99s, Std=2.26s, Mdn=8.51s). However, Wilcoxon-Mann-Whitney tests did not reveal any statistically significant differences between the Pixie user entry time across the 3 sessions. We expect however that additional practice can further improve Pixie's entry time.

Moreover, a Wilcoxon-Mann-Whitney test revealed that the entry time for Pixie

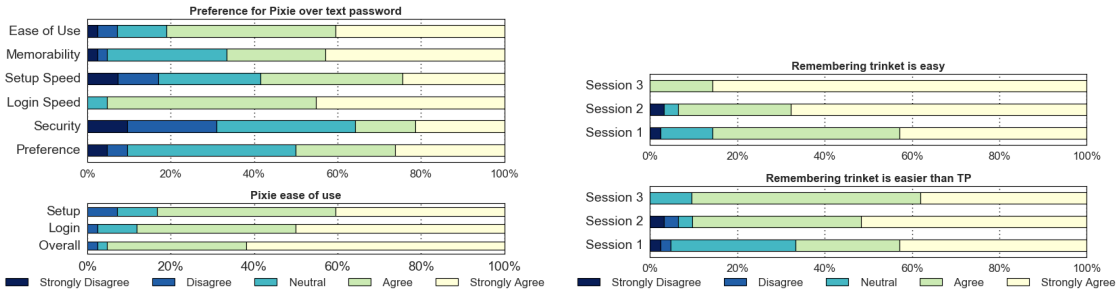
was significantly less than the entry time for text passwords in session 1 ($W = 845.0, p = 0.000$). We emphasize that in contrast to text passwords, Pixie participants did not have the opportunity to practice beyond the steps of the above procedure.

Table 4.1 compares the entry time for Pixie and other authentication solutions based on biometrics or text and graphical passwords. Although Pixie's entry time is higher compared to solutions based on face or voice, it compares well to several other solutions. For instance, Shay et al. [SKD+14] report an entry time of 11.6-16.2 for text passwords. MyFIU passwords are similar to the comp8 category in [SKD+14] (at least 8 characters, and include a lowercase English letter, uppercase English letter, and digit) for which [SKD+14] report a median entry time of 13.2s. The additional safeguards of Boehm et al.'s [BCF+13] face and eyes based biometric solution result in an entry time of 20-40s. Chiasson et al. [CFS+09] report an entry time of about 15s for Passpoints. Trewin et al. [TSK+12] reported an entry time of 8.1s for gesture (stroke) based biometric. The eye tracking solution of Liu et al. [LDGW15] requires 9.6s and the audio or haptic based solution of Bianchi et al. [BOK11] requires $10.8 - 20.1$s.

In addition, we evaluated the processing overhead of Pixie: the time required to decide if a candidate image matches the reference set. The average processing overhead of Pixie on the HTC One smartphone over 94 successful authentication trials is 0.5 seconds.

### Perception

We asked the participants to express their perception about Pixie and text passwords by providing answers to a set of questions in a 5-point Likert scale (from strongly agree to strongly disagree). In the following we presents the participants response.




(a) (b)

Figure 4.14: (a) Results at the end of session 1. (a - top) Perceived performance of Pixie compared to text passwords. Pixie dominates on ease of use, memorability and speed dimensions. (b - bottom) Pixie ease of use: 95% of participants agreed that Pixie is easy to use. (b - top) Pixie perceived memorability. 86% of participants agree that the trinkets are easy to remember after session 1, but reach consensus after session 3. (b - bottom) Perceived memorability of Pixie vs. text passwords (TP). No participant believes text passwords are more memorable after session 3.

In session 1, 81% of the participants agree that overall, Pixie is easier to use than text-based passwords (Figure 4.14(a) (top)). 83% and 86% of the participants agree or strongly agree that trinket setup and login steps are easy (Figure 4.14(a) (bottom)). 95% of participants agree or strongly agree that overall, Pixie is easy to use.

Furthermore, 86% of the participants agree or strongly agree that trinkets are easy to remember, see Figure 4.14(b) (top). 67% of the participants agree that trinkets are easier to remember than passwords, while only 5% of the participants believe the opposite, see Figure 4.14(a) (top) and Figure 4.14(b) (bottom). These results improve in sessions 2 and 3. At the end of session 3, all the participants agree that trinkets are easy to remember (Figure 4.14(b) (top)): 12 participants changed their opinion in favor of Pixie's memorability. No participants believe that text passwords are easier to remember than trinkets, see Figure 4.14(b) (bottom). A two-sample proportion test revealed that the proportion of the participants who

think Pixie is memorable, significantly increases from session 1 to session 2 and 3 ($Z = 2.36$, $p = 0.009$ and $Z = 2.05$, $p = 0.020$).

36% of the participants believe that Pixie is more secure than text passwords, and 31% of the participants believe that passwords are more secure (Figure 4.14(a) (left)). Several participants felt strongly about the security of Pixie, e.g.,:

> [P27] *"This method is even more secure than text-based passwords, because even if someone sees me during the password entry, he wouldn't know what part of the object I have selected as my trinket and cannot easily figure it out".*

68% of participants agree or strongly agree that the trinket based login is fast. 74% of participants agree or strongly agree that the trinket setup step is fast. 95% and 59% of the participants agree that Pixie's login and trinket setup steps are faster compared to the corresponding text password operations. (Figure 4.14(a) (top)). 50% of the participants say that they prefer trinkets over text passwords (Figure 4.14(a) (top, bottom bar)).

When asked if they would use trinket based authentication in real life 26% of participants said that they would use Pixie for most of their accounts, 36% would use it for at least some of their accounts, and 36% would consider using it. Only 2% of the participants (1) said that they would not use it. Several participants felt strongly about adopting Pixie:

> [P18]: *"Why isn't [Pixie] integrated with the original MyFIU mobile application as another option for signing to my account?".*

[P40]: *"I always forget my passwords [...] I always store them in my browser. I would definitely use Pixie if it is available".*

[P27]: *"I think this is a good method because I usually forget my passwords for my accounts".*

While we did not include survey questions on trinket availability, one participant asked:

[P8]: *"What if I do not wear the same watch everyday?".*

Other participants suggested to use multiple trinkets to ensure trinket availability:

[P21]: *"That would be good if we could set multiple trinkets and use any of them to authenticate".*

**Statistical analysis**. To differentiate true choice from random chance, we combine the strongly agree and agree answers into an "agreement" answer, and the strongly disagree and disagree answers into a "disagreement" answer. We then use a one-sample binomial test with a confidence interval in order to test whether the proportion of agreement of the participants with a statement is sufficiently different from a random choice (50%). Table 4.14 presents this result for the proportion of "agreement" answers to each question. Pixie is perceived easier, more memorable and faster than text passwords for login and the perceived advantage is not due to random choice. However, the participants did not perceive a significant difference in the setup speed and the security of Pixie over text passwords.

| Question | Sample proportion | 95% CI | $p$ |
|---|---|---|---|
| **Easier to use** | 80.95 | (65.88, 91.39) | 0.000* |
| **More memorable** | 66.67 | (50.45, 80.43) | 0.044* |
| **Faster Login** | 95.24 | (83.83, 99.41) | 0.000* |
| **Faster Setup** | 58.54 | (40.96, 72.27) | 0.441 |
| **More secure** | 35.71 | (21.55, 51.97) | 0.088 |

* Statistically significant result at $\alpha = 0.05$.

Table 4.14: Confidence interval for the proportion of "agreement" answers to usability and security questions comparing Pixie and text-based authentication. Pixie is perceived to be easier to use, more memorable and faster than text passwords. Pixie's perceived advantage in ease of use, memorability, and login speed is not due to random choice.

### Analysis of User Feedback

Table 4.15 shows that the general preference of Pixie over text passwords significantly correlates positively with its preference on ease of use, memorability and security and speed dimensions. The preference over text passwords is also significantly correlated with overall perception of trinket memorability and willingness to adopt Pixie. Interestingly, we observed a significant correlation between preference over text passwords on security and the participant feeling of owning a unique trinket ($\tau = 0.36, p = 0.005$).

The participant willingness to use Pixie also correlates positively with perceived memorability ($\tau_b = 0.29$), perceived ease of use ($\tau_b = 0.28$), general preference over text passwords ($\tau_b = 0.32$), preference over text passwords on security ($\tau_b = 0.28$), and preference on ease of use ($\tau_b = 0.04$). We observe a negative correlation between the willingness to use Pixie and the number of login attempts ($\tau_b = -0.16$), highlighting the impact of unsuccessful logins. However, the correlations are not statistically significant.

| **Prefer Pixie over text passwords** | $\tau_b$ | $p$ |
|---|---|---|
| **Easier to use** | 0.60 | 0.000* |
| **More memorable** | 0.54 | 0.000* |
| **More secure** | 0.38 | 0.003* |
| **Faster Setup** | 0.46 | 0.000* |
| **Faster Login** | 0.48 | 0.000* |
| **Pixie Memorability** | 0.40 | 0.003* |
| **Willingness to Use Pixie** | 0.60 | 0.000* |

* Indicates a statistically significant correlation at $\alpha = 0.05$.

Table 4.15: Kendall's Tau-b test shows significant positive correlation between preference of Pixie vs. text passwords, and its preference in terms of ease of use, memorability, security, faster setup and login time. Preference over text passwords is also significantly correlated with the overall memorability of the trinket and willingness to adopt Pixie.

### Emotional Response

The emocard experiment revealed that Pixie generates only positive emotions: 81% of the participants reported a "pleasant", and 19% reported a "neutral" experience. In addition, 47% of the participants were "calm", 34% were "average" and 19% were "excited". In contrast to Pixie, only 5% of the participants (1) reported a "pleasant" level for text passwords, while 57% reported "unpleasant" and 38% reported "neutral" levels. A one-sided test of the difference of proportions revealed that the proportion of the participants who perceived Pixie as pleasant was significantly larger than the proportion of the participants who perceived text passwords as pleasant ($Z = 4.01$, $p = 0.000$).

The Kendall's Tau-b correlations plotted in Figure 4.15 shows that the participant reports of willingness to use Pixie correlate positively with levels of pleasure and excitement, as well as Pixie's perceived ease of use. While 4 participants reported excitement for Pixie's novelty, functionality and performance, we observe

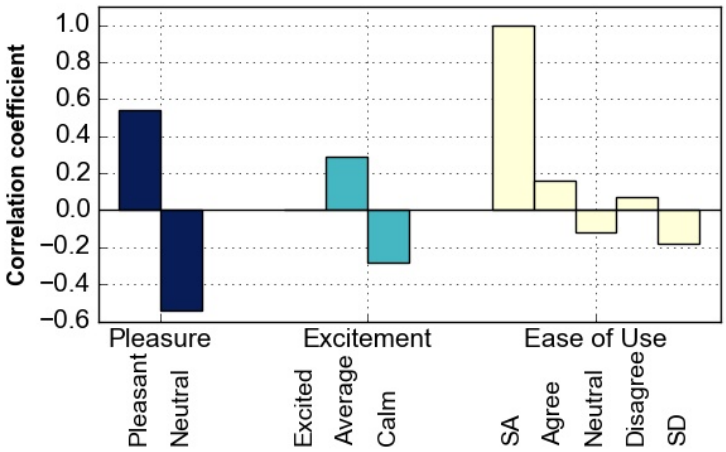



Figure 4.15: Kendall's Tau-b correlations between willingness to use, emotional responses (pleasure and excitement), and ease of use (SA/SD = Strongly Agree/Disagree), during session 3. No participant rated Pixie as unpleasant. Willingness to use correlates positively with pleasant and average levels, as well as with agreement with ease of use.

no correlation between "excited" levels and willingness to use. This is a positive finding, as authentication solutions should not generate high arousal levels.

### Trinket Choice

We manually analyzed the trinket images captured by the participants in the first session (42 trinkets) and those captured by the participants who reset their trinket in session 2 (2 trinkets). We allowed the participants to pick any nearby object as a trinket. The 42 participants picked a total of 36 unique trinkets, from 31 unique objects of 18 types, chosen from among participant owned objects and lab objects. The gum pack and watch were the most frequently chosen object types. However, all the 6 watch trinkets were different, and the 16 participants who chose a gum pack have captured 8 unique trinket images (object + angle combination).

In the discoverability step, 8 participants used their watches as trinkets. We did not observe a significant difference in user choice of trinket between the discoverabil-

| Object type | # of unique objects | # of participants | # of unique trinkets |
|---|---|---|---|
| Gum pack | 3 | 16 | 8 |
| Watch | 6 | 6 | 6 |
| Mug | 3 | 3 | 3 |
| Logo | 2 | 2 | 2 |
| Keychain | 2 | 2 | 2 |
| Car remote control key | 2 | 2 | 2 |
| Sunglasses | 2 | 2 | 2 |
| A piece of puzzle | 1 | 1 | 1 |
| Shoe | 1 | 1 | 1 |
| Kohl container | 1 | 1 | 1 |
| Backpack pin | 1 | 1 | 1 |
| Hair clip | 1 | 1 | 1 |
| Cigarette box | 1 | 1 | 1 |
| Match box | 1 | 1 | 1 |
| Water Bottle | 1 | 1 | 1 |
| iphone menu | 1 | 1 | 1 |
| University ID card | 1 | 1 | 1 |
| Tattoo | 1 | 1 | 1 |
| **Total** | **31** | **44** | **36** |

Table 4.16: Trinket choice: object types chosen by participants, along with the number of unique objects belonging to each category and number of unique trinket choice (object + angle) in the study. The gum pack and watch (used in the training step and on-screen instructions) are the types most frequently used by the participants. All the captured watch trinkets are unique.

ity and training steps: 18 participants used the same trinket in the discoverability and training steps. 8 participant chose their trinket to be their watches. The other trinket categories chose by participants that are not among those in Table 4.16 include: pen/pencil, book and computer mouse.

We have used the images captured by the participants to "brute force" the reference sets of each participant. We removed 8 reference sets as they were identical (the top view of the same gum pack). This has produced a single "success" event, for the two participants who chose the same side of the same gum pack, with very similar angles. As we described previously, the participant preference of Pixie over text passwords on security correlates significantly with the participant feeling of

owning a unique trinket. We did not observe a statistically significant difference between the feeling of owning a unique trinket and participants gender.

## 4.7 Discussion and Limitations

**Authentication speed**. Our user study shows that Pixie's authentication speed in session 1 is 25% faster than well rehearsed text passwords and improves through even mild repetition. However, Pixie's entry time is longer than the reported entry time for face based authentication solutions (see Table 4.1). This may be due to either the novelty of Pixie or the way the images are captured, i.e. using the back, not the front camera for capturing trinket images.

**Secure image storage and processing**. The storage and processing of the trinket images needs to be performed securely. While outside the focus of this chapter, we briefly discuss and compare trinket image storage and processing solutions that are performed on the remote service vs. the user's authentication device. A remote server based solution trivially protects against an adversary that captures the authentication device, as the device does not store or process sensitive user information. The image matching process is also faster on a server than on a mobile device (see § 4.5.1). The drawbacks are the overhead of transmitting candidate images over the cellular network, and the imposition on users to register a different reference image set for each remote service.

The authentication device based solution can easily associate the reference images with the user's authentication credentials (e.g. OAuth [DH12]) for multiple remote services. However, since an attacker can capture and thus access the storage of the mobile device, reference images cannot be stored or processed in cleartext. The storage and processing of reference images can however be secured

through hardware-level protection, e.g., TrustZone [Tru17], or by using privacy preserving image feature extraction solutions that work in the encrypted domain, e.g., [WHWR16, QYR+14, HLP12]. In Chapter 5, we introduce ai.lock, an image-based authentication solution similar to Pixie, that alleviate the problem of secure storage of credentials.

**Deployability**. Pixie is well suited for OAuth [DH12] authorization to access remote services from the mobile device: Pixie authenticates the user to the app on the mobile device, which can then proceed with the OAuth protocol with the remote server.

**Default authentication**. If the trinket based authentication fails a number of times (due to e.g., forgotten trinket, poor lighting conditions, unsteady hand), the user is prompted to use the default authentication solution, e.g., text password.

**Strong passwords**. Popular and ubiquitously available trinkets (e.g., iWatch, Coke can) should not be chosen as trinkets, as an adversary can easily predict and replicate them. To address this problem, Pixie can store a dataset of popular trinket images, then, during the trinket setup process, reject reference sets that match popular trinkets.

**Defense against brute force attacks**. The brute force attacks of § 4.2.3 can be made harder to launch through video "liveness" verifications, e.g., [MR16]: capture both video and accelerometer streams while the user shoots the trinket, then use video liveness checks to verify the consistency between the movements extracted from the two streams. The lack of such streams or their inconsistency can indicate a brute force attack.

**The user study**. The study presented in this chapter was the first attempt to quantify the usability aspects of an authentication solution based on trinkets. We performed the user study in a lab setting. We were able to recruit only 42 partic-

ipants, of which half did not stay until the last session. As we wanted to ensure a consistent and familiar login procedure to remote services for the participants, we only recruited students from the university who had access to myFIU, FIU's login portal. While the population of the study is not fully representative of the users who would use the system, we believe that the participants had no unfair advantage when compared to other social groups of similar age in performance: the participants in our study achieved text password authentication times on par with previously reported results (see § 4.6.2).

Pixie works by extracting invariant keypoints from the captured images, using keypoint extraction algorithms (e.g. SURF [BTVG06] and ORB [RRKB11]). These algorithms are not capable of extracting keypoints from images of object with constant shade. We attempted to address this issue by providing actionable feedback to users, and guiding them toward choosing visually complex trinkets. In addition, to ensure Pixie is able to identify the trinket images even when captured in slightly different circumstances and to lower the false reject rate, we required the users to enter 3 trinket images in the registration phase. This may partially explain why the participants in our study did not perceive Pixie as significantly faster than text passwords for the registration phase.

During the discoverability step, we observed that several participants had difficulties in understanding the in-app instructions on how to use Pixie. Similar problems have been reported for other authentication mechanisms. For instance, Bhagavatula et al. [BUI+15] reported that 7 out of 10 participants found understanding on-screen instructions difficult for iPhone fingerprint authentication. They recommend to provide clearer instructions (e.g. through a demo video) on what the users need to do. We posit that explaining the meaning of trinkets will help users during the registration phase and improve the discoverability rate of Pixie.

The consent form that we read and the participants signed prior to the study, emphasizes that the focus of the study is on the usability aspects of a new application. We observed that some of the participants might have selected trinkets without concerns over security during the study. We did not guide the participants towards choosing specific trinkets, as we intended to observe the personal or lab objects chosen by the participants. Nevertheless, we observed that the participants preference of Pixie over text passwords on security correlates significantly with the participants feeling of owning a unique trinket. This suggests that the participants could corroborate the relationship between unique trinkets and higher level of security.

The trinkets used to walk the participants through Pixie (i.e., gum pack) and the in-app user guide of Pixie (i.e., watch), appear to influence the participant trinket selection in the first session: in the discoverability step, 8 participant chose their watches as trinkets. In addition, during session 1, 9 participants chose the same gum pack as used in the Pixie walk-through without even trying a different angle, and 5 participants used their watches as trinkets. Further studies are required to understand whether other means of communicating the goals of Pixie (e.g. using a short video that guides the user on how to choose secure and unique trinkets) can reduce this bias.

Further, although 50% of the participants said they prefer Pixie over text passwords, 40% percent of the participants were undecided. This may be due to the limited experience of the participants with Pixie. In addition, 62% participants said they would use Pixie in real life. We did not observe a statistically significant correlation between being excited about using Pixie, that could be due to the novelty of the method, and willingness to use it. However, future studies are required to understand in what scenarios and situations the users are willing to adopt trinket based authentication or prefer it over text passwords.

If we were to redo the study, we would split the Likert scale questions comparing Pixie with text passwords into 2 questions asking the participants to rate any of them in terms of usability and security. In addition, we would ask the participants to justify their answers about perceived usability and security in the form of open ended questions in the post study interview.

**Real world limitations**. A comprehensive study similar to the studies conducted for biometric based solutions (e.g. [BUI+15]) may help identify potential limitations of Pixie in different situations. We discuss now two such situations.

- *Insufficient light.* In our studies, we observed that Pixie has problems with insufficient lighting. This is likely a problem shared also by face based authentication solutions. We took steps to partially address this problem, by designing image filters to identify the problematic images and provide users with actionable feedback. Note that low light photography is one of the major differentiators among mobile phone manufacturers. Newer devices are equipped with wider apertures which capture more light and optical stabilization which allow for longer exposures and thereby taking better low light photos. As mobile device cameras get more capable in capturing better low light photos over time, we expect it to be less of a problem for camera based authentication methods as well. Further, while we did not test this in our studies, we conjecture that using the camera's flash light to illuminate the trinket could also help address this problem. We leave it for future work to investigate better solutions, e.g., that leverage the ability of mobile device cameras to capture infrared light, to handle the case of reflective trinkets.
- *Unstable capture conditions.* Pixie may be harder to use in certain circumstances, e.g., while the user is walking or in public transportation, as movements might affect the quality of snapped photos. The ease of using Pixie in

such scenarios is likely to depend on the trinket object type, as for snapping a trinket image, the user needs one or both hands. Pixie has specialized filters to identify blurry images. However, a thorough evaluation of the filters in such scenarios will help identify the filter parameters that maximize usability.

**Changing and forgetting trinkets**. In contrast to biometric based authentication solutions, Pixie allows the participants to change their trinkets regularly as they change the clothes, accessories and objects that they carry. Despite the obvious security benefits of this property, people may forget to carry their trinkets, impacting Pixie's usability. The simplest solution to this problem is to fall back on default authentication using a standard approach (e.g., text password) then set a new Pixie trinket. Another solution is to allow Pixie users to have multiple trinkets, the additional trinkets could be chosen among frequently worn outfits or locations visited, alternatively there could be a single backup trinket which is known to be reliably accessible although may not be readily available such as an object kept at home or at work. This however impacts the security of Pixie, as it becomes easier to guess or brute force one of the trinkets.

Another approach is to use Pixie in conjunction with the security token concept, where the token is a printed visual token that displays a pattern (e.g., random art [PS99]). The user still needs to capture this token using Pixie, and should carry the token at all times, just like for a credit card or mobile device. Using a simple visual token is an alternative to using a QR code as trinket, e.g. [CHM15, KHX15]). In addition, Pixie does not require the additional hardware (e.g., magnetic strips and Near-field communication) used in automated access control solutions, and is thus applicable to a wider ranger of mobile and wearable devices.

We note that a survey about the gamut of objects that people carry with them as well as the variability of possession habits would enable further analysis regarding the impact of trinket changes and failure to recall.

**Shoulder surfing attack**. Choosing and using trinkets in public exposes users to shoulder surfing attacks. In this respect, Pixie is similar to ATM authentication: the user needs to be cognizant of the risks and make sure that there are no people around watching, before setting or using her trinket. We note that while personal privacy during authentication may protect Pixie users from shoulder surfing attacks, this does not hold for biometric authentication alternatives based on face and fingerprints. This is because face and fingerprints are almost public information, accessible to attackers in vulnerable social networks and government datasets. Further, we note that unlike the ATM and biometric authentication scenarios, Pixie allows the user to change her trinket once she is in a more private setting.

We have shown that Pixie is resilient to a shoulder surfing attack flavor, where the adversary learns or guesses the trinket object type and attempts to collect and use images of similar objects to brute force Pixie (see 4.5.2). We note that additional information about the trinket object or the objects owned by the user can increase the adversary chance to launch a successful attack.

In a shoulder surfing attack, the adversary still needs to capture the trinket, or obtain a copy of it to launch a successful attack. Since Pixie authentication requires a simple interaction with the user, it is also possible to combine Pixie with a token (cryptographic key) stored on the mobile device. This approach is similar to the concept of "protocredential" introduced by Corella and Lewison [CL15]. The combined Pixie and mobile device token authentication would require the user to possess both the particular mobile device that stores the token and the trinket. As an alternative, Pixie can be used in conjunction with biometric authentication solutions, e.g., [DLHB+12]: in touchscreen devices, one could use a touch gesture to mark the trinket, as an additional authentication factor.

**Field study**. We leave for future work a field study of Pixie to investigate the longer term effects of using trinket passwords on user entry times, accuracy and memorability, the factors that impact trinket choice, how users choose and change their trinkets in real life, as well as the potential improvements provided by alternative means of communication of Pixie's goals and functionality (e.g., through short video instead of text). We also leave for future work the investigation of using mental stories to associate trinkets to accounts (e.g., use credit card as trinket for bank account) and reducing the impact of interference [CFS+09, AAW15].

## 4.8 Conclusions

We introduced Pixie, a proof of concept implementation of a trinket based two-factor authentication approach that uses invariant keypoints extracted from images to perform the matching between the candidate and reference images. Pixie only requires a camera, thus applies even to simple, traditional mobile devices as well as resource limited wearable devices such as smartwatch and smartglasses.

We manually captured and collected from public datasets, 40,000 trinket images. We proposed several attacks against Pixie and have shown that Pixie achieved an EER of 1.87% and FAR of 0.02% on 122, 500 authentication attempts and an FAR of less than 0.09% on 14.3 million attack instances generated from the 40,000 images.

We performed an in lab user study to evaluate the usability aspects of Pixie as a novel authentication solution. Our experiments show that Pixie is discoverable: without external help and prior training, 86% and 78% of the participants were able to correctly set a trinket then authenticate with it, respectively. 62% of the participants expressed that they would use Pixie in real life. Pixie simplifies the authentication process: the study shows that trinkets are not only perceived as more

memorable than text passwords, but are also easily remembered 3 and 8 days after being set, without any inter-session use. In addition, Pixie's authentication speed in session 1 is 25% faster than well rehearsed text passwords and improves through even mild repetition. We believe that Pixie can complement existing authentication solutions by providing a fast alternative that does not expose sensitive user information.

The identified master images and the reference sets they match can be used (see § 4.5.2) to develop a new filter (e.g., based on supervised learning algorithms), that detects vulnerable reference sets, likely to be matched by master images. We leave this investigation for future work.

