Biometric Security

Robert Hunt, Jeremy Kalas, Patrick Lowe, Alec Stewart

May 12, 2016

1 Introduction

Creating and maintaining a secure password is the largest problem in any password-protected system. When trying to remember their password of at least eight characters with one capital-letter and one punctuation, many users simply write that password down on a sticky-note for any adversarial passerby to see [8]. A different facet of this same problem can be seen in mobile devices such as the iPhone, iPad, or any Android device which all also attempt to secure the information that their users keep on the phone through password protection; however, unlike the sticky-note example, mobile devices face the added difficulty of frequent access.

Mobile users can unlock their device as frequently as a few times a minute. Many devices seek to fix this problem by allowing users the ability to set how much time must elapse between each access before requiring a password; but, from a security standpoint, prolonged periods of time where data remains completely unprotected on the device is unconscionable.

To find a balance between ease of access and security, most modern devices have implemented biometric checks, usually in the way of a static fingerprint scan, in addition to a password protocol. The hope with this 'two-key' system is that by using the biometric check as a key for frequent access and the password itself for administrative actions (like changing the password itself or adding biometric data), the system will allow users to more conveniently set much longer passwords without adding the hassle of having to frequently use that password to access the device. The security of this system thus heavily depends on the security of the biometric data. Given that most major mobile devices rely on a fingerprint scanner for biometric verification of a user, this report seeks to analyze the security of fingerprint verification, especially in comparison to short 4 digit passcodes that are standardy used in mobile devices.

2 Previous Work

Fingerprint scanners on cellphones have recently been targeted in attempts to gain unauthorized access to a user's phone. The most common type of attack is called a spoof, which is the use of an artificial biometric input that copies the biometric of an authorized user in order to gain access [2]. Creating a spoof of a fingerprint requires both the acquisition of an authorized user's fingerprint as well as the construction of an artificial fingerprint.

One potential method of obtaining the user's fingerprint is by lifting prints off of a physical surface, such as glass or the user's phone itself. Another recently developed method can obtain a user's fingerprint by taking high definition images of a user's fingerprint and using fingerprint identification software. This attack was recently used to construct the fingerprint of German Defense Minister Ursula von der Leyen. The attacker was able to construct the Defense Minister's fingerprints from photographs taken from 3 meters away [7].

Once an attacker obtains a user's fingerprint, the attacker then must create an artificial biometric that can provide the fingerprint input to the targeted device. Simple household materials can be used in creating these artificial fingerprints. A mold of the target's fingerprint taken in Play-Doh can pass the iPhone's fingerprint sensor as being the target's fingerprint [10]. In the demonstration of the attack that obtained a the German Defense Minister's fingerprint from photos used common molding materials, namely wood glue and plaster, to create an artificial biometric with the target's fingerprint that was able to be recognized by a fingerprint scanner as the target's fingerprint. While these artificial biometrics made from molding materials seem to be effective, they can be somewhat costly to create, especially in terms of time. One alternative method of creatng artificial fingerprints requires conductive ink and a special type of paper, but using these materials and a household printer an artificial fingerprint can be quickly created that will be accepted by many phone fingerprint scanners [9].

Fingerprint authentication is widely used in many mobile devices by many manufacturers. Interestingly, some attacks work only on certain manufacturer's sensors. Android seems to have a great deal more security issues with fingerprint scanners than Apple. Android phones have been found to be vulnerable to an attack that remotely steal fingerprints from targeted phones. Additionally, artificial fingerprints created using conductive ink and paper consistently pass Android fingerprint verification whereas Apple's Touch ID does not always reliably verify the artificial fingerprint. While Apple's Touch ID may be more resistant to artificial fingerprint attacks, there are still attacks that work as shown by the attacks mentioned previously. Fingerprint verification offers benefits to users as it is easy to use and does not require the user to carry secrets, however fingerprint verification is not considered resilient to a variety of attacks, and may not outperform standard password verification [12].

3 System Overview

Before describing our threat model and relevant vulnerabilities, we describe the details of how biometric security works currently in Apple iOS devices and well as in Android devices.

3.1 Apple Touch ID

We now describe the key steps and components in Apple's Touch ID verification.

3.1.1 Touch ID Verification Process

The fingerprint scanner is only active when the capacitive steel ring that surrounds the home button detects an object that has a capacitance similar to that

of a human finger. The capacitance check then triggers an conductive imaging array to scan the object currently over the fingerprint scanner. The resulting raster scan image is temporarily stored in encrypted memory within the Secure Enclave while it is vectorized for analysis, following which it is discarded. Only a model of the fingerprint as a collection of nodes is stored permanently in the encrypted memory of the Secure Enclave, and Apple claims that constructing a user's fingerprint from this model is not feasible [4]. Touch ID allows for five unsuccessful attempts at fingerprint verification before Touch ID is disabled and will no longer unlock the device.

3.1.2 The Secure Enclave

The Secure Enclave is a coprocessor with its own secure boot and personalized software update separate from the application processor. All cryptographic operations for key management are handled by the Secure Enclave, and the Secure Enclave maintains the integrity of data protection even if the kernel has been compromised [4]. The Secure Enclave comes with its own encrypted memory, encrypted using a unique id that not even Apple knows for each device. In the Touch ID system, the Secure Enclave is responsible for handling the processing and ultimate verification of a fingerprint scan, as well as storing a representation of fingerprints that can be used to unlock the phone.

3.1.3 Touch ID Implementation Details

Aside from the actual verification process, there are other facets of the Touch ID system that are worth noting. First, Touch ID does not unlock the user's phone in all circumstances. Most notably, after the device has not been unlocked for more than 48 hours or the device has been restarted, Touch ID will not provide access to the phone, and instead a passcode is required. Additionally, Touch ID can be disabled remotely using Apple's Find My iPhone application [3]. Touch ID allows for up to five fingerprints to be enrolled and allow access to the phone, however the device's passcode is required to enroll a new fingerprint in Touch ID [4]. Touch ID can also be used for verification by third party applications. Third party applications can use the system provided API to ask a user to authenticate themselves via Touch ID.The application is only notified as to whether the Touch ID authentication was successful or not, and the application cannot access Touch ID or the data associated with the enrolled fingerprints. However, developers of third party applications can set an option that requires that Touch ID API operations do not fall back to a password or device passcode, meaning that a user can have an unlimited number of Touch ID verification attempts with this option [4].

3.2 Android Security

3.2.1 Architecture

The Android ecosystem is composed of an open source software project and individual hardware implementations. Because the hardware and software is separately controlled, the system architecture is not as tightly unified as it is for Apple. Android provides a fingerprint hardware abstraction layer (HAL) as an interface between hardware (sensors) and software (applications). Hardware