In addition, a promising approach to improve Pixie is to use more advanced image processing techniques, e.g. deep neural networks [SVI+16b], for image feature extraction and processing. Such techniques may improve Pixie's usability by (i) eliminating the requirement for capturing multiple reference images of the trinket in the registration phase, (ii) increasing the ability to extract features even from images of objects with constant shade, and (iii) further reducing FRRs. In Chapter 5, we investigate this direction.

CHAPTER 5

# AI.LOCK: IMAGE-BASED AUTHENTICATION WITH SECURE STORAGE OF CREDENTIALS

## 5.1 Introduction

Biometrics are widely used for authentication in consumer devices and business settings as they provide sufficiently strong security, instant verification and convenience for users. However, biometrics are hard to keep secret, stolen biometrics pose lifelong security risks to users as they cannot be reset and re-issued, and transactions authenticated by biometrics across different systems are linkable and traceable back to the individual identity. In addition, their cost-benefit analysis does not include personal implications to users, who are least prepared for the imminent negative outcomes, and are not often given equally convenient alternative authentication options.

In Chapter 4, we introduced Pixie, a secret image based authentication approach, where users authenticate to a remote service using arbitrary images they capture with the device camera. Pixie authentication solution can address several of the problems associated with biometric-based authentication method. For instance, using Pixie the authentication is no more tied to a visual of the user's body (e.g., face and fingerprint), but that of a personal accessory, object, or scene, i.e., the trinket.

However, Pixie has an important drawback when deployed on mobile devices (i.e. local authentication scenario of § 4.2.1): the image keypoints (features), extracted by Pixie, need to be stored and matched in cleartext on vulnerable devices.

**Challenges.** Local Image based authentication approach raises new challenges. First, an adversary who captures or compromises the device that stores the user's

reference credentials (e.g. mobile device, remote server) and has access to its storage, should not be able to learn information about the reference credentials or their features.

Second, while biometric features such as ridge flow of fingerprints or eye socket contours of faces, can be captured with engineered features and are invariant for a given user, images of objects and general scenes lack a well defined set of features that can be accurately used for authentication purposes. Improper features will generate (i) high False Accept Rates (FAR), e.g., due to non-similar images with similar feature values, and (ii) high False Reject Rates (FRR) that occur due to angle, distance and illumination changes between the capture circumstances of reference and candidate images.

In this chapter, we first introduce *ai.lock*, a practical, secure and efficient image based authentication system that converts general mobile device captured images into biometric-like structures, to be used in conjunction with secure sketch constructs [DRS04] and provide secure authentication and storage of credentials (see § 5.4). To extract invariant features for image based authentication, ai.lock leverages (1) the ability of Deep Neural Networks (DNNs) to learn representations of the input space (i.e., *embedding vectors* of images) that reflect the salient underlying explanatory factors of the data, (2) Principal Component Analysis (PCA) [F.R01] to identify more distinguishing components of the embedding vectors and (3) Locality Sensitive Hashing (LSH) [Cha02] to map the resulting components to binary space, while preserving similarity properties in the input space. We call the resulting binary values *imageprints*.

In a second contribution, we propose the LSH-inspired notion of *locality sensitive image mapping* functions ($\delta$-LSIM), that convert images to binary strings that preserve the "similarity" relationships of the input space, for a desired similarity

| Solution | FAR (%) | EER (%) | Estimated Entropy (bits) | Dataset size |
|---|---|---|---|---|
| ai.lock (MLMS) | 0.0004 | - | 18.02 | $2 \times 10^9$ |
| ai.lock (MLSS) | 0.0015 | 0.17 | 16.02 | $6 \times 10^6$ |
| iPhone TouchID [App17a] | 0.0020 | - | 15.61 | - |
| Deepface [TYRW14] (face) | - | 8.6 | - | $> 0.5 \times 10^9$ |
| SoundProof [KMSv15] (sound) | 0.1 | 0.2 | 9.97 | $> 2 \times 10^6$ |
| [SRRM16b] (eye movement) | 0.06 | 6.2 | 10.70 | $1,602$ |
| RSA SecurID [RSA17] | - | - | 19.93 | - |
| Text-based password [Bon12] | - | - | 10-20 | $7 \times 10^7$ |

Table 5.1: ai.lock variants vs. commercial and academic biometric, token-based authentication solutions, and text passwords. ai.lock MLSS variant has no false rejects, as it is evaluated under attack samples only. Under large scale datasets of powerful attacks, ai.lock achieves better entropy than state-of-the-art biometric solutions.

definition (see § 5.3). A $\delta$-LSIM function can be used to efficiently match images based on their extracted binary imageprints.

Unlike Pixie, ai.lock builds on a secure sketch variant [DRS04] to securely store reference imageprints and match them to candidate imageprints. ai.lock only stores a "hash" of the object's image (i.e., the imageprints). Furthermore, we show that on larger and more complex attack datasets, the use of DNNs enables ai.lock to achieve FAR that are at least 2 orders of magnitude smaller than those of Pixie ($\leq 0.0015\%$ vs. $0.02 - 0.08\%$), for similar FRRs ($4\%$).

**Implementation and evaluation.** We implemented ai.lock using Tensorflow [ABC+16] and Bose, Chaudhuri, and Hocquenghem (BCH) codes [Hoc59a, BRC60]. We then develop brute force image based attacks that aim to defeat ai.lock.

Particularly, to evaluate ai.lock performance under attacks, first we perform *real image attacks* using manually collected and publicly available image datasets. To evaluate ai.lock on large scale attack images, we develop *synthetic image attacks* that

use images produced by generative models [RMC15]. To evaluate the resilience of stored credentials, we introduce *synthetic credential attacks*, that use authentication credentials generated with the same distribution of the credentials extracted from manually collected images (see § 5.2.3).

We have captured, collected and generated datasets of 250,332 images, and generated 1 million synthetic credentials (see § 5.5.1). We have used these datasets to generate attack datasets containing more than 3.5 billion (3,567,458,830) authentication instances (see § 5.5.2). We have released the code and data on ai.lock's github [CaS17a].

ai.lock uses an imaging sensor to reliably extract authentication credentials similar to biometrics. Despite lacking the regularities of biometric image features, we show that ai.lock consistently extracts features across authentication attempts from general user captured images, to reconstruct credentials that can match and exceed the security of biometrics (EER = 0.71%). The estimated entropy [Sha01] of ai.lock on 2 billion image pairs is 18.02 bits, comparing favorably with state-of-the-art biometric solutions (see Table 5.1).

## 5.2 Model and Applications

### 5.2.1 System Model

We consider a user that has a camera equipped device, e.g., smartphone or tablet, a resource constrained device such as a smart watch/glasses, or a complex cyber-physical system such as a car. The user needs to authenticate to the device or an application back-end, or authenticate through the device to a remote service. For this, we assume that the user can select and easily access a physical object or scene.

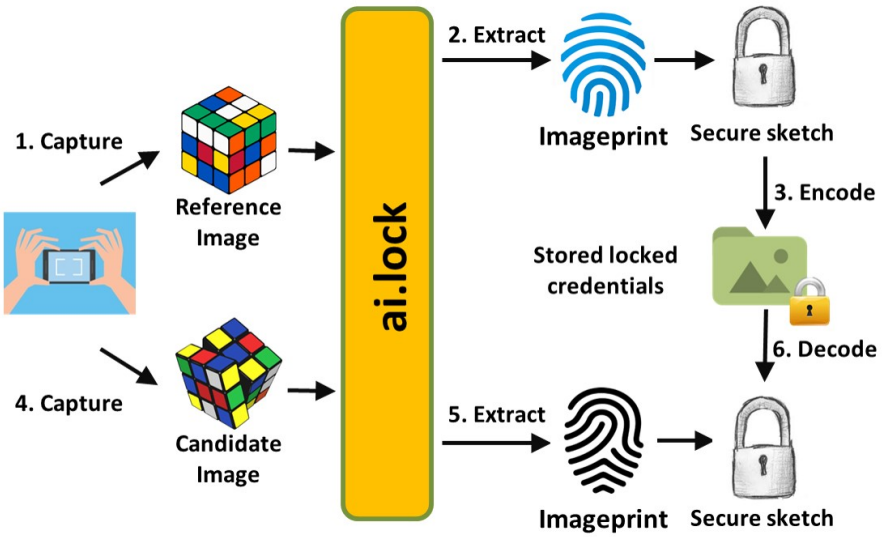



Figure 5.1: ai.lock model and scenario. The user captures the image of an object or scene with the device camera. ai.lock converts the image to a binary *imageprint*, and uses it as a biometric, in conjunction with a secure sketch solution, to securely store authentication information on the device or on a remote server. The user can authenticate only if she is able to capture another image of the same object or scene.

To set her password, the user captures the image of an object/scene with the device camera, see Figure 5.1 for an illustration. ai.lock extracts a set of features from the user's captured *reference* image, then stores this information (imageprint) securely either on the device or on a remote server. We note that, in the former case, the device can associate the reference image with the user's authentication credentials (e.g. OAuth [DH12]) for multiple remote services. To authenticate, the user needs to capture another image. The user is able to authenticate only if the *candidate* image is of the same object or scene as the reference image. Similar to e.g., text passwords, the user can choose to reuse objects across multiple services, or use a unique object per service. Using a unique object per service will affect memorability. However, due to the image superiority effect [NRW76], objects may be easier to remember than text passwords.

### 5.2.2 Application

In the following, we describe a few applications of ai.lock system.

**Alternative to biometric authentication**. Instead of authenticating with her sensitive and non-replaceable biometrics (e.g. face and fingerprint), the user uses a unique nearby scene or object that she carries, e.g., a trinket, Rubik's cube with a unique pattern, printed Random art [PS99], etc. ai.lock moves the source of information from the user to an externality, as it does not require a visual of the user's body, but that of a personal accessory, object, or scene that the user can recreate at authentication time. ai.lock improves on biometrics by freeing users from personal harm, providing plausible deniability, allowing multiple keys, and making revocation and change of secret simple.

**Location based authentication**. The user chooses as password an image of a unique scene at a frequented location (office, home, coffee shop), e.g., section of book shelf, painting, desk clutter. This approach can be generalized to enable location based access control, e.g., to provide restricted access to files and networks in less secure locations.

**Cyber-physical system authentication**. Similar to Pixie, our model supports authentication to cyber-physical systems, including car and door locks, thermostat and alarm systems, where key and PIN entry hardware [Sch17, Sec17] is replaced with a camera. To authenticate, the user needs to present her unique but replaceable authentication object to the camera.

### 5.2.3 Adversary Model

We assume an active adversary who can physically capture or compromise the device that stores the user credentials. Such an adversary can not only access the stored

credentials, but also any keying material stored on the device, then use it to recover encrypted data and to authenticate through the proper channels. However, we assume that the adversary does not have control over the authentication device while the user authenticates (e.g., by installing malware). We also assume an adversary with *incomplete surveillance* [FA12], i.e., who can physically observe the victim during authentication but cannot capture the details of the secret object.

Furthermore, we assume that the adversary has "blackbox access" to the authentication solution, thus can efficiently feed it images of his choice and capture the corresponding imageprint. The adversary can use this output to learn information from the stored credentials. More specifically, we consider the following attacks:

• **Real image attack**. The adversary collects large datasets of images, e.g., manually using a mobile camera, and online. Then, in a brute force approach, he matches each image as an authentication instance against the stored reference credentials until success.

• **Synthetic image attack**. The adversary uses the previously collected images to train a generative model, e.g. [GPAM+14], that captures essential traits of the images, then uses the trained model to generate a large dataset of synthetic images. Finally, the adversary matches each such image against the reference credentials.

• **Synthetic credential attack**. Instead of images, the adversary queries the authentication system with binary imageprints. For this, the adversary extracts the imageprints generated by the authentication solution on real images of his choice. He then generates a large dataset of synthetic credentials that follow the same distribution as the extracted credentials. Finally, he matches each synthetic credential exhaustively against the reference credentials.

• **Object/scene guessing attack**. While we do not consider shoulder surfing attacks which also apply to face based authentication [XPFM16, KFB08b], we as-

sume an adversary that is able to guess the victim's secret object/scene type. The adversary then collects a dataset of images containing the same object or scene type, then uses them to brute force ai.lock (see § 5.6.3).

Finally, we assume the use of standard secure communication channels for the remote authentication scenario where the user credentials are stored on a server.

## 5.3 Problem Definition

Let $\mathbb{I}$ denote the space of images that can be captured by a user with a camera. Let $sim : \mathbb{I} \times \mathbb{I} \rightarrow \{0,1\}$ be a function that returns true when its input images have been taken with the same camera and are of the same object or scene, and false otherwise.

Informally, the *image based authentication* problem seeks to identify a *store* function $S : \mathbb{I} \rightarrow \{0,1\}^k$, and an *authentication* function $Auth : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}$ (for a parameter $k$) that satisfy the following properties. First, it is hard for any adversary with access to only $S(R)$, for a reference image $R \in \mathbb{I}$, to learn information about $R$. That is, $S$ imposes a small entropy reduction on its input image. Second, for any candidate string $C \in \{0,1\}^*$, $Auth(S(R), C) = 1$ only if $C \in \mathbb{I}$ and $sim(R, C) = 1$. Thus, a candidate input to the $Auth$ function succeeds only if it is a camera captured image of the same object or scene as the reference image.

We observe that the *secure sketch* of [DRS04] solves this problem for biometrics: given a biometric input, the secure sketch outputs a value that reveals little about the input, but allows its reconstruction from another biometric input that is "similar". Therefore, the image based authentication problem can be reduced to the problem of transforming camera captured images of arbitrary objects and scenes into biometric-like structures.

| Symbol | Description |
|---|---|
| $\lambda$ | Length of the imageprint for a single image segment |
| $\tau$ | Error tolerance threshold |
| $c$ | Correctable number of bits |
| $s$ | Number of image segments in multi segment schema |
| $t$ | Segment-based secret sharing threshold |

Table 5.2: ai.lock notations.

Hence, we introduce the LSH-related notion of *locality sensitive image mapping* functions. Specifically, let $d : \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \mathbb{R}$ be a distance function (e.g., Hamming), where $\lambda$ is a system parameter. Then, for a given $\delta \in [0,1]$, a $\delta$-Locality Sensitive Image Mapping (LSIM) function $h$ satisfies the following properties:

**Definition 5.3.1** *$h : \mathbb{I} \rightarrow \{0,1\}^\lambda$ is a $\delta$-LSIM function if there exist probabilities $P_1$ and $P_2$, $P_1 > P_2$, s.t.:*

1. *For any two images $I_1, I_2 \in \mathbb{I}$, if $sim(I_1, I_2) = true$, then $\frac{d(h(I_1),h(I_2))}{\lambda} < \delta$ with probability $P_1$.*
2. *For any two images $I_1, I_2 \in \mathbb{I}$, if $sim(I_1, I_2) = false$, then $\frac{d(h(I_1),h(I_2))}{\lambda} > \delta$ with probability $P_2$.*

## 5.4 The ai.lock Solution

We introduce ai.lock, the first locality sensitive image mapping function, and a practical image based authentication system. In the following, we describe the basic solution, then introduce two performance enhancing extensions.

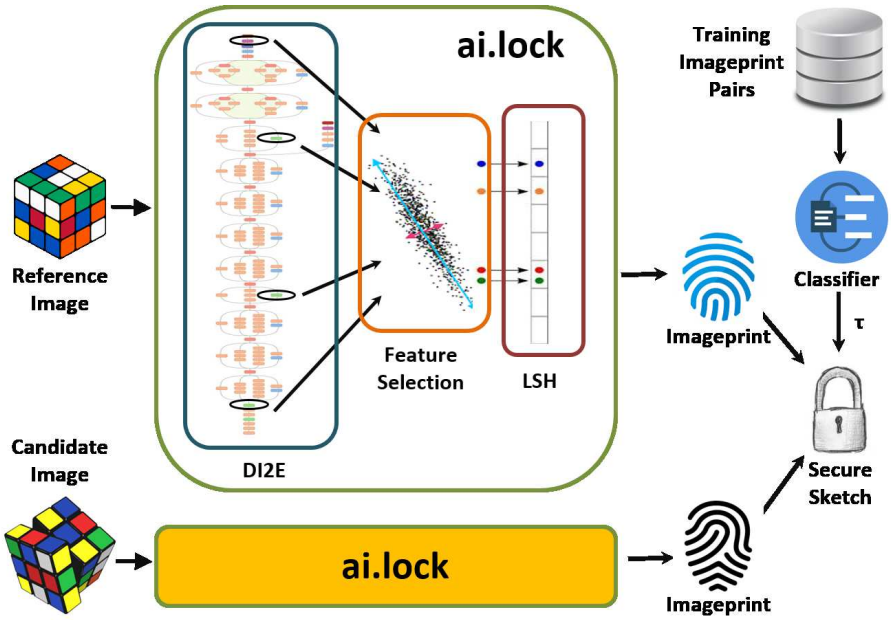



Figure 5.2: ai.lock architecture. ai.lock processes the input image through a deep neural network (i.e., Inception.v3), selects relevant features, then uses locality sensitive hashing to map them to a binary imageprint. ai.lock uses a classifier to identify the ideal *error tolerance threshold* ($\tau$), used by the secure sketch block to lock and match imageprints.

### 5.4.1 ai.lock: The Basic (SLSS) Solution

ai.lock consists of 3 main modules (see Figure 5.2): (1) Deep Image-to-Embedding (DI2E) conversion module (2) feature selection module, and (3) LSH based binary mapping module. We now describe each module and its interface with the secure sketch module. Table 5.2 summarizes the important ai.lock parameter notations.

**Deep image to embedding (DI2E) module**. Let $I$ be the fixed size input image. Let $Emb : \mathcal{I} \rightarrow \mathcal{R}^e$ be a function that converts images into feature vectors of size $e$. We call $Emb(I)$ the *embedding vector*, an abstract representation of $I$. To extract $Emb(I)$, ai.lock uses the activations of a certain layer of Inception.v3 DNN [SVI+16a] when $I$ is the input to the network. Let $e$ denote the size of the output of the layer of the DNN used by ai.lock. Thus, $Emb(I) \in \mathcal{R}^e$.

**Feature selection module**. We have observed that not all the components in the embedding feature vectors are relevant to our task (see § 5.6.1). Therefore, we reduce the dimensionality of the feature vectors to improve the performance and decrease the processing burden of ai.lock. Let $P : \mathcal{R}^e \rightarrow \mathcal{R}^p$, where $p < e$ be a function that reduces the features of an embedding to the ones that are most important. ai.lock uses PCA with component range selection as the $P$ function, and applies it to $Emb(I)$ to find a set of components that can reflect the distinguishing features of images. Thus, the vector produced by feature selection module is $P(Emb(I)) \in \mathcal{R}^p$.

**LSH based binary mapping module**. In a third step, ai.lock seeks to map $P(Emb(I))$ to a binary space of size $\lambda$ that preserves the similarity properties of the input space. A straightforward transformation of the floating point feature values of the $P(Emb(I))$ vectors to their binary representation does not satisfy the second property of the LSIM definition: significantly different feature values that differ only in the most representative bits will have a small Hamming distance.

To address this problem, we use the LSH scheme proposed by Charikar [Cha02]. Let $L : \mathcal{R}^p \rightarrow \{0,1\}^\lambda$ be such a mapping function. ai.lock uses as $L$, a random binary projection LSH as follows. Let $M$ be a matrix of size $p \times \lambda$, i.e. $\lambda$ randomly chosen $p$-dimensional Gaussian vectors with independent components. Calculate $b$ as the dot product of $P(Emb(I))$ and $M$: the projection of the feature vectors $P(Emb(I))$ on $\lambda$ randomly generated vectors.

For each coordinate of $b$, output either 0 or 1, based on the sign of the value of the coordinate. We call this binary representation of the input image $I$, i.e. $\pi(I) = L(P(Emb(I)))$, its *imageprint*. We denote the length of a single imageprint by $\lambda$. Note that, the hash value for the Charikar's method is a single bit ($\lambda = 1$). Therefore, $L$ can be viewed as a function that returns a concatenation of $\lambda$ such random projection bits.

In § 5.6.5, we provide empirical evidence that the function $h = L \circ P \circ Emb$ is a ($\tau$)-LSIM transform (see § 5.6.1), for specific $\tau$ values.

**Secure sketch**. ai.lock extends the secure sketch under the Hamming distance solution from [DRS04]: reconstruct the biometric credential, then compare its hash against a stored value. We briefly describe here the password set and authentication procedures that we use based on ai.lock generated imageprints. Let $ECC$ be a binary error correcting code, with the corresponding decoding function $D$, and let $H$ be a cryptographic hash function.

• **Image password set**. Let $R$ be the reference image captured by the user and let $\pi_R = \pi(R)$ be its ai.lock computed imageprint. Generate a random vector $x$, then compute and store the authentication credentials, $SS(R, x) = \langle SS_1, SS_2 \rangle$, where $SS_1 = \pi_R \oplus ECC(x)$ and $SS_2 = H(x)$.

• **Images based authentication**. Let $C$ be the user captured candidate image, and let $\pi_C = \pi(C)$ be its ai.lock computed imageprint (§ 5.4). Retrieve the stored $SS$ value and compute $x' = D(\pi_C \oplus SS_1)$. The authentication succeeds if $H(x') = SS_2$.

### 5.4.2 ai.lock Variants

In the following, we introduce two ai.lock extensions, intended to increase the entropy provided by ai.lock's imageprints. First, we modify ai.lock to use the embedding vectors obtained from multiple layers of Inception.v3 network. Second, we extend ai.lock to split the input image into multiple overlapping segments and concatenate their resulting binary representations.

By combining the concept of these approaches we can have 4 variants of ai.lock as follows: Single Layer Single Segment (SLSS) image, Multi Layer Single Segment (MLSS) image, Single Layer Multi Segment (SLMS) image and Multi Layer Multi

Segment (MLMS) image. In the following, we describe necessary modifications to the basic ai.lock modules to work with these variants.

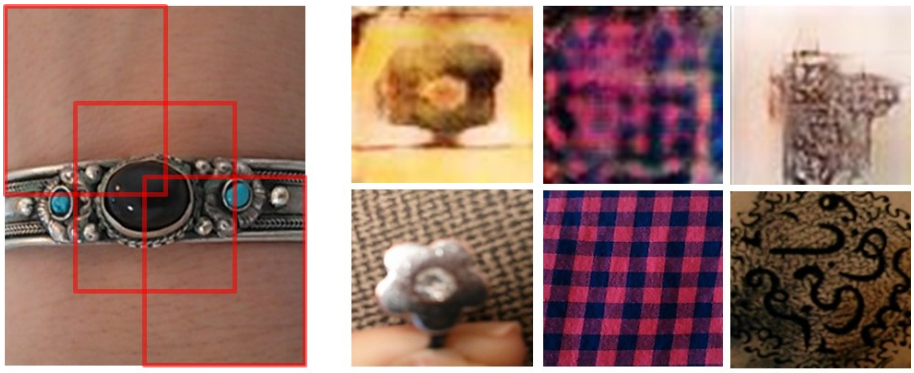
### ai.lock with Multiple DNN Layers

Representations learned by a DNN are distributed in different layers of these networks. The lower (initial) layers of convolutional neural networks learn low level filters (e.g. lines, edges, colors), while deeper layers learn more abstract representations (e.g. shapes) [ZF14]. The use of a single DNN layer prevents the basic ai.lock solution from taking advantage of both filters.

To address this issue, we propose an ai.lock extension that collects the embedding vectors from multiple ($l$) layers of Inception.v3 network. In addition, we modify the basic ai.lock feature extractor module as follows. The Principal Components (PCs) of activations for each layer are computed separately and are mapped to a separate binary string of length $\lambda$. Then, the binary strings constructed from different layers are concatenated to create a single imageprint for the input image. Thus, the length of the imageprint increases linearly with the number of layers used in this schema.

### ai.lock with Multiple Image Segments

We divide the original image into $s$ overlapping segments (see Figure 5.3(a)). We then run the basic ai.lock over each segment separately to produce $s$ different imageprints of length $\lambda$. However, we identify the PCs for the embedding vectors of each segment based on the whole size images. The intuition for this choice is that random image segments are not good samples of real objects and may confuse the PCA. We then generate the imageprint of the original, whole size image, as the concatenation of the imageprints of its segments.



(a) (b)

Figure 5.3: (a) 3 overlapping segments of an image. (b) Top: sample images generated by DCGAN, Bottom: visually similar images in Nexus Dataset to images generated by DCGAN.

**Secure sketch sharing**. We extend the secure sketch solution with a $(t, s)$-secret sharing scheme. Specifically, let $x_1, .., x_s$ be $(t, s)$-shares of the random $x$, i.e., given any $t$ shares, one can reconstruct $x$. Given a reference image $R$, let $R^{(1)}, ..R^{(s)}$ be its segments, and let $\pi_R^{(i)} = \pi(R^{(i)})$, $i \in 1, 2, 3, ..., s$ be their imageprints. Then, we store $SS(R, x) = \langle SS_1^{(1)}, .., SS_1^{(s)}, SS_2 \rangle$, where $SS_1^{(i)} = \pi_R^{(i)} \oplus ECC(x_i)$ and $SS_2 = H(x)$. To authenticate, the user needs to provide a candidate image $C$, whose segments $C^{(i)}, i = 1..s$ produce imageprints $\pi_C^{(i)} = \pi(C^{(i)})$ that are able to recover at least $t$ of $x$'s shares $x_i$.

## 5.5 Implementation and Data

We build ai.lock on top of the Tensorflow implementation for Inception.v3 network [ten17]. For the error correcting code of secure sketches, we use a BCH [Hoc59a, BRC60] open source library [Jef17], for syndrome computation and syndrome decoding with correction capacity of up to $c$ bits. The value for $c$ is calculated empirically using the training dataset (see § 5.6.1).

**Basic (SLSS) ai.lock**. In the basic ai.lock solution, we use the output of the last hidden layer of Inception.v3 network, before the softmax classifier, consisting of $2,048$ float values. Our intuition is that this layer provides a compact representation (set of features) for the input image objects, that is efficiently separable by the softmax classifier.

**Multi layer ai.lock**. For the multi DNN layer ai.lock variants, we have used 2 layers ($l = 2$). The first layer is the "Mixed_8/Pool_0" layer and the second layer is the last hidden layer in Inception.v3. The embedding vector for the "Mixed_8/Pool_0" consists of $49,152$ float values. As described in § 5.4.2, the embedding vectors of each layer are separately processed by the feature selection and LSH modules; the resulting binary strings are concatenated to form the imageprint of size $2\lambda$.

**Multi segment ai.lock**. For the multi segment ai.lock variant, we split the image into multiple segments that we process independently. Particularly, we consider 5 overlapping segments, cropped from the top-left, bottom-left, top-right, bottom-right and the center of the image. We generate segments whose width and height is equal to the width and height of the initial image divided by 2, plus 50 pixels to ensure overlap. The extra 50 pixels are added to the interior sides for the side segments. For the middle segment, 25 pixels are added to each of its sides. Each segment is then independently processed with the basic ai.lock (i.e., last hidden layer of Inception.v3, PCA, and LSH).

**Multi layer multi segment ai.lock**. This is a hybrid of the above variants: split the image into 5 overlapping parts, then process each part through Inception.v3 network, and extract the activation vectors for each of the two layers of Inception.v3 (the last hidden layer and Mixed_8/Pool_0 layer). The output of each layer for each segment is separately processed as in the basic ai.lock. Thus, the resulting imageprint of the image has $10\lambda$ bits.

### 5.5.1 Primary Data Sets

#### Real Images

**Nexus dataset**. We have used a Nexus 4 device to capture 1,400 photos of 350 objects, belonging to 33 object categories. Example of object categories in this dataset includes watches, shoes, jewelry, shirt patterns, and credit cards. We have captured 4 images of each object, that differ in background and lighting conditions. Note, this is the same dataset as we introduced in § 4.4.1.

**ALOI dataset**. We have used the "illumination direction" subset of the Amsterdam Library of Object Images (ALOI) [GBS05b] dataset. This dataset includes 24 different images of 1000 unique objects (24,000 in total) that are taken under different illumination angles.

**Google dataset**. We have used Google's image search to retrieve at least 200 images from each of the 33 object categories of the Nexus image dataset, for a total of 7,853 images. This dataset forms the basis of a "targeted" attack. This is the same dataset as we used to break Pixie (see § 4.5.2).

**YFCC100M toy dataset**. We have extracted a subset of the Yahoo Flickr Creative Commons 100M (YFCC100M) [TSF+16] image dataset (100 million Flickr images). This subset includes 126,600 Flickr images tagged with the "toy" keyword, and not with "human" or "animal" keywords.

#### Synthetic Data

**Synthetic image dataset**. Manually capturing the Nexus dataset was a difficult and time consuming process. In order to efficiently generate a large dataset of simi-

lar images, we have leveraged the ability of generative models to discover an abstract representation that captures the essence of the training samples. Generative models, including Variational AutoEncoders (VAE) [KW13] and Generative Adversarial Networks (GAN) [GPAM$^+$14], are trained to generate samples that are similar to the data they have been trained on (see § 3.1.5 for a review on GANs). Such models have been shown to be suitable for representation learning tasks, e.g., [RMC15].

We have used a DCGAN [RMC15] to generate a large set of synthetic images that are similar to the images in the Nexus dataset. Specifically, we have trained a DCGAN [RMC15] using the images of the Nexus dataset for 100 training epochs. Image augmentation, e.g., rotation, enhancement, and zoom, is performed to artificially increase the number of Nexus image dataset samples to include 20 variants per image. We then used the trained network to generate synthetic images: generate a random vector ($z$) drawn from the uniform distribution, then feed $z$ to DCGAN's generator network to construct an image. We repeated this process to generate 200,000 images, that form our *synthetic image dataset*. Figure 5.3(b) shows sample images generated by this network, alongside similar images from the Nexus dataset.

**Synthetic credential dataset**. We have generated the binary imageprints for the images in Nexus dataset based on the best parameters of ai.lock (see § 5.6.1). For each considered $\lambda$ value, we consider the value at each position of the binary imageprint as an independent Bernoulli random variable. We then calculate the probability of observing a 1 in each position based on the imageprints of the Nexus dataset. We use these probabilities to draw 100,000 random samples (of length $\lambda$) from the corresponding Bernoulli distribution for each position. The resulting random binary imageprints form our *synthetic credential dataset*. We have experimented with 10 values of $\lambda$ ranging from 50 to 500, thus, this dataset contains 1 million synthetic imageprints.

### 5.5.2 Evaluation Datasets

We use the above image datasets to generate *authentication samples* that consist of one candidate image and one reference image.

#### ai.lock attack dataset

We use roughly 85% of the images from the Nexus, ALOI and Google datasets as a *training* set to train and estimate the performance of ai.lock. We use the remaining 15% of images in each dataset (i.e., 220 Nexus, 3,600 ALOI and 1,178 Google images) as a *holdout* set. We use the holdout dataset to assess the generalization error of the final model, and as a real image attack dataset (see § 5.6.3).

We generate the samples in holdout dataset using each subset of Nexus, ALOI and Google separately as follows. Each image of the Nexus holdout dataset is chosen as a reference image once, then coupled once with all the other images in the Nexus, ALOI and Google sets, used as candidate images. Therefore, there are $\frac{220\times219}{2} = 24,090$ combinations of samples for the images in the Nexus set. For each 55 unique objects in this set, there are 6 ($\binom{4}{2}$) possible valid samples that compare one image of this object to another image of the same object. Thus, there are $55 \times 6 = 330$ valid samples in the Nexus set. We then generate $220 \times 3,600 = 792$K and $220 \times 1,178 = 259,160$ invalid samples from comparing Nexus images to images in ALOI and Google sets respectively. Therefore, the ai.lock holdout set contains a total of $1,075,250$ samples.

In addition, the training set is further divided into 5 folds, for cross validation. Each training fold contains 236, 4080 and 1335 images of Nexus, ALOI and Google datasets respectively. Therefore, there are $\frac{236\times235}{2} = 27,730$ samples for the fold's 59 unique Nexus set objects, of which $59 \times 6 = 354$ pairs are valid. Similarly, we generate $236 \times 4,080 = 962,880$ and $236 \times 1,335 = 315,060$ invalid samples, that

consist of Nexus images coupled with ALOI and Google images, respectively. Thus, each training fold has a total of $1,305,670$ samples, of which 354 are valid.

**Synthetic image attack datasets**

We divide the synthetic image dataset of § 5.5.1 into 2 equal sets, each containing 100,000 images. Then, we build two *synthetic image attack datasets* (DS1 and DS2) by repeating the following process for each subset of the synthetic image dataset: combine each Nexus dataset image, used as a reference image, with each image from the subset of the synthetic image dataset, used as a candidate image. Therefore, in total we have 140 million samples in each of DS1 and DS2.

**Synthetic credential attack dataset**

We use the synthetic credential dataset described in § 5.5.1 to build a *synthetic credential attack dataset*: for each value of $\lambda$, combine the imageprint of each Nexus dataset image, used as a reference imageprint, with each imageprint in synthetic credential dataset, used as the candidate imageprint. Hence, we have 140 million authentication samples in this dataset for each value of $\lambda$. We repeat this process for 10 values of $\lambda$, ranging from 50 to 500. Therefore, in total this dataset contains $10 \times 140$ M $= 1.4$ billion samples.

**Illumination robustness evaluation dataset**

To evaluate the performance of ai.lock under illumination changes, we use the ALOI holdout set ($3,600$ images) that includes up to 11 images of each object captured under a different illumination condition. Specifically, we pair each image in the ALOI holdout set (i.e., not used during training) with all the other images in this

| $\lambda$ | **50** | **100** | **150** | **200** | **250** | **300** | **350** | **400** | **450** | **500** |
|---|---|---|---|---|---|---|---|---|---|---|
| $\tau \times 10$ | 7.80 | 7.30 | 7.07 | 6.95 | 6.80 | 6.87 | 6.80 | 6.85 | 6.87 | 6.82 |

Table 5.3: Error tolerance threshold ($\tau$) values for the basic ai.lock obtained through cross validation over the ai.lock dataset, when using PCs with feature ranked 200-400.

set. Therefore, we have a total of $\frac{3600 \times 3599}{2} = 6,478,200$ authentication samples in the illumination robustness evaluation dataset, of which $6,306$ samples are valid.

**Entropy evaluation dataset**

We randomly selected 2 billion unique pairs of images from the YFCC100M toy dataset. In each pair, an image is considered to be the reference, the other is the candidate.

## 5.6 Experimental Evaluation

We evaluate ai.lock and its variants. First, we describe the process we used to identify the best ai.lock parameters. We use these parameters to evaluate the performance of ai.lock under the attack datasets of § 5.5.2. We also show that ai.lock is a $\delta$-LSIM function, empirically estimate its entropy, and measure its speed on a mobile device.

### 5.6.1 ai.lock: Parameter Choice

We identify the best parameters for the ai.lock variants, starting with the SLSS solution, using 5 fold cross validation on the ai.lock training dataset (see § 5.5.2).

### Best error tolerance threshold

In each of the 5 cross validation experiments, we identify the best threshold that separates the binary imageprints of the testing set. Particularly, we normalize the Hamming distance of each pair of imageprints in the test fold by the length of the imageprints. Then, we apply more than 4K different real values, between 0 and 1, as a threshold on the normalized Hamming distances of the authentication pairs to classify them. At the end of the $5^{th}$ cross validation experiment, we select the threshold that has the maximum average performance, in terms of F1 score, as the best separating threshold. We call this the *Error Tolerance Threshold*, which we denote by $\tau$.

Table 5.3 reports the $\tau$ values for basic ai.lock and MLSS ai.lock variant with different values of $\lambda$, when using PCs with feature rank 200-400. We observe that as $\lambda$ increases, the value for $\tau$ decreases: we posit that larger $\lambda$ values preserve more information about the input vectors (PCs of the embedding vectors) in the LSH output.

We translate $\tau$ to the error correcting capacity required for ECC. Specifically, for an imageprint of length $\lambda$, we choose an ECC that is able to correct up to $c = \lfloor \lambda \times (1 - \tau) \rfloor$ bits.

### Best principal component range

To identify the best PC range, we use 5 fold cross validation as follows. First, we retrieve the embedding vector (output of the last hidden layer of Inception.v3) for each image in the ai.lock training dataset. Then, for each cross validation experiments, we use 4 training folds to find the principal components of the embedding vectors. Then, we transform the embedding vectors of the test fold into the newly

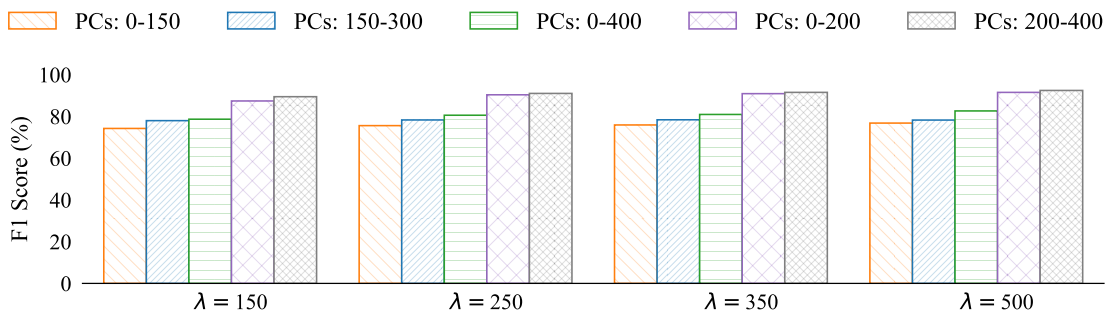



Figure 5.4: Comparison of ai.lock performance (F1 score) when using different subset of principal component feature ranks for different imageprint length ($\lambda$) values. PCs ranked 200-400 constantly outperform other tested subsets.

identified feature space. Finally, we project them into several randomly generated vectors (LSH) to construct the binary imageprint of the images.

To choose the best PCs, we have experimented with different subsets of the transformed feature space of various size including the first and second consecutive principal component sets of size 50, 100, 150, and 200, as well as, the first 400 PCs.

Figure 5.4 shows the cross validation performance achieved by ai.lock when using different subsets of PC features for different $\lambda$ values. We observe that the PCs ranked 200-400 perform consistently the best. This might seem surprising, as higher ranked PCs have higher variability and thus we expected that they would have more impact in differentiating between valid and invalid samples. We conjecture that some of the lower rank coordinates of these transformed vectors are more efficient in capturing the lower level details of the input object images that differentiate them from other object images.

**Motivation for feature selection using PCA.** We now justify the need for the PCA step of ai.lock. For this, we compare the best version of ai.lock running PCA (i.e., features ranked 200-400), with two other versions. First, we consider a baseline version (which we call "Raw"), that uses no feature selection component. Specifically, Raw applies LSH to the raw embedding vectors, then, identifies the

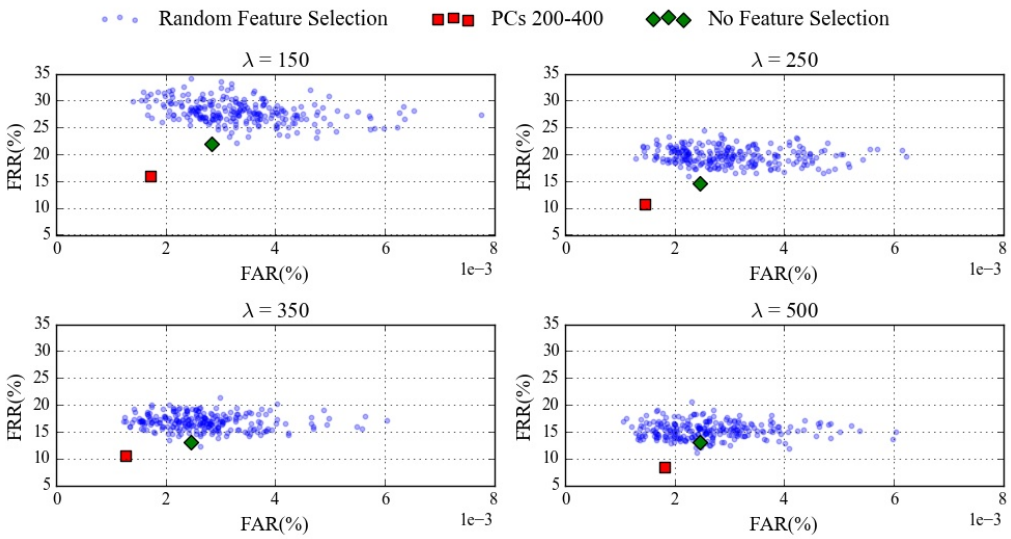



Figure 5.5: PCA motivation: FRR vs. FAR of (i) ai.lock when using PCA (with features ranked 200-400), (ii) ai.lock with no feature selection ("Raw"), and (iii) 250 independent instances of ai.lock when using a feature selection approach that randomly selects 200 features. ai.lock with PCA consistently achieves the lowest FRR and often the lowest FAR.

best threshold $\tau$ using the 5-fold cross validation experiment described previously for ai.lock. Second, we compare against an ai.lock variant where we replace the PCA component with a random choice of 200 features (of the embedding vectors) produced by the last hidden layer of Inception.v3. Figure 5.5 shows the results of this comparison for $\lambda$ values of 150, 250, 350 and 500, and 250 different instances of ai.lock with random feature selection. We observe that ai.lock with PCA (PCs of rank 200-400) consistently achieves the significantly lower FRR, and often the lowest FAR. In addition, we observe that randomly choosing the features is not ideal, as it often performs worse than when no feature selection is used at all.

**ai.lock MLSS variant**. Similar to the basic ai.lock, we have experimented with multiple ranges of PCs and $\lambda$ values to identify the $\tau$ values for MLSS ai.lock, using the 5 fold cross validation experiment on the ai.lock training dataset. Table 5.4 reports the $\tau$ values for MLSS ai.lock variant with different values of $\lambda$. As

| $\lambda$ | **200** | **250** | **300** | **350** | **400** | **450** | **500** |
|---|---|---|---|---|---|---|---|
| $\tau \times 10$ | 6.75 | 6.74 | 6.61 | 6.58 | 6.55 | 6.57 | 6.56 |

Table 5.4: Error tolerance threshold ($\tau$) values for MLSS ai.lock obtained through cross validation over the ai.lock dataset, when using PCs with feature ranked 200-400.

| $t$ (matching segment counts out of 5) | **3** | 4 | 5 |
|---|---|---|---|
| F1 score (%) for SLMS | **93.13** | 90.95 | 85.84 |
| F1 score (%) for MLMS | **95.53** | 94.64 | 92.42 |

Table 5.5: Cross validation performance (F1 score) for different values of $t$ (number of segments that need to match out of 5) when using PCs with feature rank 200-400 and $\lambda = 500$ for SLMS and MLMS variants of ai.lock. $t = 3$ consistently achieves the best performance.

mentioned before, the value of $\tau$ decreases as we increase $\lambda$.

**ai.lock Multi segment variants**. For this ai.lock variant, we identify the $\tau$ values separately for each image segment, using the 5-fold cross validation experiment explained previously. Therefore, we end up having 5 different $\tau$ values corresponding to each image segment. The $\tau$ corresponding to each segment can be used to identify if there is a match between the piece of the candidate image to the corresponding piece in the reference image. We say that the whole candidate and reference images match, when $t$ of their segments match. We have tested with $t$ ranging from 3 to 5 and observed that $t$=3 achieved the best F1 score (see Table 5.5).

### 5.6.2 Cross validation performance for trained ai.lock model

We now report the cross validation performance of ai.lock with the parameters identified above, for $\lambda$ ranging from 50 to 500. Figures 5.6(a)-(c) compare the F1 score, FAR and FRR values of the best version of the ai.lock variants (basic SLSS,

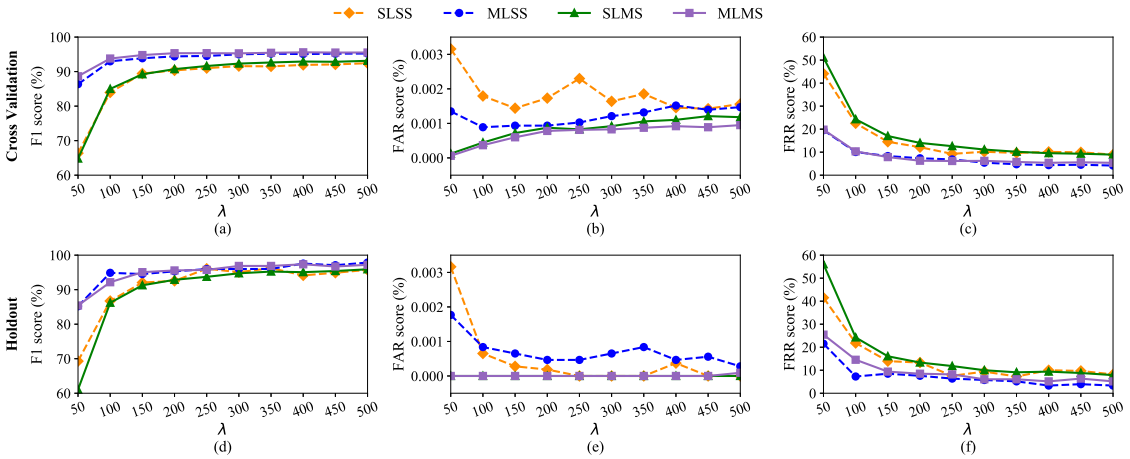



Figure 5.6: (a-c) ai.lock cross validation performance, and (d-f) ai.lock holdout performance using different ai.lock variants: Single Layer Single Segment (SLSS), Multi Layer Single Segment (MLSS), Single Layer Multi Segment (SLMS), Multi Layer Multi Segment (MLMS). Exploiting information from multiple Inception.v3 DNN layers (multi layer variants) lowers the FRR, while splitting images into smaller segments (multi segment variants) lowers the FAR. The MLMS variant of ai.lock consistently achieves the lowest FAR, that can be as low as 0% for the holdout dataset.

SLMS, MLSS, and MLMS) over the 5-fold cross validation experiments, using ai.lock training dataset. The performance of all ai.lock variants improves with increasing the value of $\lambda$. The MLMS ai.lock achieves the best performance, with an F1 score of 95.52% and FAR of 0.0009% when $\lambda = 500$. The MLSS ai.lock also consistently improves over the basic ai.lock, with a smaller FRR and a smaller or at most equal FAR. Its FRR (4.18% for $\lambda = 500$) is slightly smaller than that of MLMS variants (5.36%), but it exhibits a slight increase in FAR. For large values of $\lambda$, the FRR of SLMS and SLSS are almost equivalent.

The average cross validation Equal Error Rate (EER, the rate at which the FAR = FRR) of ai.lock for the SLSS and MLSS variants is less than 0.67% and 0.17% respectively when using PCs with feature rank $200 - 400$ and $\lambda = 500$.

| $\lambda$ | **50** | **150** | **250** | **350** | **500** |
|---|---|---|---|---|---|
| FAR$\times 10^{+6}$ | 33.87 | 4.34 | 3.29 | 0.69 | 0.20 |

Table 5.6: SLSS ai.lock performance on synthetic attack DS1. The FAR decreases significantly as $\lambda$ grows from 50 to 500. The FAR when $\lambda = 500$ is only $0.2 \times 10^{-6}$.

The purpose of the LSH-based transformation is to encode the feature vector of an image extracted by a DNN into a binary string. Our conjecture is that larger lambda values extract more high quality information about the feature vectors, which in turn leads to lower FAR and FRR. This is partly due to the random nature of the LSH we used (see Figure 5.10), where roughly half of the bits among different images are different, and images of the same object have a smaller distance overall. Using more LSH bits reduces the variance of the distance that was due to perturbations from using a random projection, hence provides a better separation between TP and FP image comparisons.

### 5.6.3 ai.lock Under Attack

#### Holdout dataset, real image attack

The performance over the ai.lock holdout set is reported in Figure 5.6(d)-(f). As before, the performance of all the ai.lock variants improves with the increase in $\lambda$. In agreement with the results of the cross validation experiments, we conclude that exploiting information from multiple Inception.v3 layers decreases the FRR, while using information from multiple image segments decreases the FAR. In addition, the MLMS ai.lock variant achieves the highest F1 score (97.21% for $\lambda = 500$). The SLMS and MLMS schema consistently achieve the lowest FAR, which is as low as 0% on the holdout dataset.

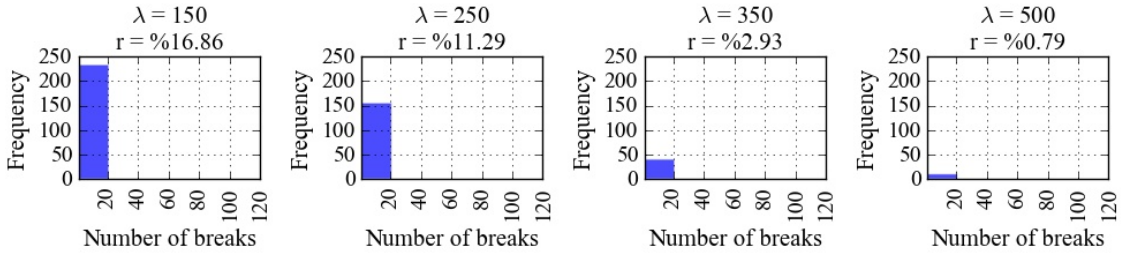



Figure 5.7: Histogram of the number of broken reference images, using the synthetic attack dataset DS1. The $x$ axis shows the range for the number of times a breakable Nexus reference image is defeated by the attack images and the $y$ axis shows the number of such breakable images. A majority of the "broken" reference images are defeated only by a small number of candidate images. The ratio of broken references ($r$) decreases significantly when $\lambda$ increases.

### Synthetic image attack

We use the synthetic attack dataset DS1 of § 5.5.2 to evaluate the performance of SLSS ai.lock, using the trained parameters of § 5.6.1. We emphasize the importance of achieving a low FAR in this experiment: this powerful adversary can generate and try many synthetic images.

Table 5.6 shows the performance of ai.lock in classifying these attack samples. The FAR decreases significantly with $\lambda$, and is as low as 0.00002% when $\lambda = 500$.

Figure 5.7 shows the histogram of the number of times when the Nexus image references are broken using the synthetic image dataset DS1, for different $\lambda$ values. The value $r$ indicates the percentage of the reference images that have been broken at least once. The proportion of the reference images that have been broken at least once decreases significantly by increasing $\lambda$: from 16.86% to 0.79% (11 Nexus images) when $\lambda$ is 150 and 500 respectively. A majority of the broken references are broken only by a small number of candidate images: when $\lambda = 500$, only 2 of the 11 broken images have been broken 5 times by the synthetic images in DS1. The empirical average number of trials until finding the first matching synthetic image, over the 11 broken reference images, is 31,800.

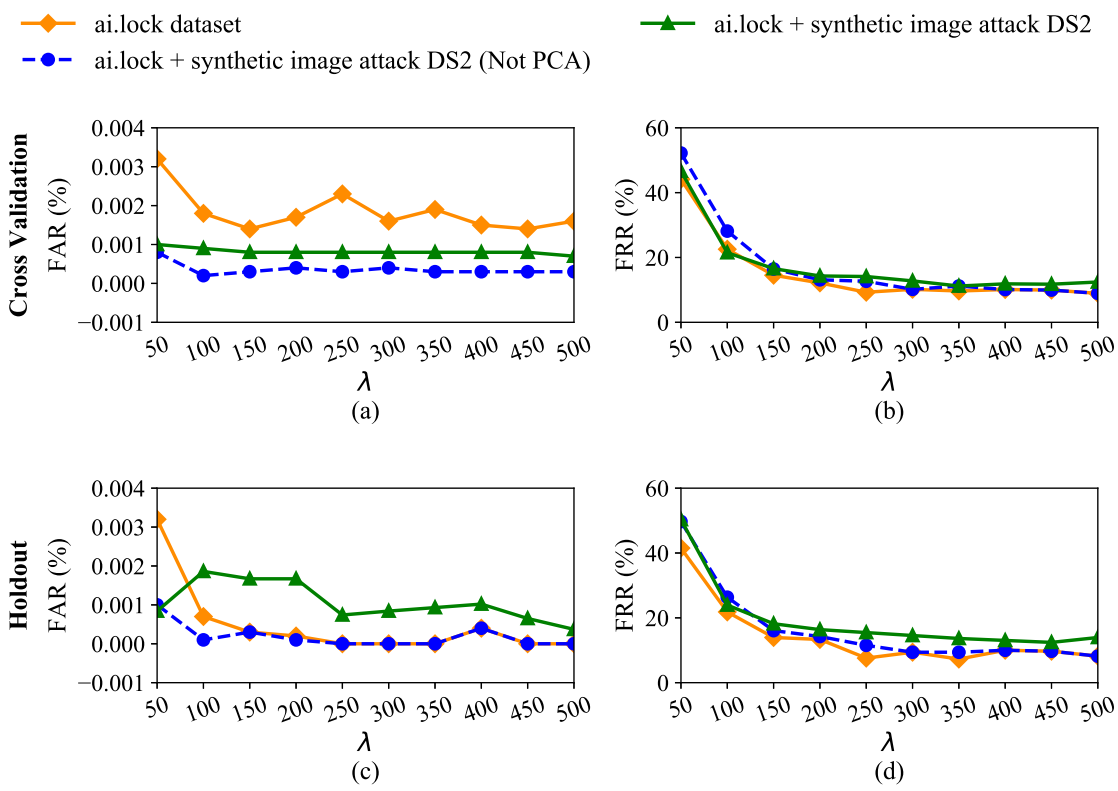



Figure 5.8: (a) Cross validation FAR, (b) Cross validation FRR , (c) Holdout FAR, and (d) Holdout FRR of SLSS ai.lock when trained over the ai.lock and synthetic image attacks of DS2.

**Vaccinated ai.lock**. To further improve the ai.lock resistance to synthetic image attacks, we use the synthetic image attack dataset DS2 (see § 5.5.2) along with the ai.lock training dataset, to train ai.lock. Specifically, we divide the synthetic image attack dataset DS2 into 5 folds and distribute them into the 5 training folds of the ai.lock dataset. In other words, we train ai.lock on an additional $236 \times 20{,}000 = 4{,}720{,}000$ invalid authentication samples. The holdout set remains untouched and is used to evaluate the effectiveness of this approach. Then, we train ai.lock with SLSS as before using the cross validation experiment (see § 5.6.1).

We experimented with two cases. First, the invalid synthetic image attack samples in DS2 contribute to both PCA-based feature selection and the error tolerant

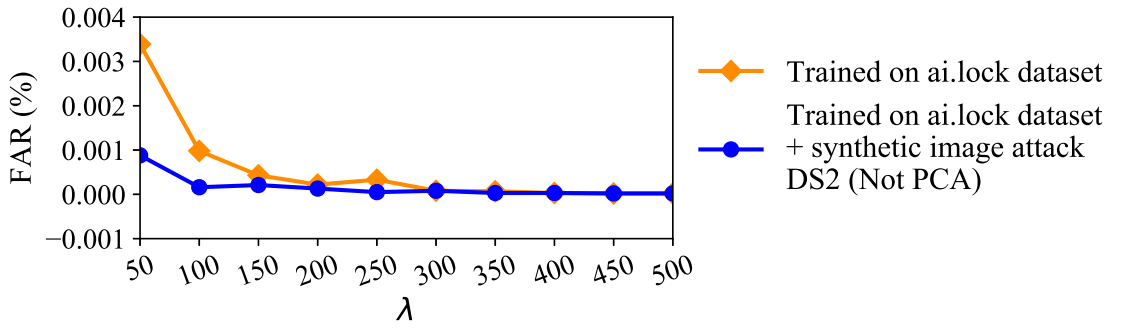



Figure 5.9: FAR of ai.lock on synthetic image attack, when trained on the ai.lock dataset vs. when trained also on DS2. The "vaccinated" ai.lock improves its resistance to the synthetic image attack: the FAR drops by more than 74%, 51% and 59% when $\lambda$ is 50, 150 and 350 respectively.

threshold ($\tau$) discovery processes. Second, those samples are only used in the process of discovering $\tau$. Figure 5.8 shows the cross validation FAR and FRR (a, b) as well as the performance over the holdout set (c, d). In both experiments, we observed a drop in the FAR of ai.lock, however, the FRR increases. The FAR improvement is higher for the second case. We conjecture that the inclusion of synthetic, not camera captured images, is misleading the PCA based feature selection module into capturing irrelevant information.

We used the ai.lock trained on the synthetic image attack dataset DS2 to evaluate its performance over the synthetic image attack DS1. Figure 5.9 compares the performance of ai.lock when trained on the ai.lock dataset and when trained on the ai.lock and the synthetic dataset DS2. Training also over synthetic image attack samples helps ai.lock to be more resilient to synthetic image attack, especially for small values of $\lambda$.

**Synthetic credential attack**. Table 5.7 shows the FAR values for ai.lock under the synthetic credential attack dataset described in § 5.5.2. For all values of $\lambda$ greater than 300, the FAR of ai.lock is equal to 0. Even for a $\lambda$ of 50, the FAR is $11.89 \times 10^{-4}\%$. This is an important result: even a powerful adversary who can

| $\lambda$ | 50 | 150 | 250 | 350 | 500 |
|---|---|---|---|---|---|
| FAR$\times 10^{+6}$ | 11.89 | 0.09 | 0.03 | 0.000 | 0.000 |

Table 5.7: SLSS ai.lock performance on the synthetic credential attack. ai.lock is unbreakable under 1.4 billion samples of the synthetic credential attack: its FAR is 0 when $\lambda \geq 300$.

| # of words in image search query | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Dataset size ($d_{size}$) | 12,413 | 24,882 | 26,418 | 26,766 |
| Avg # of trials before FA (random order) | 12,078 | 23,205 | 24,641 | 25,028 |
| Avg # of trials before FA (guessing attack) | 12,034 | 22,755 | 23,921 | 24,488 |
| Portion of broken references (%) | 5.0 | 9.0 | 10.9 | 9.0 |

Table 5.8: ai.lock under the object guessing attack. The average number of trials before the first false accept (FA) drops only slightly in the object guessing attack scenario when compared to a random ordering of attack images. Thus, knowledge of the authentication object type provides the adversary only nominal guessing advantage.

create and test synthetic credentials on a large scale, is unable to break the ai.lock authentication.

### Object/Scene Guessing Attack

**Data**. We have asked a graduate student to tag each of the 55 unique object images in the Nexus holdout set with 1 to 4 words. For each value of the number of tags per image (i.e., 1 to 4), and each object image, we collected 300-500 images provided by Google's image search engine. Thus, we generated 4 Google image datasets, one for images found when searching with 1 tag, another when searching with 2 tags, etc. In total, we have collected 90,479 images.

**ai.lock performance under object guessing attack**. We use the 4 collected image datasets from Google to generate a total of 19,905,380 "guessing attack" authentication samples, and use them to evaluate the *guessing entropy* [DMR04] of ai.lock under an object/scene guessing attack (see § 6.2).

Specifically, using each of the 4 Google image datasets we perform the following two brute force attacks. The first attack emulates an object guessing attack: re-order the images in the Google dataset to start the brute force attack with the images of the same object type, then continue with images of other object categories in a random order. Finally, count the number of trials before the first match (false accept) occurs. The second attack is a standard brute force attack: randomly shuffle the images in the Google image dataset and use them to brute force each image in the Nexus holdout set. We use the second attack as a baseline, to determine if knowledge of the object type impacts the trial count to success. In both attacks, we count each of the unbreakable reference images as "success" at $d_{size}$ trials, where $d_{size}$ is the number of images in the corresponding Google image dataset.

Table 5.8 summarizes the ai.lock performance under the object/scene guessing attack scenario. We observe an increase in the portion of the Nexus images that are broken when the simulated adversary uses more words to describe the authentication objects for collecting the attack image dataset. However, for all experiments, the average number of trials before success drops only slightly in the object guessing attack scenario compared to the baseline. This is due to the fact that the reference images were mostly broken with images of different object categories. We conclude that knowledge of the secret object type does not provide the adversary with a significant guessing advantage.

### 5.6.4 Resilience to Illumination Changes

We evaluate the resilience of ai.lock to illumination changes using the 6,478,200 authentication samples of the illumination robustness evaluation dataset (§ 5.5.2). While the FAR of the MLMS variant of ai.lock (for $\lambda = 500$ and $t = 3$) remains

very small (0.006%), its FRR increases to 16.9%. Decreasing the required matching segments count ($t$) to 2, reduces the FRR to 11.43%, which results in a slightly higher FAR of 0.010%.

### 5.6.5 Is ai.lock $\delta$-LSIM?

We now evaluate if the basic ai.lock (SLSS) variant, with the parameters identified in § 5.6.1 preserves the similarity of the input space, i.e., if it satisfies the LSIM properties (see Definition 5.3.1). We use the ai.lock holdout set to evaluate the probability of obtaining the same hash value for valid and invalid samples.

Let $\pi_i$ and $\pi_j$ be the imageprints corresponding to two images in the ai.lock holdout set. Let $d_H(\pi_i, \pi_j)$ denote the Hamming distance and $S_H(\pi_i, \pi_j)$ denote the normalized *Hamming similarity* of these imageprints, i.e., $S_H(\pi_i, \pi_j) = 1 - \frac{d_H(\pi_i, \pi_j)}{\lambda}$.

The output of ai.lock can be considered either as a single bit or a string of bits. In the former case, the imageprints consist of the concatenation of the output of multiple hash functions, while in the later case, the entire imageprint is assumed to be the ai.lock hash value. In the following, we empirically evaluate the $P_1$ and $P_2$ values (see Definition §5.3.1), for the case where the ai.lock individual imageprint bits are considered as the hash value. We further show that ai.lock is also a $\delta$-LSIM function when the entire ai.lock imageprint is considered as the hash value (multi-bit).

#### ai.lock with single bit hash value is a $\delta$-LSIM

We show that ai.lock with a single bit hash value is a $\delta$-LSIM (see Definition 5.3.1).

ai.lock uses Charikar's random projection LSH [Cha02]. Therefore, for any embedding vector (the input to LSH function) $u$ and $v$, $Pr[1\ bit\ collision] = 1 -$

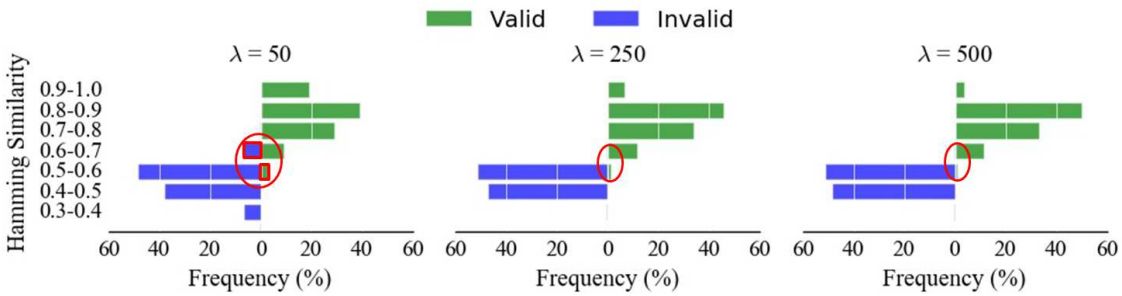



Figure 5.10: Histograms of normalized Hamming similarity between imageprints of valid and invalid authentication samples in the ai.lock holdout set. The red rectangles pinpoint the focus areas: valid samples with Hamming similarity below 0.6 and invalid samples with Hamming similarity above 0.6. Higher values of $\lambda$ provide more effective separation between valid and invalid samples: when $\lambda = 500$, no invalid samples have similarity above 0.6.

| $\lambda$ | **150** | **350** | **500** |
|---|---|---|---|
| $P_1$ | 0.799 | 0.797 | 0.796 |
| $P_2$ | 0.500 | 0.500 | 0.500 |

Table 5.9: Average probability of collision, for valid ($P_1$) and invalid ($P_2$) samples in the ai.lock holdout set per imageprint bit basis. In all cases, $P_1 > P_2$, thus conclude that ai.lock with single bit hash value is an LSIM function.

$\frac{\theta(u,v)}{\pi}$, where $\theta(u, v)$ denotes the angle between $u$ and $v$. We use the angle between the feature vectors of images in the ai.lock holdout set to compute the average probability of collision: 0.79 for valid and 0.50 for invalid authentication samples.

Figure 5.10 shows the histogram of normalized Hamming similarity between imageprints in the valid and invalid samples of the ai.lock holdout set. Unsurprisingly, most invalid samples have a Hamming similarity between 0.4 and 0.6: different images have imageprints that are similar in around half of their bits (see also Table 5.9). We observe that the overlap between the Hamming similarities of valid and invalid samples significantly reduces for higher values of $\lambda$.

In addition, we compute these probabilities empirically by counting the number of times when the hash values collide for valid and invalid samples, after the LSH

| $\lambda$ | **150** | **350** | **500** |
|---|---|---|---|
| $P_1$ | 8.6e-1 | 9.3e-1 | 9.1e-1 |
| $P_2$ | 2.8e-6 | 0.0 | 0.0 |

Table 5.10: Average probability of collision, for valid ($P_1$) and invalid ($P_2$) samples in the ai.lock holdout set, when the ai.lock imageprint is considered as image hash value and at most $c = \lfloor \lambda \times (1 - \tau) \rfloor$ bits of error is allowed. In all cases, $P_1 > P_2$, thus conclude that ai.lock is an LSIM function.

transformation. We then use this count to compute the average probability of collision for a valid ($P_1$) and invalid ($P_2$) authentication samples (see Table 5.9). We observe the remarkable similarity of these values, to the ones above, computed analytically. As $\lambda$ increases, the empirical $P_1$ approaches the analytic lower bound (0.79). We perform a Mann-Whitney one-sided test with alternative hypothesis $P1 > P2$. This test suggests that there is a significant gap between $P_1$ and $P_2$ ($p - value = 0.00, \alpha = 0.05$) for all cases, hence, ai.lock is a $\delta$-LSIM on the Nexus holdout dataset.

**ai.lock with multi-bit hash value is a $\delta$-LSIM**

We set $\delta = \tau$, where $\tau$ is the error tolerance threshold obtained from the ai.lock training process (see Table 5.3), for different values of $\lambda$. Table 5.10 shows the $P_1$ and $P_2$ values achieved by the basic ai.lock over the holdout dataset. We perform Mann-Whitney one-sided test with alternative hypothesis $P1 > P2$. Based on the observed $p - value = 0.00$, ($\alpha = 0.05$), for different values of $\lambda$, we conclude that the alternative hypothesis is true, hence, ai.lock is a $\delta$-LSIM function over the holdout dataset.

We can also compute the probability of collision for imageprints in an error tolerant way using the probability of collision for a single bit ($Pr_{singleBit\_collision}$). Let X be the random variable denoting the number of matching bits for a pair

| $\lambda$ | **Avg Pr[**$collision_+$**]** | **Avg Pr[**$collision\_$**]** |
|---|---|---|
| 150 | 8.4e-01 | 5.9e-06 |
| 350 | 9.0e-01 | 4.1e-06 |
| 500 | 9.0e-01 | 2.2e-06 |

Table 5.11: The average probability of imageprints collision for genuine and fake pairs of images in ai.lock holdout set when at most $c = \tau \times \lambda$ error is allowed. ai.lock hash LSH-like property which maps the similar images to binary strings with higher probability of collision.

of imageprints. We assume the extraction of each imageprint bit is an independent random variable with probability $Pr_{singleBit\_collision}$. Therefore, $Pr[X >= \overline{c}] = (Pr_{singleBit\_collision})^{\overline{c}}(1 - Pr_{singleBit\_collision})^{c}$. These values are reported in Table 5.11. We observed that the collision probabilities computed based on ai.lock dataset are very close to those estimated based on probability of collision for a single bit.

Compared to ai.lock with single bit hash value, we observe concatenating multiple hashes enlarges the gap between $P_1$ and $P_2$ values.

### 5.6.6 On the Entropy of Imageprints

We have used the entropy evaluation dataset (see § 5.5.2) to empirically calculate the entropy of the imageprints generated by the ai.lock variants. The empirical entropy of an authentication solution is proportional to the size of the keyspace that the attacker needs to search to find a match for the authentication secret. For biometric information, estimating this size is difficult. In such cases, the entropy can be estimated as $-log_2(\frac{1}{FAR})$ [O'G03]. We performed this study for different values of $\lambda$ and the best parameter choice of ai.lock (see § 5.6.1), using the entropy evaluation dataset.

On the 2 billion image pairs in the entropy evaluation dataset, the FAR of the SLSS ai.lock variant is 0.020% and 0.035% when $\lambda$ is 50 and 500 respectively, for an

| $\lambda$ | **150** | **250** | **350** | **500** |
|---|---|---|---|---|
| DI2E module (Inception v.1) | 0.7 | 0.7 | 0.7 | 0.7 |
| DI2E module (Inception v.3) | 1.9 | 1.9 | 1.9 | 1.9 |
| PCA + LSH module | 0.044 | 0.049 | 0.051 | 0.066 |

Table 5.12: Processing time (in seconds) of SLSS ai.lock modules, for different values of $\lambda$. The performance of the DNN module does not depend on $\lambda$ and is 0.7s for Inception.h5. The combined performance of the PCA and LSH modules increases with $\lambda$ but is under 70ms even when $\lambda = 500$. When using Inception.h5, the overall ai.lock speed is below 0.8s.

entropy of 12.28 bits and 11.48 bits. We have visually inspected several hundreds of image pairs that resulted in false accepts and observed that a significant proportion were due to images that contained the same object type, e.g. ribbons, helmets, etc. This result is not unexpected: the SLSS variant uses only the last hidden layer of Inception.v3 network. Since Inception.v3 is trained for image classification task, it is expected to have similar activations on the last hidden layer for images of the same object type. We expect to eliminate this situation by requiring the match between activations of multiple inception layers (multi layer variant).

The FAR of the MLMS ai.lock variant on the entropy evaluation dataset, for $\lambda$ values of 500 and 150, is 0.0007% and 0.0004% respectively. Therefore, the estimated entropy of ai.lock imageprints is 17.14 and 18.02 bits respectively.

In the cross validation experiments reported in § 5.6.4, MLMS ai.lock achieved best performance (F1 score) when $\lambda = 500$. The calculated FAR from the entropy evaluation dataset is consistent with the cross validation FAR (0.0009%). When $\lambda$ is 500, the FRR of ai.lock MLMS is 5.37%. However, when $\lambda = 150$, MLMS achieves a lower FAR of 0.0006%, for a higher FRR of 7.85%.

### 5.6.7 ai.lock Speed

We have implemented ai.lock using Android 7.1.1 and Tensorflow 0.12.1 and have evaluated its speed using 1,000 images of the Nexus dataset on a Nexus 6P smartphone (Qualcomm Snapdragon 810 CPU and 3GB RAM). Table 5.12 shows the average processing time of the 3 main ai.lock modules for different values of $\lambda$. Independent of the value for $\lambda$, ai.lock's DI2E module takes 1.9s to compute the activations of all the layers of Inception.v3. When using Inception.h5 [SLJ+15] (a smaller network), DI2E module takes 0.7s. The combined PCA and LSH speed increases with the value of $\lambda$, but is below 70ms for $\lambda = 500$. The processing overhead of ai.lock is below 2s and 1s using Inception.v3 and Inception.h5 respectively.

To minimize its impact on user experience on a Nexus 6P, ai.lock needs to use Inception.h5. The most significant processing overhead of ai.lock is on computing the activation of the DNN, which directly depends on the size of the network. Note that compressing the network using the DNN distillation approach [HVD15] can alleviate this overhead. In addition, future device and Inception improvements will likely improve the ai.lock performance and accuracy.

## 5.7 Discussion and Limitations

**Default authentication, revocation and recovery**. If the image based authentication fails a number of times or the ai.lock secret is not available, the authentication falls back to the default authentication mechanism, e.g. text passwords.

**Strong passwords**. ai.lock benefits from users choosing strong, high-entropy and unique objects for authentication. ai.lock can use datasets of images of frequently occurring, thus low entropy, objects and learn to reject similar objects during their registration by the user. Further, the image classification task can be adapted to

detect images belonging to classes of weak, low-entropy authentication objects. In addition, similar to text passwords, users could be encouraged to pick an ordered combination of personal objects for authentication.

**Usability**. Although usability is not the focus of this chapter, we expect ai.lock to share several limitations with face based authentication mechanisms due to their similarities in the form factor. These include susceptibility to inappropriate lighting conditions [BUI+15]. While the FAR of ai.lock remains small under illumination changes, its FRR increases, affecting its usability. However, DNNs are capable of learning representations that are invariant to input changes, e.g. lighting, translation, etc. Thus, the DI2E module of ai.lock can be further fine-tuned to be more resistant to illumination changes. We leave the investigation of such improvement for future work. In addition, we leave for future work investigating alternative, more advanced DNN models and exhaustive search for the choice of layer to be used in DI2E module of ai.lock.

In Chapter 4, we have evaluated the usability aspects of an image based authentication approach, and have shown that (1) the user entry time was significantly shorter compared to text passwords on a mobile device, (2) the participants were able to remember their authentication objects 2 and 7 days after registering them, and (3) the participants perceived object based authentication to be easier to use than text passwords, and were willing to adopt it. As the user interface of ai.lock is similar to Pixie, the directions we identified in § 4.7 for investigating the usability aspects of Pixie, will also apply to ai.lock. Particularly, further studies are required to understand (1) the user choice of the secret objects or scenes and whether it impacts the secret key space, (2) the ability of ai.lock to filter out common or low-entropy images, (3) the scenarios where users are willing to adopt ai.lock authentication and (4) other limitations associated to ai.lock authentication.

**Shoulder surfing**. Similar to face based authentication, ai.lock is vulnerable to shoulder surfing attacks where the adversary captures images of the objects or scenes used by victims. However, ai.lock eliminates remote attacks, e.g., [PLK+12], moves the target away from sensitive body features, and enables users to trivially change their image-passwords. Similar to biometrics, ai.lock can also benefit from liveness verification techniques [RATC17], that ensure that the adversary has physical access to the authentication object or scene, to prevent sophisticated image replay attacks. In addition, in § 5.6.3 we show that the knowledge of the authentication object type does not provide the adversary with significant advantage when launching a brute force attack.

**Multi-factor authentication**. ai.lock can also be used in conjunction with other authentication solutions. For instance, the image password set and authentication steps described in § 2.1.1 can take advantage of a secondary secret (e.g. password, PIN), increasing the number of authentication factors to improve security. To this end, let $r$ be a random salt. We modify $x$ in the fuzzy biometric protection solution outlined in § 5.4.1 to be the randomized hash of the secondary secret computed using salt $r$. Randomized hashing ensures the required formatting and bit length for $x$ can be achieved using key derivation function (e.g. HKDF [KE10]), etc. The random salt $r$ needs to be stored along with the other authentication credentials, i.e. $SS(R, x)$.

**Compromised device**. Our model assumes an adversary that physically captures a victim's device and thus has black-box access to the authentication function. ai.lock is not resilient to an adversary who installs malware on the victim device. Such malware may for instance leverage PlaceRaider [TRCK13] to construct three dimensional models of the environment surrounding the victim, including the authentication object.

Trusted hardware can secure ai.lock and even obviate the need for secure sketches. However, it would reduce the number of devices where ai.lock can be applied. Techniques similar to AuDroid [PSJA15] could be employed to ensure that unauthorized processes or external parties cannot access and misuse the device camera, however, they may still leave ai.lock vulnerable to cache attacks [LGS+16].

## 5.8 Conclusions

In this chapter, we introduced ai.lock, a secure and efficient image based authentication with secure storage of authentication credentials. We have presented a suite of practical yet powerful image based attacks and built large scale attack datasets. We have shown that even under our powerful attacks, ai.lock achieves better entropy than state-of-the-art biometric authentication solutions.

We have implemented an ai.lock in Android using Tensorflow [ABC+16] and shown that it is resilient to attacks. Its FAR on 140 million synthetic image attack samples is $0.2\times10^{-6}\%$. ai.lock was unbreakable when tested with 1.4 billion synthetic credential attack samples. Further, we show that ai.lock is a $\delta$-LSIM function, over images that we collected (see § 5.6.5). ai.lock is fast, imposing an overhead of under 1s on a Nexus 6P device.

ai.lock security can be tuned by changing the length of the binary imageprints ($\lambda$). Longer imageprints can preserve more information about the input images, resulting in better overall performance of ai.lock. However, this cannot be arbitrary large due to the limitations of the current binary error correcting codes. In addition, certain level of security (FAR) versus usability (FRR) can be achieved by adjusting error tolerance threshold ($\tau$).

CHAPTER 6

# CEAL: IMAGE-BASED KEY AUTHENTICATION

## 6.1 Introduction

Phishing attempts [Ram10, RAM+18] often impersonate user trusted contacts (e.g., social networking friends, e-mail contacts) and services (e.g., financial institutions, online markets). Therefore, in today's Internet the ability to verify the authenticity of online contacts or services is of paramount importance.

Public key cryptography is the dominant and reliable method for verifying the identity of an entity over the Internet and in secure end-to-end communications. One central problem to designing a public key encryption system is to facilitate the process of evaluating the authenticity of a binding between a public key and an entity (i.e., its owner).

There are two major approaches for addressing this problem: (1) a public key infrastructure (PIK) in which one or more centralized third-party Certificate Authorities (CAs) certify the authenticity of pairs of key and their ownership, and (2) a web of Trust (WoT) which decentralizes the task of authenticating public keys by relying on a chain of individual endorsements (i.e., signatures) to the link between the owner and public key. Numerous security incidents, e.g. DigiNotar [Fis12], TrustWave [Con12], have shown the vulnerabilities associated with failure of centralized CAs. On the contrary, WoT benefits from being independent of any central point of failure. However, its deployment raises several usability issues including challenges in verifying the keys for the first time and issues with recovering the keys [FVY14].

Nevertheless, to address public key authentication problem for decentralized systems without pre-defined authorities (e.g., SSH [GT06], OpenPGP [CDF+07], and secure messaging applications [UDB+15]) manual key verification of the key by a hu-

man verifier is used. Public keys are long strings of arbitrary bits and this makes the process of comparison difficult for the human verifier. In practice, *key fingerprint*, short hashes of the public keys are used to simplify this process for users: A user transfers the reference key fingerprint corresponding to her public key to another contact through a reliable out-of-bound channel, e.g., a secure key server, trusted web site, etc. To authenticate this user, a contact can manually compute the key fingerprint corresponding to the contact's key and compare it to the reference key fingerprint (see Figure 1.2).

Recent study by Tan et al. [TBB+17] have shown that the Visual Key Fingerprint Generation (VKFG) solutions, that represent the key fingerprint using an image (e.g. Vash [vas14]), can increase usability and attack resistance of the key fingerprint verification. Yet, 10% of the generated attack images that they generated for Vash [vas14], when modeling a similar adversary as we describe in this chapter, were missed by human verifiers.

We introduce CEAL (CrEdential Assurance Labeling), a novel approach to generate visual fingerprint representations of cryptographically strong public strings. CEAL's generated images (i.e. *ceal*) stands out from existing approaches in three significant aspects: i) involves a learning step where the target style and domain of the fingerprint images are captured into a generator model from a large collection of sample images rather than hand curated as a collection of rules, hence providing a unique capacity for easy customization, ii) integrates a model of the visual discriminative ability of human perception so the resulting fingerprint image generator avoids mapping distinct keys to images which are not distinguishable by humans, iii) deterministically generates visually pleasing fingerprint images from an input vector where the vector components are designated to represent visual properties which

are either readily perceptible to human eye, or imperceptible yet are necessary for accurately modeling the target image domain.

**Challenges.** To build CEAL, we need to address several important challenges:

• **Human distinguishability**. Due to the wide variety in the visual systems of the humans, identifying the space of human distinguishable images is a difficult task. However, humans are good at identifying shapes, colors and objects [Wil66]. In addition, the human visual system is better at distinguishing changes in images when their content is more natural [PTT00]. We exploit these properties to generate realistic images that are distinguishable by average humans.

• **Generating human distinguishable images**. While using GAN [GPAM+14] would ensure that generated images are realistic, thus easy to compare, our experiments have shown that not all the components of the input to GAN (i.e. the input latent vector), will result in a perceptible change in the generated images when modified. Furthermore, the perceptability of a change could depend on the values of the other components (see § 6.5.2). This severely limits the application of GAN to fingerprint image generation, as the indistinguishable images mean that an attacker can forge a key that has a fingerprint image similar to the authentic one, thereby successfully spoof the human verification.

To address this problem, we employ a mechanism that efficiently evaluates the distinguishability of the generated images by the generator network during training of a GAN and provides a feedback for the generator to generate images that are human distinguishable. While ideally one would use humans to verify the satisfaction of this requirement, this process does not scale well.

• **Automatic classification of human distinguishability**. The above challenge suggests the need for an automatic solution. For this, we build a Human Perception Discriminator (HPD), a classifier that can predict whether two images are perceived as distinct by human verifiers.

However, training a classifier that accurately predicts the perception of all human visual systems, using limited human labeled datasets that we collect, is a difficult task. Specially, given the wide variety in the visual systems of the humans who will compare these images in real life (range of ages, visual acuity, color blindness, etc). Instead, we settled to build an Human Perception Discriminator (HPD) that only has high precision: if it predicts that two images are different, they will be perceived to be similar by human verifiers, only with a very small probability. We show that even with such a HPD, that may have a lower recall, we are able to train a GAN that satisfies our requirements.

• **Input mapping impact on human-distinguishability of generated images**. Our experiments with Vash, state-of-the-art VKFG [TBB+17], revealed that not all the bits of the input string, when modified, result in a human-perceptive change in the generated images (see § 6.7.4). To ensure all ceal images that are generated, using any input string, are human distinguishable, we conjecture that it is possible to build a GAN with a special latent vector. Particularly, when a subset of latent vector components (called *major components* are changed even individually, it can result in human-perceptible changes in the generated images, while the other (*minor components* cannot individually produce such changes and encode relatively imperceptible characteristics of the images. We built the constraints of major and minor components into CEAL training procedure, which not only decomposed these components in the latent vector, but also pushed the efficiency of the major components to encode larger keys. We then use the major components to estimate the capacity of CEAL.

• **Capacity**. In addition to the human aspects, the security of a key fingerprint depends on the encoding *capacity* of a solution, i.e., the max number of unique distinguishable images the algorithm can generate, where the larger capacity solution

is stronger against attacks. CEAL is able to push the key payload capacity of the images to the limit of human perception. The separation of the major and minor components allows the use of error correction codes to properly encode the input vectors to CEAL into a representation that will guarantee the generated image to be human distinguishable (see Definition 6.4.2).

**Implementation and evaluation.** We implemented and trained CEAL using Tensorflow [ABC+16]. We then show that ceal images are distinguishable and it is computationally hard for even powerful adversaries to find a collision. We run brute force attacks using 156 million attack images generated for 79 million target inputs. We then use HPD to identify likely successful attack images along with their broken targets. We use MTurk to label these images. Out of 308 potential attack samples we identified, only 0.97% was missed by human.

In addition, the human verifiers can quickly compare ceal images: on average, it took 2.04s for our workers to compare similar (attack) pairs of ceal images.

## 6.2 Model and Applications

### 6.2.1 System Model

We consider a key fingerprint based authentication scenario where every identity represents his keying material or online identify (e.g. email address, IP address, Bitcoin account, etc.) using an image (i.e. key fingerprint of his key or identity). To authenticate the identity, one should obtain the key fingerprint of the contact in advance through a secure channel. Upon authentication verification, the user computes the key fingerprint of the online identify and compares it to his reference (see Figure 1.2).

### 6.2.2 Applications

We now discuss several applications of visual key fingerprint solutions.

**Bitcoin Clipboard Attack Prevention**. Visual key fingerprints can prevent clipboard hijacking attacks performed on Bitcoin users [Sub18]. In this type of attack, a malware gains access to the clipboard of a user while he copy-pastes a Bitcoin address, and replaces it with the attacker's address. Key fingerprints for Bitcoin addresses can prevent this attack: The user compares the fingerprint of the copy-pasted address with the reference of the recipient's address.

**Phishing Attack Prevention**. Visual key fingerprints can also provide both server and contact authentication in online communications. Social network, e-mail, and financial service providers can use visual key fingerprints to prevent phishing attacks, by allowing their users to authenticate both the service and their contacts on the site. For instance, CEAL can provide a visual clue of the identity of a website that is visited by a user, i.e., the domain names of website.

**Authentication in E2EE Apps**. Key fingerprint solutions can be used for authentication in different online system such as End-to-End Encrypted (E2EE) applications on smartphones (e.g., WhatsApp [Wha], Viber [Vib], Facebook messenger [Con16]). To authenticate the other party in the communication, the user needs to manually 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 sites, etc).

**Avatars**. Similar to identicons [Par07], visual key fingerprints can also be used to represent unique avatars for users of online wiki pages, forums or who post blog comments. For instance, the email, IP address, or the browser fingerprint of the user can be used to generate a ceal for her identity, while helping preserve the user's identity and improve the user experience of the web site visitors.

**Device Pairing**. Visual key fingerprints can be used to pair devices (e.g., Bluetooth Secure Simple Pairing using ECDH [Pad17]): the user can confirm the identity of the other device by verifying the ceal corresponding to the device's key.

**Security Indicators**. A visual key fingerprint generated by CEAL can provide a visual password hint for users while they are typing their password: the ceal corresponding to the user's password is shaped as the user enters the character and the user can verify if he has entered the right password without requirement to display the password on the screen.

**File Integrity Check**. Key fingerprints can provide a more usable alternative for checking the integrity of files downloaded from the Internet. Instead of comparing hash values (of the downloaded file and a reference from a trusted site), the user will compare their fingerprints.

### 6.2.3 Adversary Model

We assume an adversary who attempts to generate input keys whose visual fingerprints will be perceived by a human verifier to be similar to the fingerprint of a specific victim (see Figure 6.1). We assume that the adversary has blackbox access to the VKFG function. While the adversary can brute-force search the input space, we also consider a $(\gamma, d)$-adversary, similar to that of Dechand et al. [DSB$^+$], who picks candidate strings within Hamming distance $d < \gamma$ to the victim's key $K$. The adversary can then apply VKFG to the candidate strings, to generate attack visual fingerprints.

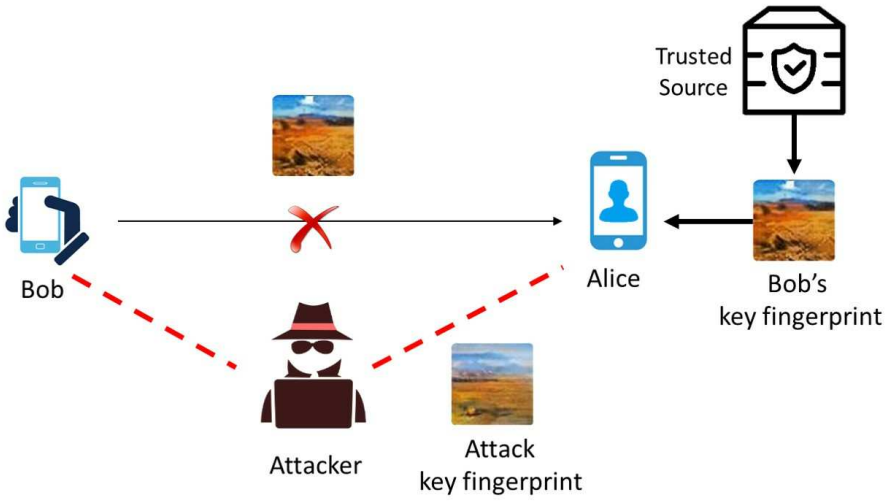



Figure 6.1: Adversary model: Let's assume that Alice is trying to verify the identify of his contact, Bob. For this, Alice computes the fingerprint of the public key of the contact and compares it to a trusted reference of Bob's key fingerprint that she has obtained previously through a secure out of band channel. However, the adversary can perform a man-in-the-middle-attack. Particularly, the adversary attempts to impersonate the victim (Bob), by using a public key whose corresponding fingerprint image will be perceived to be the same as that of the victim (Bob).

## 6.3 Problem Definition

Informally, we seek to construct a set of images, where each image can be 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 the set of images that we generate. In the following, for simplicity, we also refer to input strings as *keys*. This will allow us to represent a given input string with an image, which will not be confused for another input's image representation. For practical applications, 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 string 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 set of RGB images $\mathcal{I}$, and a function $HPD_{ratio} : \mathcal{I} \times \mathcal{I} \rightarrow [0, 1]$ 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 $I_i, I_j \in \mathcal{I}$, $P_u^{i,j} = 1$ if and only if the human perceives the images to be different, $P_u^{i,j} = 0$ otherwise. Then, if $h$ is the number of human experiments conducted per each image pair, $HPD_{ratio}(I_i, I_j) = \frac{\sum_{u=1}^{h} P_u^{i,j}}{h}$.

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

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

In practice, it is very challenging to build a generator that satisfies the VKFG requirement for all possible human visual systems. However, having access to a $HPD_{ratio}$ function, would immediately 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 perception experiments for any given pair of images. However, given a sufficient number of annotations, a regression predictor model $HPD_{predict} : \mathcal{I} \times \mathcal{I} \rightarrow [0, 1]$ may be used to approximate the $HPD_{ratio}$ function, $E(|HPD_{predict}(I_1, I_2) - HPD_{ratio}(I_1, I_2)|) < \epsilon$. We show that, even when a small number of annotated data is present, a very limited classification model $HPD_{equal} : \mathcal{I} \times \mathcal{I} \rightarrow \{0, 1\}$ which can detect distinguishable image pairs with high precision at the cost of low recall, $P(HPD_{ratio} > 0 \mid HPD_{equal}(I_1, I_2) = 1) < \epsilon$, is sufficient for training a generator which satisfies the VKFG requirement

### 6.3.1 Requirements for a Key Fingerprint Generator

Here, we briefly summarize the requirements for a VKFG function.

• **Human-distinguishability of fingerprints**. Any pair of fingerprint images that can be mapped from the key space should be distinguishable by humans.

• **Capacity**. To be resistant against attacks, the solutions needs to have sufficiently large *capacity*, i.e., the number of unique distinguishable images that the VKFG can generate should be sufficiently large.

• **Ease of comparison**. Humans should be able to quickly compare any generated images for equality.

## 6.4 The CEAL System

In practice, it is very challenging to build a generator that satisfies the VKFG requirement for all possible human visual systems (see § 6.3). Instead, we propose to build a weak visual key fingerprint generation function $VKFG_{weak} : \{0,1\}^{\gamma'} \rightarrow \mathcal{I}_{\mathcal{W}}$, where $\mathcal{I}_{\mathcal{W}} \subset \mathcal{I}$. Let $d_H$ denote the Hamming distance. The $VKFG_{weak}$ is not able to guarantee that key pairs will be distinguishable if their $d_H$ is within $d$, $E(HPD_{ratio}(I_i, I_j) \mid d_H(K_i, K_j) < d) < 1 - \epsilon$. However, for key pairs whose $d_H$ value is at least $d$, $VKFG_{weak}$ is able to guarantee human distinguishability, i.e., $\forall K_i, K_j \in \{0,1\}^{\gamma'}, d_H(K_i, K_j) \geq d \iff HPD_{ratio}(I_i, I_j) = 1$, where $I_i, I_j \in \mathcal{I}_{\mathcal{S}}$, $VKFG_{weak}(K_i) = I_i$, and $VKFG_{weak}(K_j) = I_j$. Therefore, our problem reduces to building such an instance of $VKFG_{weak}$ and identifying the minimum value for d that satisfies the above requirements.

We show that it is possible to build a VKFG using a $VKFG_{weak}$ with the help of an error correcting code encoder, $ECC$. Let $ECC : \{0,1\}^{\gamma} \rightarrow \{0,1\}^{\gamma'}$, with minimum distance of d, hence $\forall K_i, K_j \in \{0,1\}^{\gamma}, K_i \neq K_j \implies d_H(ECC(K_i), ECC(K_j)) \geq$

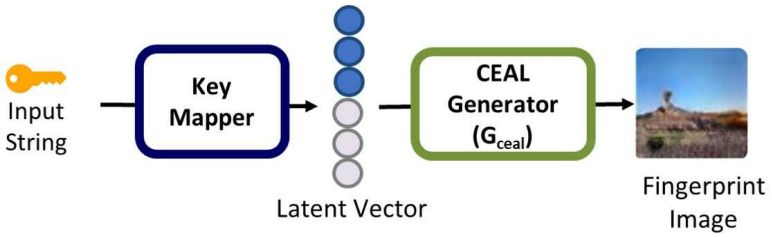



Figure 6.2: CEAL System. CEAL accepts as input a binary string and used input mapper module to map the input to the input latent vector components to $G_{ceal}$). The generator network, then generates the visual key fingerprint (ceal) corresponding to the input string.

$d$. We can apply the $ECC$ to the input and apply $VKFG_{weak}$ to the encoded string, which makes sure that the input to $VKFG_{weak}$ are always human distinguishable by definition of the $VKFG_{weak}$. Therefore, $VKFG_{weak} \circ ECC : \{0,1\}^{\gamma} \rightarrow \mathcal{I}_{\mathcal{W}'}$, where $\mathcal{I}_{\mathcal{W}'} \subset \mathcal{I}_{\mathcal{W}}$, and $\forall I_1, I_2 \in \mathcal{I}_{\mathcal{W}'}, HPD_{ratio}(I_1, I_2) = 1$.

We introduce CEAL (CrEdential Assurance Labeling), a VKFG function that uses a GAN [GPAM+14] to generate realist images and address the requirements of § 6.3.1. CEAL has two major components: (1) CEAL DCGAN a DCGAN [RMC15] network with an additional discriminator, i.e. a human perception discriminator; (2) Key Mapper (KMap). The process of generating a ceal image for an input string is depicted in Figure 6.2. Let $K$ be the input (e.g. (truncated) hash of the user's key, i.e. its binary fingerprint). Let $\gamma = |K|$. The KMap module in CEAL converts the input key into a latent vector $L$. Let $\lambda = |L|$, $\gamma < \lambda$. The latent vector is then used as input to the generator network of CEAL DCGAN ($G_{ceal}$). It then generates ceal corresponding to the input string.

To thwart adversaries who can generate $K'$ to be at small Hamming distance from $K$ (see § 6.2.3), we design CEAL to generate image fingerprints that are visually different even when the keys are similar. For this, we define the following image pair generation (IPG) process, that takes as input a *seed latent vector* with length $\lambda$ and

an index $i \in \{1, 2, 3, ..., \lambda\}$, and outputs two vectors $v_1$ and $v_2$, also of length $\lambda$:

**Definition 6.4.1** *(***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 the extreme values of each component. We use these values to maximize the effect of a component in generated images when generating ceals.*

We further introduce the following conjecture:

**Conjecture 6.4.2** *(***Major and Minor Components***). We conjecture that a subset of the latent vector components, when changed individually, can produce perceptible changes in the generated images. We call these "major components" of the latent vector. Further, we conjecture that a disjoint subset of the latent vector components, when changed individually, do not produce perceptible changes in the output images.*

According to Conjecture 6.4.2, we further conjecture that only the major components contribute to the entropy or capacity of the CEAL VKFG function, while the minor components can help CEAL generate realistic images or maintain other visual aspects of the image.

Let $M$ be a system parameter, the number of major components of the latent vector. Thus, the number of minor components is $m = \lambda - M$. We select the values for the $M$ major components 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 $(-1, 1)$.

In the following, we use CEAL to denote the system and *ceal* to denote its output image for a given user input. We now describe each module of CEAL.

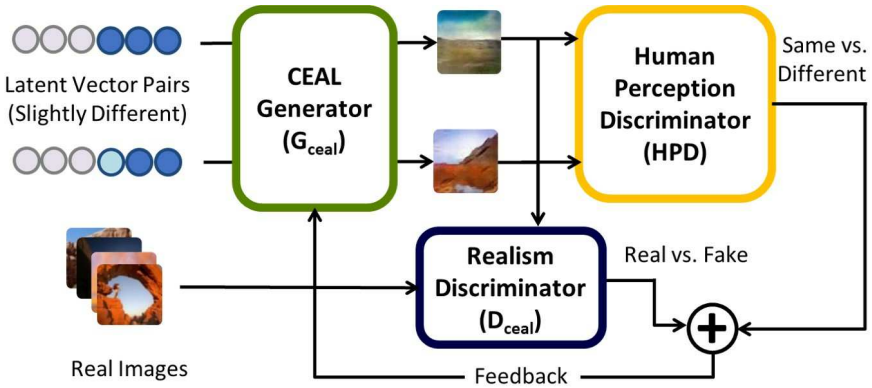



Figure 6.3: CEAL DCGAN architecture and training. We use the combination of Discriminator loss and HPD loss to train the generator to generate distinguishable and realistic images. We also learn a latent vector that consist of major and minor components (see Conjecture 6.4.2).

### 6.4.1 CEAL DCGAN

We introduce CEAL DCGAN, a DCGAN [RMC15]-based deep generative model (see Figure 6.3). CEAL DCGAN architecture is similar to the architecture of DCGAN [RMC15]. The heart of the CEAL DCGAN is a generator network, i.e., the CEAL generator ($G_{ceal}$), that can generate realistic and human distinguishable images. For an input key ($K$), we use KMap (§ 6.4.2) to transform the binary key fingerprint of a $K$ into the major and minor components, that are then concatenated to form the input latent vector to CEAL generator. The $G_{ceal}$ then generates the image (i.e. ceal) corresponding to $K$.

We train the generator network using two classifiers (see Figure 6.3): (1) the CEAL discriminator ($D_{ceal}$) that is trained to differentiate between real images, from a dataset of images, and synthetically generated images by $G_{ceal}$; (2) HPD classifier that is trained to estimate the likelihood that a human will label a pair of images as either same or different.

We train the discriminator network of CEAL DCGAN similar to conventional GAN using a real dataset of images (see § 3.1.5). However, Human Perception

Discriminator (HPD) is a classifier we train to estimate the human distinguishability of the image pairs ($HPD_{ratio}$), see § 6.3. The output of this classifier is referred to as $HPD_{predict}$. In the following, we first describe the HPD. We then describe how we used trained HPD along with $D_{ceal}$ to train CEAL DCGAN to generate realistic and distinguishable images.

### 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.

**The HPD Architecture and Training Process**. We build HPD using a DNN. The high level architecture of the HPD classifier network, illustrated in Figure 6.4, is similar to a Siamese network [CHL05]. 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 trained Inception.v1 [SLJ+15] network (see § 3.1.4). It then extracts $50,176$ image features i.e., the activations of the 'Mixed_5c" layer of inception.v1. In § 6.6.1 we experimentally justify the choice of the layer. Following the Inception.v1 network, HPD adds to both of its twin networks, several additional fully connected layers.

To train the HPD network, we do not update the weights of Inception.v1 layers. However, we optimize the weights of the (three) additional fully-connected layers, using weighted contrastive loss [CHL05] with L2 regularization. The purpose of this loss is to enable the network to differentiate between the two images and regularization is used to prevent overfitting. Equation 3.1 shows how the weights are updated based on the weighted contrastive loss for two input samples X1 and X2.

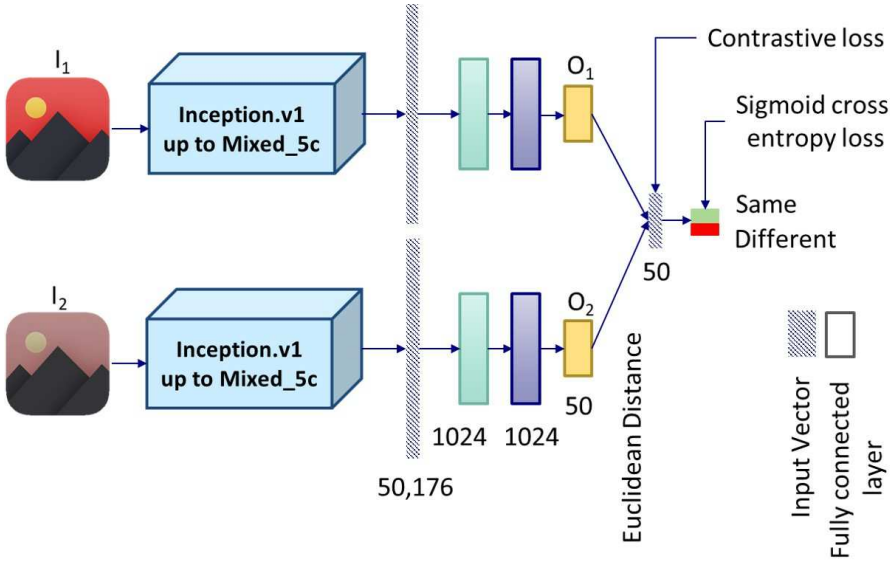



Figure 6.4: Human Perception Discriminator (HPD) architecture. HPD passes input images $I_1$ and $I_2$ through the Inception.v1 network, applies 3 fully connected layers to generate image feature vectors $O_1$ and $O_2$, computes the Squared Euclidean distance between $O_1$ and $O_2$ and passes it through a fully connected layer to the computed distance. HPD classifies $I_1$ and $I_2$ as different or same based on this distance.

After training the (three) additional layers in the twin Siamese network, we freeze the network weights and feed their derived output, i.e., the component-wise squared differences between the last layers of the networks for the input image pair, to an additional fully connected layer (here, with 1 neuron, i.e. HPD output, with sigmoid activation function). We optimize this layer's weights using well known weighted cross-entropy loss and L2 regularization of the layers weights. We train this layer to classify the image pairs into either of "same" or "different" classes, based on the squared Euclidean distance between the image pair features that is obtained from the Siamese network. As we describe in § 6.6, we decide the choice of architecture (including the number of layers and nodes in each layer) through hyper-parameter search.

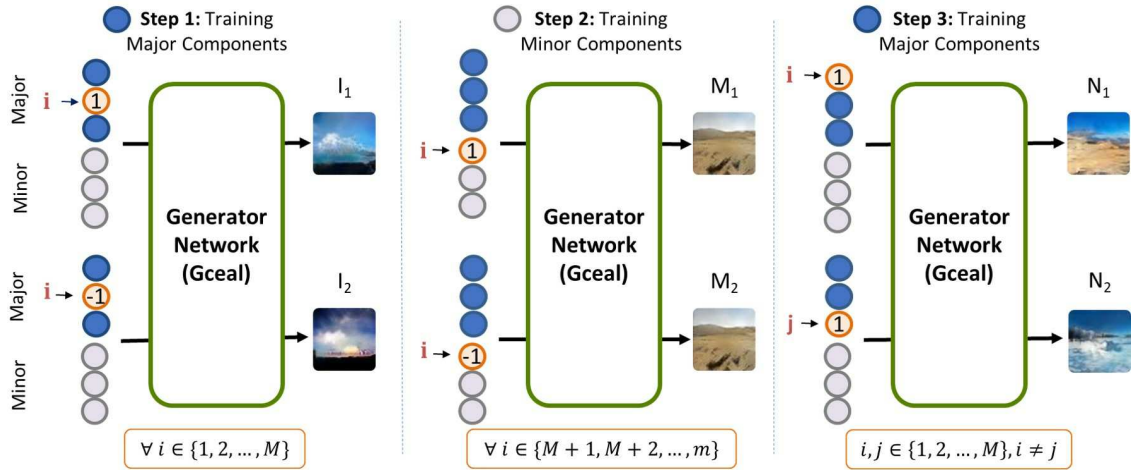



Figure 6.5: Image pairs generated while training CEAL: Step 1 focuses on promoting CEAL to generate different images when a major component value is flipped (set to 1 and -1). Step 2, focuses on promoting minor components to not change the visual characteristic of generated images when their values are modified. Step 3, prompts CEAL to generate diverse set of images by further training it using major components. (see § 6.4.1).

### Training CEAL DCGAN

While the DCGAN [RMC15] generates realistic images, we train CEAL DCGAN to generate images that are both realistic, and visually distinguishable by human. To achieve this, in each training epoch, we train $G_{ceal}$ in 3 steps, shown below. In each step, we randomly create a set of latent vector pairs, that we generate from a set of random seed latent vectors whose components are uniformly selected from (-1,1). We then use $G_{ceal}$ to generate the corresponding image pairs. Similar to DCGAN, we train $G_{ceal}$ using the output (real/fake) of discriminator ($D_{ceal}$) for the generated images by $G_{ceal}$. In addition, we use the HPD (§ 6.4.1) to compute the $HPD_{predict}$ corresponding to each image pairs. We then use the $HPD_{predict}$s as a feedback to $G_{ceal}$ about the visual characteristics of the images that it generated: adjusts the weights of $G_{ceal}$ based on an objective that is a function of $HPD_{predict}$.

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 a general form as shown in Equation 6.1. Therefore 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 $D_{ceal}$.

$$L(\theta_{G_{ceal}}) = \alpha \times HPD_{loss} + G_{loss} \tag{6.1}$$

**Human Distinguishability**. We leverage the input latent vector to control the visual characteristics of the images generated by $G_{ceal}$. Specifically, we train $G_{ceal}$ 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 § 6.3).

Let $M$ and $m = \lambda - M$ be the number of major and minor components in the latent vector input to $G_{ceal}$. We now describe each of the 3 steps, i.e., how we generate the 3 sets of latent vector pairs, and the $HPD_{loss}$ function that we use in that training step. Each latent vector pair that we generate, is different in $d = 1$ or $d = 2$ specific major or minor components.

• **Step 1**. Generate $M$ random seed latent vectors. Then, for each index $i \in \{1, 2, 3, ..., M\}$, use the IPG($i$) of Definition 6.4.1, 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, we define the $HPD_{loss}$ for the pair of images to be: $HPD_{loss}(v_1, v_2) = cross\_entropy(1, HPD_{predict}(I_1, I_2))$.

• **Step 2**. Generate $m$ random seed latent vectors. For each minor position $i \in \{M+1, M+2, ..., \lambda\}$, form sample latent vector pairs $v_1$ and $v_2$ as in Definition 6.4.1. 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 same (class 0). To achieve this, we 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.** Generate one batch of random seed latent vectors (here, 64). For each 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 $v_2[j] = 1$. Thus, $v_1$ and $v_2$ only differ in the $i$-th and $j$-th components. Let $N_1$ and $N_2$ be the images that are generated by $G_{ceal}$ from $v_1$ and $v_2$ respectively. We define the loss of the generator as $HPD_{loss} = cross\_entropy(1, HPD_{predict}(N_1, N_2))$. This step seeks to train $G_{ceal}$ to use any 2 major components to impose different effects on the visual characteristic of generated images.

**Realism**. In each epoch, the discriminator is also trained similar to conventional DCGAN to discriminate between the real images from a particular dataset and synthetic images generated by $G_{ceal}$ in all the 3 steps above. Subsequently, $G_{ceal}$ is trained using the classification signal provided by $D_{ceal}$: we included this signal as $G_{loss}$ in the overall loss function used for training $G_{ceal}$ in each step (see Equa-

tion 6.1). This process encourages $G_{ceal}$ to generate previously unseen images, that look like the images in the real image dataset, thus deceive $D_{ceal}$ to classify them as real images.

### 6.4.2 Key Mapper (KMap)

KMap takes as input user data (e.g., public key, shared key, Bitcoin address, IP address, domain name) and outputs a latent vector $L$ of length $\lambda$. For this, KMap first computes a cryptographic hash of the input to produce $K$, its binary key fingerprint, of length $\gamma$.

To generate the major components of the latent vector $L$, KMap employs an error correcting code with encoder $E$ (see § 3.3) that encodes a binary string of length $\gamma$ into a code word of length $\leq M$ (i.e., the number of major components). Specifically, KMap computes $E(K)$, then performs a one-to-one mapping between the bits of the code word $E(K)$ and the major components of $L$: $L[i]$ = -1 if $E(K)[i] = 0$ and $L[i] = 1$ if $E(K)[i] = 1$, $i = \{1, 2, 3, ..., M\}$. The indices of these components in the latent vector are arbitrary selected. If $|E(K)| < M$, we set $L[i]$ = -1 for $M - |E(K)|$ other $i$ positions of the major components.

The error correcting code enables KMap to compensate for the training limitation of the generator network of the CEAL DCGAN (§ 6.4.1) and fine tune the distinguishability of ceal images that it generates (see § 6.6).

KMap uses then a pseudo random number generator $R$ seeded with $K$, to randomly select the values for $m = \lambda - M$ minor components of $L$: $L(i) \in U(-1, 1)$, $i \in \{M + 1, M + 2, M + 3, ..., \lambda\}$.

## 6.5 Data

To evaluate CEAL, we use several datasets of real and synthetically generated images. We describe them in the following sections.

### 6.5.1 Real Outdoor Image Dataset

We use a subset of 150,113 outdoor landscape images (mountains, ocean, forest) of 64 by 64 pixels, from the MIT Places205 dataset [ZLX+14, Out18]. In addition, we manually collected 35 additional images that represent outdoor scenes using Google image search. We selected images that include only a few objects and colors, e.g., horizon and landscapes. These are used as obviously same images in our surveys (see § 6.5.2).

### 6.5.2 Ground Truth Human Perception Dataset

We train a DCGAN network with random uniform input latent vector of length $\lambda = 100$, using the real outdoor image dataset of § 6.5.1. We stopped training the network when we started to observe realistic images similar to the ones in training dataset (after 10 epochs). We refer to this trained network as "vanilla DCGAN".

We generated two datasets of synthetic image pairs using vanilla DCGAN and collected their labels using MTurk workers. For this, 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 either "same" or "different" images. After analyzing the workers responses, 318 image pairs were labeled as different and 240 pairs were labeled as same. In following, we describe the two labeling processes that generated this dataset.

**Participants.**. We collected labels from 500 human workers: 337 female and 163 male, with an age range of 18 to 84 (M=44.54, SD=14.64). 85.5% of our participants had college education or higher. 220 (40.5%), 251 (50.3%), 26 (5.2%), 21 (4.0%) participants used a desktop, laptop and mobile device, or tablet to answer the surveys respectively.

### Labeling Process 1

We used the vanilla DCGAN network to generate 100 synthetic "different" image pairs using 100 random seed latent vectors ($v$) and IPG of Definition 6.4.1 for $i \in \{1, 2, 3, ..., 100\}$. We assume that the images in each such pair are perceived as being "different" by humans.

In addition, we generated 40 identical image pairs: 32 pairs from the real outdoor scenes image dataset (§ 6.5.1) plus 8 pairs from randomly selected synthetic images among the above 100 image pairs of the previous step. We used proportional sampling to divide the total of 140 image pairs (100 "different", 40 "same") into 4 groups of size 35 (25 assumed "different", 10 assumed "same"). We then recruited 4 groups of 100 different MTurk workers (400 workers in total) and asked each group of 100 workers to label each of the 35 image pairs in one of the groups. Thus, each image pair received 100 labels, one from each worker to which the pair was shown; each worker labeled 35 image pairs.

To avoid collecting low quality labels from inattentive workers, we have included an attention test (a.k.a. golden task [LWZF17]) at the beginning of the surveys. We did not collect the labels from workers who failed to answer the attention test correctly. In addition, we removed the responses from speeders [GMS15], i.e., workers who completed a survey in less than a minute, which is about one standard error less than the average worker response time. We also removed the answers from workers

who made more than 10 errors (or 10% error) 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. Subsequently, we have the labels from at least 94 workers for each image pair.

We then assigned to each image pair its assumed ("same" or "different") label, only if more that 90% of the worker responses agreed with it. Otherwise, we assigned the opposite label. This is because we wish to have high confidence for the image pairs labels. By this choice, we will be conservative in the case of "different" images: we don't want to have an image pair labeled as different if not almost all of our workers agreed. Consequently, 75 and 65 of the image pairs were respectively labeled as different and same by our workers.

**Verification Device**. The device on which the comparison is taking place does not have a significant affect on user performance and time to compare image pairs. Particularly, we studied the quality of responses collected from 400 MTurk workers in the DCGAN image labeling Process 1. 160, 201, 21, and 18 participants used a desktop, laptop, mobile phone, or tablet to complete the surveys respectively. A Kruskal-Wallis test, did not show a significant difference between the number of errors made (w.r.t. the hypothetical labels) by participants responding to surveys using either of four devices, i.e., desktop, laptop, mobile phone, and tablet, to complete the surveys (P-value = 0.93). We also did not observe any significant difference between the overall time it took for the participants using different devices to complete our surveys (P-value = 0.06).

### Labeling Process 2

Following the Labeling Process 1 (§ 6.5), we identified the index of 3 random components in the input latent vector to vanilla DCGAN whose corresponding generated

images were labeled with relatively high error rates by workers (respectively, 52%, 27% and 20%). This error rate was calculated with respect to the hypothetical labels we assumed for images. We then performed a second labeling experiment, to determine if the error rate we observed was due to the fact that the component always produces indistinguishable image pairs when its value is flipped or this is due to other factors, e.g. the contribution of all the other components on what the image looks like.

First, for each of the 3 image pairs with the relatively high error rate in labeling Process 1, with hypothetical label of "different", we generate 99 variant image pairs as follows: Let $j$ be the index of the component that we flipped to generate this particular image pair in Process 1 (which resulted in a high error rate). Also, let $v$ be the seed latent vector (see Definition 6.4.1) corresponding to this image pair. For all $i \in \{1, 2, 3, ..., 100\}$ index values, where $i \neq j$, we use the IPG of Definition 6.4.1 to obtain two copies of $v$ that only differ in the $i$-th component, then use the vanilla DCGAN to obtain an assumed "different" image pair. In total, we generate 297 ($99 \times 3$) image pairs that are hypothetically different.

Second, for each random 10 components (inducing previous 3 component) with relatively high error rate in Process 1, with hypothetical label of "different", we generate 10 image pairs using a new seed latent vector randomly. We obtain 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 generate 100 image pairs.

Further, we used a total of 49 unique hypothetically same pairs in this study: 28 from real and synthetic images that were labeled correctly by a majority of the workers in Labeling Process 1, 3 real images that we collected from Google images (see 6.5.1) and 18 synthetic images randomly selected from the above Process 1.

We selected the image pairs for each survey as follows. We split the 397 assumed "different" and a random subset of size 10 from 49 "same" image pairs into 10 different sets, each for a different survey. Except for one residual survey, each survey consists of 50 image pair comparisons: 30 image pairs out of 297 image pairs of first step, 10 image pairs out of 100 image pairs of second step, and 10 assumed same image pairs. We asked 10 MTurk workers to label the image pairs in each set as either "same" or "different" (total of 100 workers).

As before, we eliminated the labels provided by speeders and the workers who failed the attention check at the beginning of the surveys. In total, we removed responses from 13 workers. Then, for each image pair, we assigned it the assumed "different" or "same" 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. In total, 243 images were labeled as different, while 203 image pairs were labeled as same. We found no disagreement between the labeling results for 28 hypothetically same image pairs that were common in Process 1 and 2.

The Spearman correlation test did not reveal any significant monotonic correlation between the error rate for components in Process 1, and image pairs corresponding to these components, in both experiments. Therefore, we conclude that the visual characteristics of a generated image is determined by a combination of effects of each component in the latent vector.

### 6.5.3 HPD Classifier Dataset

In order to train the Human Perception Discriminator (HPD) classifier, we have generated 6 different datasets of synthetic image pairs, containing a total of 26,802 image pairs, including the labeled image pairs from ground truth human perception

| Dataset Name | # pairs | Similarity |
|---|---|---|
| Labeled Synthetic Image Pairs | 558 | 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 6.1: Size of 6 generated image pair datasets, of either "same", "different" or "mixed" image pairs, used to train the HPD classifier.

dataset. Table 6.1 lists these datasets and their corresponding number of image pairs. In the following, we describe each dataset.

**Set 1: Labeled Synthetic Image Pairs.** This dataset is the gold standard human perception labeled dataset of § 6.5.2 (318 "different" and 240 "same" image pairs).

**Set 2: Unrealistic DCGAN Image Pairs.** In order to train the HPD to correctly classify visually similar, but random noise images, as "same" 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. We label each pair as "same". That is, we wish to train the HPD classifier to classify these image pairs as being the same, as this is how a human verifier will see them (gray images with random noise).

**Set 3: Minor Change in Latent Vector**. To increase the number of synthetic "same" image pairs in the synthetic datasets, we chose a random seed latent vector

and (1) used it to generate one image of the pair and (2) chose a random component of the seed latent vector and multiplied its value by $c \in [0, 1]$, 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 manually sampled and verified that these image pairs look the same.

**Set 4: Blob Image Pair Dataset**. First, we generated 20 different blobs of random shapes and colors. Then, we generated 1,000 realistic images using the vanilla DCGAN model using random input latent vectors. We then form 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 compute the Delta E CIE 2000 [SWD05] score, representing colors that are perceived to be different by humans [Sch11]. We accept the composite image if this score exceeds 50. In total, we generated 2,108 "blob" image pairs.

**Set 5: 10%-different Image Pair Dataset**. We generated 1,024 different image pairs as follows: generate a random seed latent vector, copy it to $v_1$ and $v_2$, select 10 random latent components (out of 100) 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, where we found 10% to be the smallest percentage of difference that resulted in always distinguishable image pairs.

**Set 6: 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 either of 5 enhancements, change (1) image

| 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 | 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_model_2 | - | 0.78 | 0.17 | 0.54 | 0.04 | 0.62 | 0.93 | - | 0.004 | - | - | 0.12 | - | 0.63 | 0.001 | 0.54 |

Table 6.2: Performance of the best HPD classifier and its underlying Siamese-like network, over different HPD classifier datasets.

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

## 6.6 Implementation

We have built CEAL in Python using Tensorflow 1.3.0. In this section, we describe the process we used to identify the parameters for which CEAL components,HPD, CEAL DCGAN and KMap, performs best. In the case of the first two components, we discuss the networks training and hyper parameter tuning.

### 6.6.1 HPD Training and Parameter Choice

**Inception.v1 Layer Choice**. We experimented with using activations of different layers of the Inception.v1, for image feature extraction in HPD (see 6.4.1). Specifically, we performed 200 runs of each 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 kept the architecture and initial weights of the fully connected layers weights in HPD identically. We then trained each of the 3 networks for 1000 epochs. We repeated this process 200 times. We then compared

the performance of trained classifiers using either of these 3 sets of features, 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 use the 6 datasets of § 6.5.3 to train and evaluate HPD. Particularly, we randomly split each synthetic dataset (except the Labeled Synthetic image pairs), into training ( 80% of samples) and holdout ( 20%) sets: we use the training sets to train the HPD classifier, then test its performance over holdout sets. For the Labeled Synthetic image pairs dataset, we make sure the number of ground truth image pairs that are labeled as same and different are 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 DCGAN to ensure CEAL DCGAN 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 a quite balance FPR and FNR (highest F1).

Specifically, we experimented with different numbers of fully connected layers in the twin network of HPD and different numbers of neurons in each layer, and adding drop-out with different probabilities for the last hidden layer of the Siamese network. Further, instead of computing the Euclidean distance between the output of the Siamese network for the two images in a pair, we also tested by concate-

nating these outputs and feeding them directly into the following fully connected layer(s). Figure 6.4 shows the best performing architecture for the HPD network. Table 6.2 shows the performance of the Siamese network and of the HPD networks that we trained and used to train CEAL DCGAN. In addition, we also use 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_model_2, has a higher precision on the hold out datasets compared to model 1 (see Table 6.2). We also, tested with this network to train and evaluate CEAL.

### 6.6.2 CEAL DCGAN Parameter Choice

In addition to using different HPD models to train CEAL DCGAN, we experimented with 2 different architectures using different number of neurons in the first layer of $G_{ceal}$ (i.e. 8,192 and 16,384). The size of this layer also directly impacts the number of following convolution transpose (a.k.a deconvolution) layers in $G_{ceal}$: more input neurons, larger layers.

We also performed a grid search in the parameters of the CEAL DCGAN 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 ceal generator (see Equation 6.1). For best performing parameters, we also tested with different weight initialization for the networks weights.

We trained the CEAL DCGAN using the process described in § 6.4.1, for 5 epochs, with batch size 64, and the Adam optimizer [KB15] to minimize Equation 6.1 for each step. We completed an epoch when all the images in the outdoor image dataset were shown to the discriminator. In order 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 $\alpha$ is increased, the $HPD_{loss}$ decreases faster (see Equation 6.1). However, the quality of the images is reduced for large values of $\alpha$. In addition, we observed that it is harder to train networks with larger values of $\lambda$s: the quality of images generated by CEAL DCGAN and their distinguishability decreases as we increase $\lambda$. Finally, we observed that when the size of the nodes in the first layer of $G_{ceal}$ is increased, the network generates smoother (blurred) with lower quality images.

We also experimented with the number of times that the generator network is trained using the three steps described in § 6.4.1, in each training epoch of $G_{ceal}$. Using the first and third steps, we train $G_{ceal}$ to prompt major components to result in perceptible change when their values are modified. However, the second step focuses on training minor components to make them cause imperceptible change in the output images when their values are changed from -1 to 1. 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.

Evaluating the generative models is a hard task. Theis et al. [TOB15] discuss that the generative models should be evaluated with respect to the application domain, since performance using a certain criterion does not necessarily extend to good performance with respect to another criterion. In order to compare the trained networks, we use the values for the $HPD_{loss}$ as well as the $G_{loss}$. The former is an indicator of how different the generated images are as perceived by human, while the latter is an indicator on how realistic and visually similar the generated images are compared to the images in the real image dataset used for training the discriminator.

We have manually evaluated the quality of the images generated by the networks we trained. The parameters for the best performing network using HPD_model_1 are $\alpha = 40$, $\lambda$ =256, and $M = m = 128$. The values of the $HPD_{loss}$ and $G_{loss}$ for this model were 0.6 and 12, respectively.

### 6.6.3 Key Mapper Parameter Choice

In early experiments, we observed that in order to consistently achieve human distinguishability, we need to flip the values of more than 1 major component (see § 6.3). To identify the minimum number of major components that need to be modified to achieve consistent human distinguishability, we manually inspected CEAL image pairs generated by flipping $d$ major components, where $d = 1, 2, 3, ..., 128$. We identified $d > 10$ to be a suitable value. Therefore, through this paper we consider 3 different values for $\gamma$ i.e., 92, 85 and 78. For this, we use BCH(127, 92, 5), BCH(127, 85, 6) and BCH(127, 78, 7) respectively in the key mapper module to transform the binary key fingerprint of length $\gamma$ into a binary string of length 127 that are at least in Hamming distance of 11, 13, and 15 respectively.

Note, as we show in our experiments, the choice of $\gamma$ and the BCH error tolerance (t) can be used to tune the security vs capacity of CEAL (see § 6.7.3). Based on our attack results, $\gamma = 78$ was selected as the parameter that achieves highest security (see $S$ 6.7.3 and 6.7.3). In addition, in § 6.7.2, we found that recruited human subjects labeled all the ceal samples that were generated using latent vectors that are different in $d = 15$ major components as different. Note, we did not used any randomness provided by minor components: the minor components are the same between target and attack strings. Therefore, we use a BCH(n=127, k=78, t=7) for KMap, an ECC with minimum Hamming distance of 15 bits that transforms a

message of length 78 into a code word of length 127. Thus, CEAL DCGAN accepts binary key fingerprints of length $\gamma = 78$ bits. Based on this setting, the maximum capacity of CEAL is $2^{78}$, i.e., CEAL can generate $2^{78}$ unique and distinguishable images. In § 6.3, we described that we can achieve human distinguishability with some error. In the following, we estimate this error to be $\sim 1.02\%$.

## 6.7 Empirical Evaluation

In this section we use human participants to evaluate the CEAL system with parameters identified in § 6.6, and compare it against Vash [vas14], the state-of-the-art visual fingerprint solution. In the following, we first describe the procedure we employed to run the user studies, then investigate Vash and report vulnerabilities that we identified. We evaluate the effects of major and minor components on human perception, and the resilience of CEAL against the adversary described in § 6.2.3. Finally, we compare the human distinguishability and verification speed of CEAL and Vash.

### 6.7.1 User Study Procedure

Throughout our evaluation, we followed an IRB-approved protocol to recruit Amazon Mechanical Turk workers to evaluate the performance of CEAL and Vash [vas14]. Specifically, we have recruited 519 adult, US-based workers, to compare a total of 6,579 image pairs: 6,309 CEAL and 270 Vash generated image pairs.

We asked each worker to compare either 35 or 50 pairs of images, and paid them \$0.4 or \$0.5, respectively. To verify worker attention, and discard data from inattentive ones, we included 5 attention check questions in each survey: 3 obviously different pairs of images, and 2 pairs of same (duplicated) images. For CEAL, we

generated an obviously different attention check image pair from a random seed latent vector, and flipped (1 vs. -1) a random set of its 100 major components (out of 128). For Vash, we generated obviously different attention check image pairs randomly. We manually verified that all these image pairs look indeed different.

We removed the answers from 12 participants who had incorrectly answered more than 2 (out of 5) attention check questions in the study.

Thus, we collected labels from 507 human workers: 180 female and 327 male, with an age range of 18 to 84 (M=35.41, SD=10.38). 90.94% of our participants had college education or higher. 253 (50%), 249 (49%), 5 (1%) participants used a desktop, laptop and mobile device to answer the survey respectively.

Overall, for each image pair, we collected annotations from at least 3 workers. In the Vash user studies of § 6.7.4, we collected at least 10 labels for each image pair. We then used majority voting [LWZF17] to aggregate the labels assigned by the workers to each image pair, and produced the human-assigned label.

### 6.7.2 Choice of Major Component Count

We first leverage the above human workers to evaluate the effects of changing the values for the major and minor components of the input latent vector to the CEAL DCGAN, on the visual characteristics of the generated images.

For this, we first evaluated CEAL's ability to meet one of our training objectives, i.e., that major components have perceptible impact on the generated images while minor components do no cause perceptible change in the images. than minor components. For this, we used the IPG of Definition 6.4.1 ($\lambda$=256), to generate two datasets $D_1$ and $D_2$ of image pairs, each containing 8,128 image pairs. We generated the image pairs in $D_1$ using random latent vectors that were different in only one

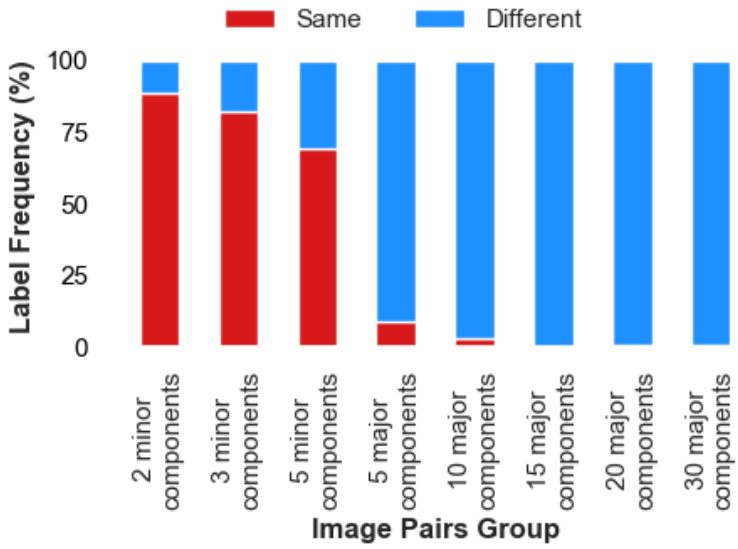



Figure 6.6: The distribution of "different" and "same" labels as annotated by MTurk workers. The number of image pairs that are identified as same decreases as we flip more number of minor components or major components.

major component, while the pairs in $D_2$ differ in only one minor component. We then used our HPD models (see Table 6.2) to compute the probability that image pairs are perceived as different by a human verifier. A t-test showed that indeed, the scores given by HPD to image pairs in $D_1$ are significantly higher than those given to image pairs in $D_2$ (P-value = 0.00).

In a second experiment we evaluated the ability of human workers (see § 6.7.1) to perceive changes in images when we changed the value for major and minor components in the latent vector. Specifically, 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. Further, we generated another set of 3,000 image pairs, by flipping the values of 2, 3 and 5 randomly chosen minor components in the latent vectors of 1,500, 1,000 and 500 images respectively.

Figure 6.6 shows the percentage of images that were annotated as "different" and "same" by workers, for each of the 8 different types of image pairs in our study. We

| Attack Dtataset | Size | # attacks found by HPD | # human verified attacks |
|---|---|---|---|
| $(78, 1)$-adversary | 78M | 13 | 0(0.00%) |
| $(78, d)$-adversary | 78M | 295 | 3 (1.02%) |

Table 6.3: Attack image datasets we generated to break CEAL. We show the dataset size, the portion of the (target, attack) samples that were identified by HPD_model_1, and the number of attack images validated by human workers.

observe that 91.0% of the image pairs that had only 5 major components flipped, were recognized as different by workers. This is significantly higher than the portion of samples labeled as different when we flipped 5 minor components (Z = 14.35, P-value = 0.00). In addition, the number of identified different images increased as we flipped more major components: when we flipped 15 major components, none of the image pairs were identified as being the same.

Thus, we found that even if we take out the randomness provided by minor components to the ceal images, the generated images are distinguishable if enough number of major components (here, $> 15$) are flipped. Thus, in the following, we set $d$ to 15.

### 6.7.3 CEAL Under Attack

#### CEAL Under $(78, 1)$-Attack

We now evaluate CEAL under brute force attacks perpetrated by the adversary defined in § 6.2.3. We first consider a $(\gamma, 1)$-adversary, who can find usable inputs that are within 1-Hamming distance of victim input, and uses them to generate attack ceal images. Specifically, for $\gamma$=78 (§ 6.7.3 includes a similar evaluation for $\gamma$=92), we generated 1 million target inputs randomly. Then, for each such input, we considered all $\gamma$ "attack" strings that are within 1-Hamming distance, and used


Figure 6.7: Sample (target, attack) image pairs we generated for $(\gamma = 78, d = 1)$-attacks, along with the human subjects' labeled for these image pairs.

the CEAL DCGAN to generate ceal images corresponding to the target and attack strings. Therefore, in total we generated 78 million ceal image pairs for $\gamma = 78$. Note, we select d=1 as it is highly likely to generate similar images for similar input to a GAN.

We first used the HPD classifier to decide if the generated image pairs would be perceived as being the same by a human verifier. Out of 1 million target ceal images, 13 of them were broken (only) once according to the HPD_model_1 (see Table 6.3 top). We then presented these 13 presumably broken ceal images to human verifiers (see § 6.7.1). None of these images were labeled as being the same by the recruited workers. Figure 6.7 represents several target and attack ceal images that we generated.

### CEAL Under $(78, d)$-Attack

We now consider a $(\gamma, d)$-adversary (§ 6.2.3), where $1 \leq d \leq \gamma$. 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, then for each target input, we randomly selected an "attack" string that is within Hamming distance $d$ from the target. We generated

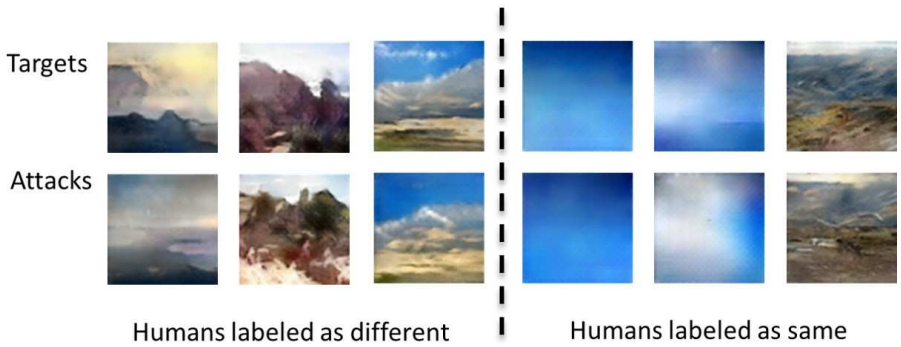



Figure 6.8: Sample (target, attack) image pairs we generated for $(\gamma = 78, d)$-attacks, along with the human subjects' labeled for these image pairs.

the ceal images corresponding to each target and attack strings pair. Thus, in total we generate $\gamma$ million ceal image pairs, organized into $\gamma$ datasets, each containing 1 million (target, attack) ceal image pairs. We present results over an evaluation for $\gamma$ of 78. We include results for $\gamma$ of 85 and 92, in § 6.7.3.

We then run HPD_model_1 over the (target, attack) image pairs that we generated. For $\gamma = 78$, HPD_model_1 predicted 295 of image pairs as indistinguishable. When we presented these image pairs to human workers (§ 6.7.1), only 3 of them were verified as being the same (see Table 6.3 bottom).

Based on the small false accept rate of CEAL on this attack and the $(\gamma, 1)$-attack of § 6.7.3, we conclude that for $\gamma$=78, CEAL is resilient to adversaries that are significantly more powerful that the ones considered in previous work [DSB$^+$, TBB$^+$17]. Under attacks of similar strength, Tan et al. report a false accept rate of 12% for Vash [vas14] and 10% for the OpenSSH Visual Host Key [LLvG09]). This is significantly larger than the CEAL false accept rate (i.e. $3/(13 + 295) < 1\%$). Figure 6.8 represents several (target, attack) ceal images in this experiment, along with their labeled that we collected using MTurk.

| $\gamma$ | # attack samples | # broken ceals HPD_model_1 | # human verified attacks |
|---|---|---|---|
| 92 | 92 M | 443 | 11 (2.48%) |
| 85 | 85 M | 379 | - |
| 78 | 78 M | 295 | 3 (1.02%) |

Table 6.4: Number of broken ceal images in $(\gamma, d)$-Attacks as identified by HPD_model_1.

**CEAL Under (92, 1)-Attack**

Similar to § 6.7.3, we consider and adversary with access to all attack keys whose binary fingerprint is in 1-Hamming distance of a target key. For $\gamma = 92$, out of 1 million target ceal images, 27 of them were broken at least once according to HPD_model_1 The maximum number of times (out of $\gamma$) that a target key fingerprint was broken under HPD_model_1 is once. We manually verified the identified (target, attack) image pairs. Although, several of the image pairs were indeed similar, we did not find any of the 27 image pairs to be undistinguished.

**CEAL Under $(\gamma, d)$-Attack**

We now report the performance of a $(\gamma, d)$-adversary when breaking CEAL with $\gamma$ of 92, 85 and 78. Similar to attack performed in § 6.7.3, we generate $\gamma$ million pairs of (target, attack) samples. Table 6.4 shows the total number of "broken" ceals for each value of $\gamma$ using HPD_model_1. We observe that only a small number of ceal images were broken according to HPD_model_1. Also, this value decreases when using a KMap that uses a BCH code with higher error tolerance (i.e., higher minimum Hamming distance between the code-words).

We labeled 443 and 295 pairs of images that were identified by HPD for $\gamma$ equal to 92 and 78 respectively using the user study procedure described in § 6.7.1.

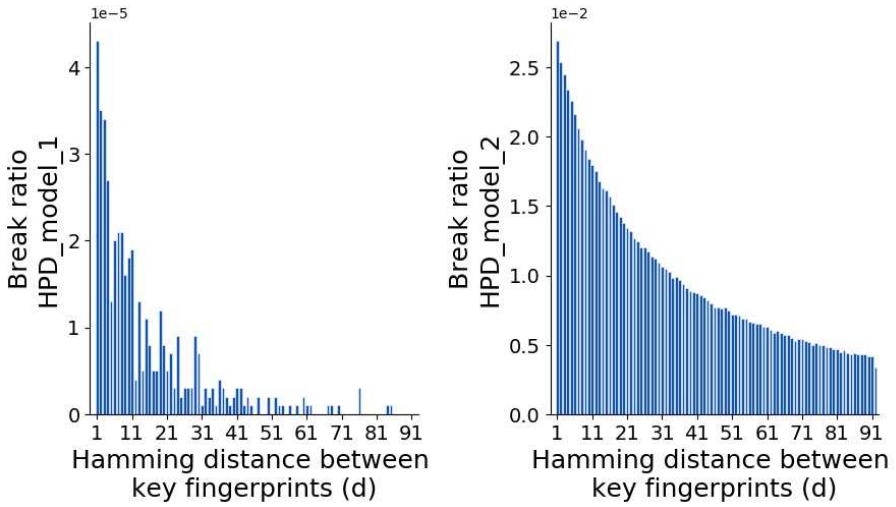



Figure 6.9: $(\gamma = 92, d)$-adversary: The break ratio of 1 million target ceal images for each value of d, where d is the Hamming distance between the attack and the target binary fingerprints and $0 < d < 93$ according to (left) HPD_model_1 and (right) HPD_model_2 (see § 6.7.3)

Our workers identified 11 and 3 of image pairs as same respectively in each group. In addition to the above Mturk study, we manually labeled 379 image pairs from experiments with $\gamma = 85$. We only identified 5 image pairs out of 379 image pairs to be visually very similar.

In addition to HPD_model_1, we used HPD_model_2 to identify potential successful attack samples in 92M pairs of images we generated to model a $(\gamma = 92, d)$-adversary. Figure 6.9 shows the portion of broken ceal images in each of the 92 datasets according to (left) HPD_model_1 and (right) HPD_model_2. As expected, the number of broken ceal images decreases as the Hamming distance between the target and attack binary key fingerprints increases. We observe the same effect for $\gamma = 85$ and $\gamma = 78$.

As HPD_model_2 has a higher FPR compared to HPD_model_1, it identified a larger number (906,678) pairs as potential attack samples. To validate the results of the HPD_model_2, we used the workers from § 6.7.1 to also annotate 1,557 ran-

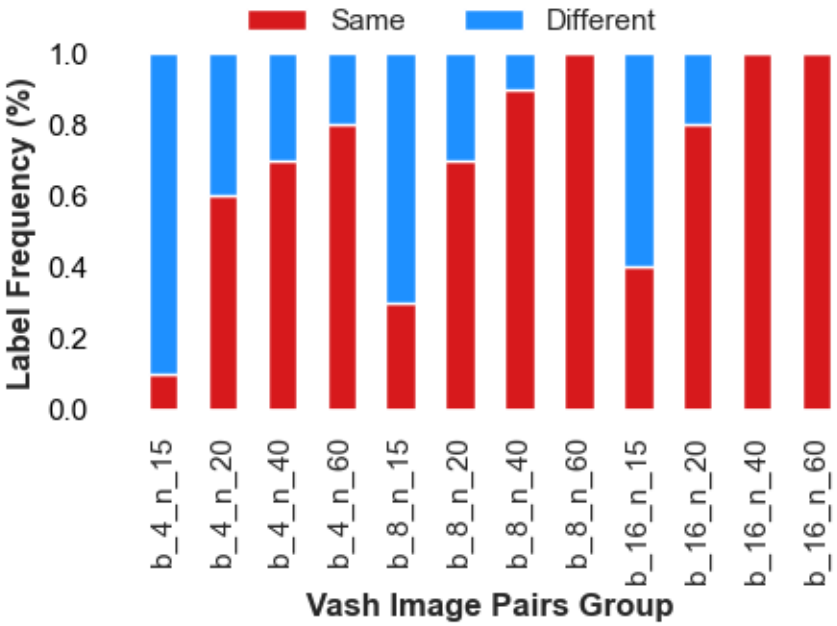



Figure 6.10: Distribution of "different" and "same" labels as annotated by human workers for Vash image pairs. The number of image pairs that are identified as same decreases as the number of buckets (b) and number of nodes (n) in the tree are decreased.

domly chosen, successful attack image pairs that were identified by HPD_model_2 for $d = 1$. Only 23 image pairs (out of 1,557) were identified as being the same by our participants. Thus, as we discussed previously, even with the imperfect HPD models we have, we are able to train CEAL DCGAN to generate ceals that are indistinguishable only with small probability (0.01 for $\gamma = 78$).

### 6.7.4 Human-Distinguishability of Vash

To evaluate the ability of Vash [vas14] to generate human-distinguishable images, we generated 120 Vash image pairs, all different, as follows. We first quantized the random values used to select each operation into the Vash tree (see § 2.3.2), into 32 buckets, and quantized the operation parameter values into $b$ buckets of the same lengths. We experimented with values of $b$ in $\{4, 8, 16\}$. We then generated random

trees until we had 30 trees of each size $N \in \{15, 20, 40, 60\}$. Then, we 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 $q$. The values for $q$ that we used for each group of trees are $\{0.25, 0.125, 0.0625\}$ respectively (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 § 6.7.1 to label these pairs using 40 human workers. Each image pair was labeled by 10 workers.

Figure 6.10 shows the portion of image pairs in each category that were labeled as either same or different images by our workers. 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 experiment in § 6.7.5), 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, and that Vash is unlikely to provide second pre-image resistance.

### 6.7.5 CEAL vs. Vash

We estimate a lower bound on capacity of Vash and CEAL using the method proposed by Orlitsky et al. [OSV07]. We also report the time to compare ceal images and compare it to Vash.

**Data.** We generate 10K images randomly (from random keys) using Vash and

| Key fingerprint representation | Attack dataset size | # attacks found by HPD | Verified attacks |
|---|---|---|---|
| CEAL | ∼50M | 1 | 0 (0%) |
| VASH | ∼50M | 150 | 24 (16%) |

Table 6.5: 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.

CEAL (total 20K images). Then for each dataset of images, generated using Vash and CEAL, we use HPD_model_1 to predict if pairwise images are human distinguishable (total $\frac{10K \times 9999}{2} = 49,995,000$ comparisons). Here, we also evaluate performance of HPD_model_1 on 120 Vash images pairs and labels that we collected from user study reported in § 6.7.4. On 120 image pairs, HPD_model_1 has a FAR of 0.21, FNR of 0.14 and F1 of 0.76. This results confirms that our trained HPD model, also perform well in identifying distinguishable changes in images (e.g. Vash) that are dramatically different that the images used in training (nature images).

To estimate the number of distinguishable images for each VKFG, we compute $\hat{k}(N_r, r) = \frac{{N_r}^2}{2r}$ where $N_r$ is the number of samples until observing $r$ repetition, i.e., human indistinguishable images (see [OSV07]). Note, we use this method as a lower bound estimate for the capacity of VKFG, 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. Therefore, it is not possible to check if capacity of CEAL is $2^{78}$ using a only 10K samples.

HPD identified 150 ($3^{-4}\%$) and 1 ($2^{-6}\%$) Vash and ceal image pairs as indistinguishable in the first 10K samples of each. We then labeled these image pairs using MTurk (see Table 6.5). Particularly, we labeled the identified Vash images using 15 MTurk workers using similar procedure as described for previous Vash user study (see § 6.7.4). We also labeled the 1 identified ceal image pair using MTurk using the

procedure described in § 6.7.1. Out of 150 (target, attack) image pairs, 24 image pairs were identified as same by our workers (16%). 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 aligned with the results reported by Hsiao et al. [HLS+09]. However, after user study the identified ceal image pair was labeled as different by our workers. Therefore, we did not identify any indistinguishable images in the first 10K samples of ceal.

**Vash vs CEAL Images Comparison Time**. We compare the response time of participants to (target, attack) image pairs we generated for 150 Vash image pairs as well as 309 ceal image pairs (295 images from § 6.7.3, 13 images of § 6.7.3 and 1 image pairs identified in the above experiment). The average comparson time over Vash attack images is 3.03s (M=1.4s, SD=5.42s), while for ceal is 2.04s (M=1.5s, SD=3.44s). A t-test revealed that the time to compare ceal attack images is significantly shorter than the time they took to compare Vash attack images (P-Value = 0.024).

In addition, to compare timing for Vash and ceal images with almost the same level of distinguishability, we compare the time to compare the ceal and Vash images that were distinguishable by human $\sim 70\%$ of the times. For this, we consider the time to compare Vash image pairs that were generated using $n = 15$ for different values of $q$ (see § 6.7.4) to the time it took to compare ceal images that were generated using latent vectors with 5 different minor components (see § 6.7.2). The average comparison time over ceal image pairs is 4.82s (M=3.71s, SD=4.43s). However, this value for Vash images is 6.15s (M=3.71s, SD=5.89s). Again, a t-test revealed that the average comparison time for ceal images is significantly lower than that of Vash (P-Value = 0.020).

## 6.8 Discussion and Limitations

**Increasing entropy**. One way to increase the entropy of the CEAL key fingerprint generator, is to design and train multiple generators (see § 6.6), 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, we note that 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.** As we defined in § 6.3, we use a HPD, a classifier that we use to predict human distinguishability ($HPD_{ration}$) of image pairs. However, due to data and training limitations, our classifier has some error. However, we show that even using a weak classifier, we are able to train a generator network to generate images that are human distinguishable.

For instance, we manually verified ceal image pairs that were identified as same by our workers in § 6.7.3. The input strings for the verified attack images were within a Hamming distance of 2, 4 and 6 from their target. We manually checked these image pairs and observed that 2 of these images represent a blue sky with shadows from different angles (see Figure 6.8). We conjecture that, by training HPD to further identify texture-less images as being the same, and retraining CEAL DCGAN, we can avoid generating such images.

**Usability and effectiveness**. The results of our studies do not generalize to the entire population, as we performed them on only a subset of MTurk workers, which are also not representative of the entire population. For instance, we conjecture that workers who work on visualization tasks are less likely to suffer from vision loss problems. Further, MTurk workers 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.

We also note that key fingerprint comparisons should be robust to key fingerprints 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. In addition, we leave for future work the evaluation of the memorability of CEAL-generated images, which may help improve long term recognition and verification of key fingerprints as it bring meaning to key fingerprint images, e.g. compared to several textual representation, bar-codes, etc.

In addition, commonly used key text-based fingerprints have been shown to be ignored by users [Gut11]. Thus, an extensive study is required to understand if representing key fingerprints as realistic and familiar-looking images, will help draw user attention, and increase verification rates.

## 6.9 Conclusions

In this chapter, we built the first visual fingerprint solution with built-in input distribution properties, and have shown that it is substantially superior to state-of-the-art solutions, in terms of entropy, human accuracy and speed of evaluation.

We leave for future work an investigation into the use of CEAL with key stretching methods to improve entropy, investigating alternative HPD and GAN (e.g. [KALL18]) architectures, and including other types of classifiers during training, e.g. to build ceals appropriate for color blind people.

# BIBLIOGRAPHY


[AAW15] Mahdi Nasrullah Al-Ameen and Matthew Wright. Multiple-password interference in the geopass user authentication scheme. In *Proc. Workshop Usable Secur.(USEC)*, pages 1–6, 2015.

[ABC+16] 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 *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, GA, 2016. USENIX Association.

[ACJP14] Sunpreet S Arora, Kai Cao, Anil K Jain, and Nicholas G Paulter. 3d fingerprint phantoms. In *2014 22nd International Conference on Pattern Recognition*, pages 684–689, Aug 2014.

[akw] akwizgran. Encode random bitstrings as pseudo-random poems. https://github.com/akwizgran/basic-english.

[App17a] About touch id security on iphone and ipad, 2017. `https://support.apple.com/en-us/HT204587`.

[App17b] Use touch id on iphone and ipad, 2017. https://support.apple.com/en-us/HT201371.

[ATC17a] Mozhgan Azimpourkivi, Umut Topkara, and Bogdan Carbunar. Camera based two factor authentication through mobile and wearable devices. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2017.

[ATC17b] Mozhgan Azimpourkivi, Umut Topkara, and Bogdan Carbunar. A secure mobile authentication alternative to biometrics. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 28–41. ACM, 2017.

[Bal87] Dana H. Ballard. Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, AAAI'87, pages 279–284. AAAI Press, 1987.

[BCF+13] Arman Boehm, Dongqu Chen, Mario Frank, Ling Huang, Cynthia Kuo, Tihomir Lolic, Ivan Martinovic, and Dawn Song. Safe: Secure authentication with face and eyes. In *2013 International Conference on Privacy and Security in Mobile Systems (PRISMS)*, pages 1–8, June 2013.

[BCTPL16] Jorge Blasco, Thomas M. Chen, Juan Tapiador, and Pedro Peris-Lopez. A survey of wearable biometric recognition systems. *ACM Comput. Surv.*, 49(3):43:1–43:35, September 2016.

[BCVO12] Robert Biddle, Sonia Chiasson, and P.C. Van Oorschot. Graphical passwords: Learning from the first twelve years. *ACM Comput. Surv.*, 44(4):19:1–19:41, September 2012.

[BML06] Lucas Ballard, Fabian Monrose, and Daniel Lopresti. Biometric authentication revisited: Understanding the impact of wolves in sheep's clothing. In *Proceedings of the 15th Conference on USENIX Security Symposium*, Berkeley, CA, USA, 2006.

[BOK11] Andrea Bianchi, Ian Oakley, and Dong Soo Kwon. *Spinlock: A Single-Cue Haptic and Audio PIN Input Technique for Authentication*, pages 81–90. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[Bon12] Joseph Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *2012 IEEE Symposium on Security and Privacy*, pages 538–552, May 2012.

[BRC60] 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.

[BS00] Sacha Brostoff and M Angela Sasse. Are passfaces more usable than passwords? a field trial investigation. In *People and Computers XIV: Usability or Else!* 2000.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. *SURF: Speeded Up Robust Features*, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[BUI+15] Chandrasekhar Bhagavatula, Blase Ur, Kevin Iacovino, Su Mon Kywe, Lorrie Faith Cranor, and Marios Savvides. Biometric authentication

on iphone and android: Usability, perceptions, and influences on adoption. *Proceeding of Usable Security (USEC)*, pages 1–2, 2015.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.

[CaS17a] CaSPR Lab. ai.lock, 2017. https://github.com/casprlab/ai.lock.

[CaS17b] CaSPR Lab. Pixie Application, 2017. https://play.google.com/store/apps/details?id=org.image.password.trinket.v1.

[CaS17c] CaSPR Lab. Pixie Data, 2017. https://drive.google.com/drive/folders/0B-qU-nMycga7S1FKbURsU1BlVmc?usp=sharing.

[CaS17d] CaSPR Lab. Pixie Demo, 2017. https://casprlab.github.io.

[CaS17e] CaSPR Lab. Pixie Source Code, 2017. https://github.com/casprlab/pixie.

[CDF+07] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. Openpgp message format. `https://tools.ietf.org/html/rfc4880`, 2007.

[CDH+16] 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, USA, 2016. Curran Associates Inc.

[CDK+12] Alexei Czeskis, Michael Dietz, Tadayoshi Kohno, Dan Wallach, and Dirk Balfanz. Strengthening user authentication through opportunistic cryptographic identity assertions. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 404–414, New York, NY, USA, 2012. ACM.

[CFS+09] Sonia Chiasson, Alain Forget, Elizabeth Stobert, P. C. van Oorschot, and Robert Biddle. Multiple password interference in text passwords and click-based graphical passwords. In *Proceedings of the 16th ACM*

*Conference on Computer and Communications Security*, CCS '09, pages 500–511, New York, NY, USA, 2009. ACM.

[Cha02] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 380–388, New York, NY, USA, 2002. ACM.

[CHL05] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, June 2005.

[CHM15] Pan Chan, Tzipora Halevi, and Nasir Memon. *Glass OTP: Secure and Convenient User Authentication on Google Glass*, pages 298–308. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[CHY+14] Charles F Cadieu, Ha Hong, Daniel LK Yamins, Nicolas Pinto, Diego Ardila, Ethan A Solomon, Najib J Majaj, and James J DiCarlo. Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS computational biology*, 10(12):e1003963, 2014.

[CL15] Francisco Corella and Karen Pomian Lewison. Protecting credentials against physical capture of a computing device, April 23 2015. https://www.google.com/patents/US20150113283.

[CMAB15] Ivan Cherapau, Ildar Muslukhov, Nalin Asanka, and Konstantin Beznosov. On the impact of touch id on iphone passcodes. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 257–276. USENIX Association, 2015.

[Con12] Lucian Constantin. computerworld trustwave admits issuing man-in-the-middle digital certificate; mozilla debates punishment, February 2012. https://www.computerworld.com/article/2501291/internet/trustwave-admits-issuing-man-in-the-middle-digital-certificate–mozilla-debates-punishment.html.

[Con16] Kate Conger. TechCrunch: Facebook Messenger adds end-to-end encryption in a bid to become your primary

messaging app. https://techcrunch.com/2016/07/08/messenger-adds-end-to-end-encryption, 2016.

[DBML18] Chris Donahue, Akshay Balsubramani, Julian McAuley, and Zachary C. Lipton. Semantically decomposing the latent spaces of generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[Den12] Li Deng. Three classes of deep learning architectures and their applications: a tutorial survey. *APSIPA transactions on signal and information processing*, 2012.

[DH12] Ed. D. Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, IETF, October 2012.

[DJV+14] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Icml*, volume 32, pages 647–655, 2014.

[DLHB+12] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and i know it's you!: Implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 987–996, New York, NY, USA, 2012. ACM.

[DLHvZ+14] Alexander De Luca, Marian Harbach, Emanuel von Zezschwitz, Max-Emanuel Maurer, Bernhard Ewald Slawik, Heinrich Hussmann, and Matthew Smith. Now you see me, now you don't: Protecting smartphone authentication from shoulder surfers. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 2937–2946, New York, NY, USA, 2014. ACM.

[DLHvZH15] Alexander De Luca, Alina Hang, Emanuel von Zezschwitz, and Heinrich Hussmann. I feel like i'm taking selfies all day!: Towards understanding biometric authentication on smartphones. In *Proceedings of*

*the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1411–1414, New York, NY, USA, 2015. ACM.

[DMR04] Darren Davis, Fabian Monrose, and Michael K. Reiter. On user choice in graphical password schemes. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 11–11, Berkeley, CA, USA, 2004. USENIX Association.

[DNO08] Paul Dunphy, James Nicholson, and Patrick Olivier. Securing passfaces for description. In *Proceedings of the 4th Symposium on Usable Privacy and Security*, 2008.

[DOT01] Pieter Desmet, Kees Overbeeke, and Stefan Tax. Designing products with added emotional value: Development and appllcation of an approach for research through design. *The Design Journal*, 4(1):32–47, 2001.

[DP00] Rachna Dhamija and Adrian Perrig. Déjà vu: A user study using images for authentication. In *Proceedings of the 9th Conference on USENIX Security Symposium - Volume 9*, SSYM'00, pages 4–4, Berkeley, CA, USA, 2000. USENIX Association.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. *Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data*, pages 523–540. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

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

[DY07] Paul Dunphy and Jeff Yan. Do background images improve "draw a secret" graphical passwords? In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 36–47, New York, NY, USA, 2007. ACM.

[EGSD18] Gamaleldin F Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial reprogramming of neural networks. *arXiv preprint arXiv:1806.11146*, 2018.

[FA12] Nathaniel Wesley Filardo and Giuseppe Ateniese. *High-Entropy Visual Identification for Touch Screen Devices*, pages 182–198. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[FFFP04] Li Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 178–178, June 2004.

[Fis12] Dennis Fisher. threadpost final report on diginotar hack shows total compromise of ca servers, October 2012.

[F.R01] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559–572, 1901.

[FVY14] Conner Fromknecht, Dragos Velicanu, and Sophia Yakoubov. Certcoin: A namecoin based decentralized authentication system 6.857 class project. 2014.

[GBS05a] Jan-Mark Geusebroek, Gertjan J. Burghouts, and Arnold W.M. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[GBS05b] Jan-Mark Geusebroek, Gertjan J. Burghouts, and Arnold W.M. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[GEB16] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[Gem] Gemalto. Grid authentication. https://safenet.gemalto.com/multi-factor-authentication/authenticators/grid-authentication.

[GGC+08] H. Gao, X. Guo, X. Chen, L. Wang, and X. Liu. Yagp: Yet another graphical password strategy. In *2008 Annual Computer Security Applications Conference (ACSAC)*, pages 121–129, Dec 2008.

[GMS15] 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.

[Gol] Jeff Goldman. 74 percent of organizations using two-factor authentication face user complaints, January. http://www.esecurityplanet.com/network-security/74-percent-of-organizations-using-two-factor-authentication-face-user-complaints.html.

[Goo17] Google 2-step verification user guide, 2017. https://www.google.com/landing/2step/.

[GPAM+14] 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. Curran Associates, Inc., 2014.

[GRGB+12] Javier Galbally, Arun Ross, Marta Gomez-Barrero, Julian Fierrez, and Javier Ortega-Garcia. From the iriscode to the iris: A new vulnerability of iris recognition systems. *Black Hat Briefings USA*, 2012.

[GSS+06] Michael T Goodrich, Michael Sirivianos, John Solis, Gene Tsudik, and Ersin Uzun. Loud and clear: Human-verifiable authentication based on audio. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 10–10. IEEE, 2006.

[GT06] J. Galbraith and R. Thayer. The secure shell (ssh) public key file format. `https://www.ietf.org/rfc/rfc4716.txt`, 2006.

[Gut11] Peter Gutmann. Do users verify ssh keys. *Login*, 36:35–36, 2011.

[HDLE16] Marian Harbach, Alexander De Luca, and Serge Egelman. The anatomy of smartphone unlocking: A field study of android lock screens. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 4806–4817, New York, NY, USA, 2016. ACM.

[HLP12] C. Y. Hsu, C. S. Lu, and S. C. Pei. Image feature extraction in encrypted domain with privacy-preserving sift. *IEEE Transactions on Image Processing*, 21(11):4593–4607, Nov 2012.

[HLS+09] 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 *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, pages 105–114. IEEE, 2009.

[Hoc59a] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2(147-156):8–5, 1959.

[Hoc59b] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2(2):147–56, 1959.

[HPOH12] Eiji Hayashi, Bryan Pendleton, Fatih Ozenc, and Jason Hong. Webticket: Account management using printable tokens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 997–1006, New York, NY, USA, 2012. ACM.

[Hui00] Huima. The bubble babble binary data encoding. `http://web.mit.edu/kenta/www/one/bubblebabble/spec/jrtrjwzi/draft-huima-01.txt`, 2000.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[Jef17] Jeff Kent. python-bchlib, 2017. https://github.com/jkent/python-bchlib/.

[JMM+99] Ian Jermyn, Alain Mayer, Fabian Monrose, Michael K. Reiter, and Aviel D. Rubin. The design and analysis of graphical passwords. In *Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8*, SSYM'99, pages 1–1, Berkeley, CA, USA, 1999. USENIX Association.

[JNN08] Anil K. Jain, Karthik Nandakumar, and Abhishek Nagar. Biometric template security. *EURASIP J. Adv. Signal Process*, 2008:113:1–113:17, January 2008.

[JS02] A. Juels and M. Sudan. A fuzzy vault scheme. In *Proceedings IEEE International Symposium on Information Theory,*, pages 408–, 2002.

[JW99] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, CCS '99, pages 28–36, New York, NY, USA, 1999. ACM.

[JW15] A. H. Johnston and G. M. Weiss. Smartwatch-based biometric gait recognition. In *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–6, Sept 2015.

[KAH+16] Mohamed Khamis, Florian Alt, Mariam Hassib, Emanuel von Zezschwitz, Regina Hasholzner, and Andreas Bulling. Gazetouchpass: Multimodal authentication using gaze and touch on mobile devices. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 2156–2164, New York, NY, USA, 2016. ACM.

[KALL18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations (ICLR 2018)*, May 2018.

[KB15] D. P. Kingma and J. L. Ba. Adam: a method for stochastic optimization. In *Conference on Learning Representations (ICLR 2015)*, May 2015.

[KE10] H. Krawczyk and P. Eronen. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869, IETF, May 2010.

[Kee15] Thomas P Keenan. Hidden risks of biometric identifiers and how to avoid them. *BlackHat USA*, 2015.

[KFB08a] Klaus Kollreider, Hartwig Fronthaler, and Josef Bigun. Verifying liveness by multiple experts in face biometrics. In *IEEE Computer Vision and Pattern Recognition Workshops*, 2008.

[KFB08b] Klaus Kollreider, Hartwig Fronthaler, and Josef Bigun. Verifying liveness by multiple experts in face biometrics. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–6, June 2008.

[KHX15] R. Khan, R. Hasan, and J. Xu. Sepia: Secure-pin-authentication-as-a-service for atm using mobile and wearable devices. In *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pages 41–50, March 2015.

[KMSv15] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Čapkun. Sound-proof: Usable two-factor authentication based on ambient sound. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, pages 483–498, Berkeley, CA, USA, 2015. USENIX Association.

[Koh95] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. pages 1137–1143, 1995.

[KvdVB16] Radhesh Krishnan Konoth, Victor van der Veen, and Herbert Bos. How anywhere computing just killed your phone-based two-factor authentication. In *Proceedings of the 20th International Conference on Financial Cryptography and Data Security*, Barbados, February 2016.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[KW17] Keith Kirkpatrick and Clint Wheelock. Global Unit Shipments and Revenue by Biometric Modality, Technology, Use Case, Industry Segment, and World Region: 2016-2025. February 2017. `https://www.tractica.com/research/biometrics-market-forecasts/`.

[KWB17] Matthias Kümmerer, Tom Wallis, and Matthias Bethge. Deepgaze ii: Predicting fixations from deep features over time and tasks. In *17th Annual Meeting of the Vision Sciences Society (VSS 2017)*, pages 1147–1147, 2017.

[LB+95] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[LCS11] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011.

[LDGW15] Dachuan Liu, Bo Dong, Xing Gao, and Haining Wang. *Exploiting Eye Tracking for Smartphone Authentication*, pages 457–477. Springer International Publishing, Cham, 2015.

[LGS+16] Moritz Lipp, Daniel Gruss, Raphael Spreitzer, Clémentine Maurice, and Stefan Mangard. Armageddon: Cache attacks on mobile devices. In *25th USENIX Security Symposium*, pages 549–564, 2016.

[LL16] Wei-Han Lee and Ruby Lee. Implicit sensor-based authentication of smartphone users with smartwatch. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*, HASP 2016, pages 9:1–9:8, New York, NY, USA, 2016. ACM.

[LLvG09] Dirk Loss, Tobias Limmer, and Alexander von Gernler. The drunken bishop: An analysis of the openssh fingerprint visualization algorithm, 2009.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LS15] N. Lalithamani and M. Sabrigiriraj. Palm and hand vein-based fuzzy vault generation scheme for multibiometric cryptosystem. *The Imaging Science Journal*, 63(2):111–118, 2015.

[LWZF17] G. Li, J. Wang, Y. Zheng, and M. Franklin. Crowdsourced data management: A survey. In *IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 39–40, April 2017.

[MB14] Päivi Majaranta and Andreas Bulling. *Eye Tracking and Eye-Based Human–Computer Interaction*, pages 39–65. Springer London, London, 2014.

[MCF+10] Emanuele Maiorana, Patrizio Campisi, Julian Fierrez, Javier Ortega-Garcia, and Alessandro Neri. Cancelable templates for sequence-based biometrics with application to on-line signature recognition. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 40(3):525–538, May 2010.

[MCP+15] D. Menotti, G. Chiachia, A. Pinto, W. R. Schwartz, H. Pedrini, A. X. Falcão, and A. Rocha. Deep representations for iris, face, and fingerprint spoofing detection. *IEEE Transactions on Information Forensics and Security*, 10(4):864–879, April 2015.

[MK16] Sanjeev Miglani and Manoj Kumar. India's billion-member biometric database raises privacy fears, March 2016. http://www.reuters.com/article/us-india-biometrics-idUSKCN0WI14E.

[MKS+16] William Melicher, Darya Kurilova, Sean M. Segreti, Pranshu Kalvani, Richard Shay, Blase Ur, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Michelle L. Mazurek. Usability and security of text passwords on mobile devices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 527–539, New York, NY, USA, 2016. ACM.

[ML09] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, pages 331–340, 2009.

[mon] WPMonsterID. `http://scott.sherrillmix.com/blog/blogger/wp_monsterid/`.

[MPR05] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. In *2005 IEEE Symposium on Security and Privacy (S P'05)*, pages 110–124, May 2005.

[MR16] Bogdan Carbunar Mahmudur Rahman, Umut Topkara. Movee: Video Liveness Verification for Mobile Devices with Built-in Motion Sensors. *IEEE Transactions on Mobile Computing (TMC)*, 15(5), 2016.

[MWFZ15] Weizhi Meng, Duncan S Wong, Steven Furnell, and Jianying Zhou. Surveying the development of biometric user authentication on mobile phones. *IEEE Communications Surveys Tutorials*, 17(3):1268–1293, thirdquarter 2015.

[NNJ07] Karthik Nandakumar, Abhishek Nagar, and Anil K. Jain. *Hardening Fingerprint Fuzzy Vault Using Password*, pages 927–937. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[NRW76] Douglas L Nelson, Valerie S Reed, and John R Walling. Pictorial superiority effect. *Journal of Experimental Psychology: Human Learning and Memory*, 2(5):523, 1976.

[O'G03] Lawrence O'Gorman. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12):2021–2040, Dec 2003.

[OJN08] Thian Song Ong, Andrew Teoh Beng Jin, and David Chek Ling Ngo. Application-specific key release scheme from biometrics. *IJ Network Security*, 6(2):127–133, 2008.

[OKS+13] Maina M Olembo, Timo Kilian, Simon Stockhardt, Andreas Hülsing, and Melanie Volkamer. Developing and testing a visual hash scheme. In *EISMC*, pages 91–100, 2013.

[OSV07] A. Orlitsky, N. P. Santhanam, and K. Viswanathan. Population estimation with performance guarantees. In *2007 IEEE International Symposium on Information Theory*, pages 2026–2030, June 2007.

[Out18] Jun-Yan Zhu outdoor 64 image dataset, 2018. https://github.com/junyanz/iGAN.

[Pad17] John Padgette. Guide to bluetooth security. *NIST Special Publication*, 800:121, 2017.

[Par07] Don Park. Visual security: 9-block ip identification, January 2007. https://web.archive.org/web/20080703155519/http://www.docuverse.com/blog/don security-9-block-ip-identification.

[Pas17] Passfaces: Two factor authentication for the enterprise, 2017. http://www.passfaces.com/index.htm.

[PCS17] S. Pouyanfar, S. C. Chen, and M. L. Shyu. An efficient deep residual-inception network for multimedia classification. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 373–378, July 2017.

[Pet15] Andrea Peterson. Opm says 5.6 million fingerprints stolen in cyberattack, five times as many as previously thought,

September 2015. https://www.washingtonpost.com/news/the-switch/wp/2015/09/23/opm-now-says-more-than-five-million-fingerprints-compromised-in-breaches/.

[PHJ16] K. Patel, H. Han, and A. K. Jain. Secure face unlock: Spoof detection on smartphones. *IEEE Transactions on Information Forensics and Security*, 11(10):2268–2283, Oct 2016.

[PLK+12] Iasonas Polakis, Marco Lancini, Georgios Kontaxis, Federico Maggi, Sotiris Ioannidis, Angelos D. Keromytis, and Stefano Zanero. All your face are belong to us: Breaking facebook's social authentication. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 399–408, New York, NY, USA, 2012. ACM.

[PPJ03] Salil Prabhakar, Sharath Pankanti, and Anil K. Jain. Biometric recognition: security and privacy concerns. *IEEE Security Privacy*, 1(2):33–42, Mar 2003.

[PS99] Adrian Perrig and Dawn Song. Hash visualization: A new technique to improve real-world security. In *International Workshop on Cryptographic Techniques and E-Commerce*, pages 131–138, 1999.

[PSJA15] Giuseppe Petracca, Yuqiong Sun, Trent Jaeger, and Ahmad Atamli. Audroid: Preventing attacks on audio channels in mobile devices. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 181–190, 2015.

[PTT00] 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.

[QYR+14] Zhan Qin, Jingbo Yan, Kui Ren, Chang Wen Chen, and Cong Wang. Towards efficient privacy-preserving image feature extraction in cloud computing. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 497–506, New York, NY, USA, 2014. ACM.

[Ram10] Zulfikar Ramzan. Phishing attacks and countermeasures. In *Handbook of information and communication security*, pages 433–448. Springer, 2010.

[RAM+18] Scott Ruoti, Jeff Andersen, Tyler Monson, Daniel Zappala, and Kent Seamons. A comparative usability study of key management in secure email. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 375–394, Baltimore, MD, 2018. USENIX Association.

[RATC17] M. Rahman, M. Azimpourkivi, U. Topkara, and B. Carbunar. Video liveness for citizen journalism: Attacks and defenses. *IEEE Transactions on Mobile Computing*, PP(99):1–1, 2017.

[RCCB07] Nalini K. Ratha, Sharat Chikkerur, Jonathan H. Connell, and Ruud M. Bolle. Generating cancelable fingerprint templates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(4):561–572, April 2007.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[RSA17] Rsa securid suite, 2017. https://www.rsa.com/en-us/products-services/identity-access-management/securid.

[Rus80] James Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6), 1980.

[Sam17] Samsung gear sm-v700 smartwatch, 2017. http://www.samsung.com/uk/consumer/mobile-devices/wearables/gear/SM-V7000ZKABTU/.

[Sch11] Zachary Schuessler. Delta E 101. `http://zschuessler.github.io/DeltaE/learn`, 2011.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[Sch17] Schlage keypad locks user guide, 2017. http://www.schlage.com/content/dam/sch-us/documents/pdf/installation-manuals/23780042.pdf.

[Sec15] Replacing rsa securid: Why are customers switching to duo security?, 2015. https://duo.com/blog/replacing-rsa-securid-why-are-customers-switching-to-duo-security.

[Sec17] Ford securicode keyless entry keypad, 2017. http://owner.ford.com/how-tos/vehicle-features/locks-and-security/securicode-keyless-entry-keypad.html.

[Sed14] Ami Sedghi. Proportion of people working from home reaches record high, 2014. https://www.theguardian.com/news/datablog/2014/jun/04/proportion-of-employed-working-from-home-reaches-record-high.

[Sha01] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[SJSN14] Maliheh Shirvanian, Stanislaw Jarecki, Nitesh Saxena, and Naveen Nathan. Two-factor authentication resilient to server compromise using mix-bandwidth devices. In *Proceedings of NDSS*, 2014.

[SKD+14] Richard Shay, Saranga Komanduri, Adam L. Durity, Phillip (Seyoung) Huh, Michelle L. Mazurek, Sean M. Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Can long passwords be secure and usable? In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 2927–2936, New York, NY, USA, 2014. ACM.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[SPC15] P. Samangouei, V. M. Patel, and R. Chellappa. Attribute-based continuous user authentication on mobile devices. In *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–8, Sept 2015.

[SRRM16a] Ivo Sluganovic, Marc Roeschlin, Kasper B. Rasmussen, and Ivan Martinovic. Using reflexive eye movements for fast challenge-response authentication. In *Proceedings of the 2016 ACM SIGSAC Conference on*

*Computer and Communications Security*, CCS '16, pages 1056–1067, New York, NY, USA, 2016. ACM.

[SRRM16b] Ivo Sluganovic, Marc Roeschlin, Kasper B. Rasmussen, and Ivan Martinovic. Using reflexive eye movements for fast challenge-response authentication. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1056–1067, New York, NY, USA, 2016. ACM.

[SSB+14] Stefan Schneegass, Frank Steimle, Andreas Bulling, Florian Alt, and Albrecht Schmidt. Smudgesafe: Geometric image transformations for smudge-resistant user authentication. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pages 775–786, New York, NY, USA, 2014. ACM.

[Sub18] William Suberg. Report: 2.3 Million Bitcoin Addresses Targeted by Malware That 'Hijacks' Windows Clipboard. `https://www.viber.com/security-overview`, 2018.

[Suo06] Xiaoyuan Suo. *A design and analysis of graphical password*. PhD thesis, 2006.

[SVI+16a] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, June 2016.

[SVI+16b] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[SWD05] 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):21–30, 2005.

[SWKW13] Florian Schaub, Marcel Walch, Bastian Könings, and Michael Weber. Exploring the design space of graphical passwords on smartphones. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, SOUPS '13, pages 11:1–11:14, New York, NY, USA, 2013. ACM.

[SZL15] Ling Shao, Fan Zhu, and Xuelong Li. Transfer learning for visual categorization: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 26(5):1019–1034, May 2015.

[TA08] Hai Tao and Carlisle Adams. Pass-go: A proposal to improve the usability of graphical passwords. *IJ Network Security*, 7(2):273–292, 2008.

[TAP17] Taps - make touchscreen gloves using a sticker w/ touch id, 2017. https://www.kickstarter.com/projects/nanotips/taps-touchscreen-sticker-w-touch-id-ships-before-x?token=5b586aa6.

[TBB+17] 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 2017 CHI Conference on Human Factors in Computing Systems*, pages 3787–3798, 2017.

[ten17] tensorflow. Inception in tensorflow, 2017.

[TGN06] Andrew B. J. Teoh, Alwyn Goh, and David C. L. Ngo. Random multispace quantization as an analytic mechanism for biohashing of biometric and random identity inputs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1892–1901, December 2006.

[TK03] Tetsuji Takada and Hideki Koike. Awase-e: Image-based authentication for mobile phones using user's favorite images. In *Human-computer interaction with mobile devices and services*. 2003.

[TMSA13] Julie Thorpe, Brent MacRae, and Amirali Salehi-Abari. Usability and security evaluation of geopass: a geographic location-password scheme. In *Proceedings of the 9th Symposium on Usable Privacy and Security*, 2013.

[TOB15] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. In *Conference on Learning Representations (ICLR 2015)*, May 2015.

[TRCK13] Robert Templeman, Zahid Rahman, David J. Crandall, and Apu Kapadia. Placeraider: Virtual theft in physical spaces with smartphones. In *Annual Network and Distributed System Security Symposium*, 2013.

[Tru17] ARM trustzone, 2017. https://www.arm.com/products/security-on-arm/trustzone/.

[TSF+16] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Commun. ACM*, 59(2):64–73, January 2016.

[TSK+12] Shari Trewin, Cal Swart, Larry Koved, Jacquelyn Martino, Kapil Singh, and Shay Ben-David. Biometric authentication on a mobile device: A study of user effort, error and task disruption. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 159–168, New York, NY, USA, 2012. ACM.

[TYRW14] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, June 2014.

[UDB+15] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith. Sok: Secure messaging. In *2015 IEEE Symposium on Security and Privacy*, pages 232–249, May 2015.

[vas14] The Vash: Visually pleasing and distinct abstract art, generated uniquely for any input data. `https://github.com/thevash`, 2014.

[VAS17] VASCO digipass go product, 2017. https://www.vasco.com/products/two-factor-authenticators/hardware/one-button/index.html.

[vdE17] Benjamin Dumke von der Ehe. go-unicornify overview, November 2017.

[Vib] Viber Encryption Overview. `https://www.viber.com/security-overview`.

[Vuz17] Vuzix m300 smart glasses, 2017. https://www.vuzix.com/products/m300-smart-glasses.

[VVBVS15] Wouter Van Vlaenderen, Jens Brulmans, Jo Vermeulen, and Johannes Schöning. Watchme: A novel input method combining a smartwatch

and bimanual interaction. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 2091–2095, New York, NY, USA, 2015. ACM.

[VWO+17] Elham Vaziripour, Justin Wu, Mark O'Neill, Jordan Whitehead, Scott Heidbrink, Kent E. Seamons, and Daniel Zappala. Is that you, alice? A usability study of the authentication ceremony of secure messaging applications. In *Symposium on Usable Privacy and Security*, pages 29–47, 2017.

[wae16] White paper: Whatsapp encryption overview. `https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf`, 2016.

[Wei06] Daphna Weinshall. Cognitive authentication schemes safe against spyware. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2006.

[Wek17] Weka: Data mining software in java, 2017. http://www.cs.waikato.ac.nz/ml/weka/.

[Wha] WhatsApp: End-to-end encryption. `https://faq.whatsapp.com/en/general/28030015?lang=en`.

[WHWR16] Q. Wang, S. Hu, J. Wang, and K. Ren. Secure surfing: Privacy-preserving speeded-up robust feature extractor. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 700–710, June 2016.

[Wil66] L. G. Williams. The effect of target specification on objects fixated during visual search. *Perception & Psychophysics*, 1(5):315–318, Sep 1966.

[WWB+05a] Susan Wiedenbeck, Jim Waters, Jean-Camille Birget, Alex Brodskiy, and Nasir Memon. Authentication using graphical passwords: effects of tolerance and image choice. In *Proceedings of the Symposium on Usable Privacy and Security*, 2005.

[WWB+05b] Susan Wiedenbeck, Jim Waters, Jean-Camille Birget, Alex Brodskiy, and Nasir Memon. Passpoints: Design and longitudinal evaluation of a graphical password system. *Int. J. Hum.-Comput. Stud.*, 63(1-2):102–127, July 2005.

[WWB+05c] Susan Wiedenbeck, Jim Waters, Jean-Camille Birget, Alex Brodskiy, and Nasir Memon. Passpoints: Design and longitudinal evaluation of a graphical password system. *International Journal of Human-Computer Studies*, 63(1):102–127, 2005.

[XPFM16] Yi Xu, True Price, Jan-Michael Frahm, and Fabian Monrose. Virtual u: Defeating face liveness detection by building virtual models from your public photos. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 497–512, Austin, TX, 2016. USENIX Association.

[YCBL14] 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, Cambridge, MA, USA, 2014. MIT Press.

[YN04] Yangli Hector Yee and Anna Newman. A perceptual metric for production testing. In *ACM SIGGRAPH 2004 Sketches*, SIGGRAPH '04, pages 121–, New York, NY, USA, 2004. ACM.

[YPL15] H. Yoon, S. H. Park, and K. T. Lee. Exploiting ambient light sensor for authentication on wearable devices. In *2015 Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec)*, pages 95–100, Oct 2015.

[ZAH15] Ziming Zhao, Gail-Joon Ahn, and Hongxin Hu. Picture gesture authentication: Empirical analysis, automated attacks, and scheme evaluation. *ACM TISSEC*, 17(4):14, 2015.

[ZF14] Matthew D. Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*, pages 818–833. Springer International Publishing, Cham, 2014.

[ZFSK14] X. Zhao, T. Feng, W. Shi, and I. A. Kakadiaris. Mobile user authentication using statistical touch dynamics images. *IEEE Transactions on Information Forensics and Security*, 9(11):1780–1789, Nov 2014.

[ZLX+14] 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 - Volume 1*, NIPS'14, pages 487–495, Cambridge, MA, USA, 2014. MIT Press.

# VITA

## MOZHGAN AZIMPOURKIVI

| | |
|---|---|
| 2012-present | Ph.D., Computer Science<br>Florida International University<br>Miami, Florida |
| 2015 | M.S., Computer Science<br>Florida International University<br>Miami, Florida |
| 2011 | M.S., Information Technology<br>Sharif University of Technology<br>Kish Island, Iran |
| 2009 | B.S., Software Engineering<br>Alzahra University<br>Tehran, Iran |

PUBLICATIONS AND PRESENTATIONS

[1] Azimpourkivi, Mozhgan, Umut Topkara, and Bogdan Carbunar. "A secure mobile authentication alternative to biometrics." In Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC), pp. 28-41. ACM, 2017.
[2] Azimpourkivi, Mozhgan, Umut Topkara, and Bogdan Carbunar. "Camera based two factor authentication through mobile and wearable devices." Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 1 (IMWUT), no. 3 (2017): 35.
[3] Rahman, Mahmudur, Mozhgan Azimpourkivi, Umut Topkara, and Bogdan Carbunar. "Video liveness for Citizen journalism: attacks and defenses." IEEE Transactions on Mobile Computing 16, no. 11 (2017): 3250-3263.
[4] Potharaju, Rahul, Bogdan Carbunar, Mozhgan Azimpourkivi, Venugopal Vasudevan, and S. S. Iyengar. "Infiltrating social network accounts: attacks and defenses." In Secure System Design and Trustable Computing, pp. 457-485. Springer, Cham, 2016.
[5] Carbunar, Bogdan, Mizanur Rahman, Mozhgan Azimpourkivi, and Debra Davis. "GeoPal: Friend Spam Detection in Social Networks Using Private Location Proofs." In 2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), pp. 1-9. IEEE, 2016.
[6] Rahman, Mahmudur, Mozhgan Azimpourkivi, Umut Topkara, and Bogdan Carbunar. "Liveness verifications for citizen journalism videos." In Proceedings of

the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec), p. 17. ACM, 2015.
[7] Jain, Paras, Shang-Tse Chen, Mozhgan Azimpourkivi, Duen Horng Chau, and Bogdan Carbunar. "Spotting Suspicious Reviews via (Quasi-) clique Extraction." Appeared in IEEE Symposium on Security and Privacy 2015, 09 2015.
[8] Khalkhali, Iman, Reza Azmi, Mozhgan Azimpour-Kivi, and Mohammad Khansari. "Host-based web anomaly intrusion detection system, an artificial immune system approach." International Journal of Computer Science Issues (IJCSI) 8, no. 5 (2011): 14.
[9] Azimpour-Kivi, Mozhgan, and Reza Azmi. "A webpage similarity measure for web sessions clustering using sequence alignment." In 2011 International Symposium on Artificial Intelligence and Signal Processing (AISP), pp. 20-24. IEEE, 2011.
[10] Azimpour-Kivi, Mozhgan, and Reza Azmi. "Applying Sequence Alignment in Tracking Evolving Clusters of Web-Sessions Data: An Artificial Immune Network Approach." In 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks, pp. 42-47. IEEE, 2011.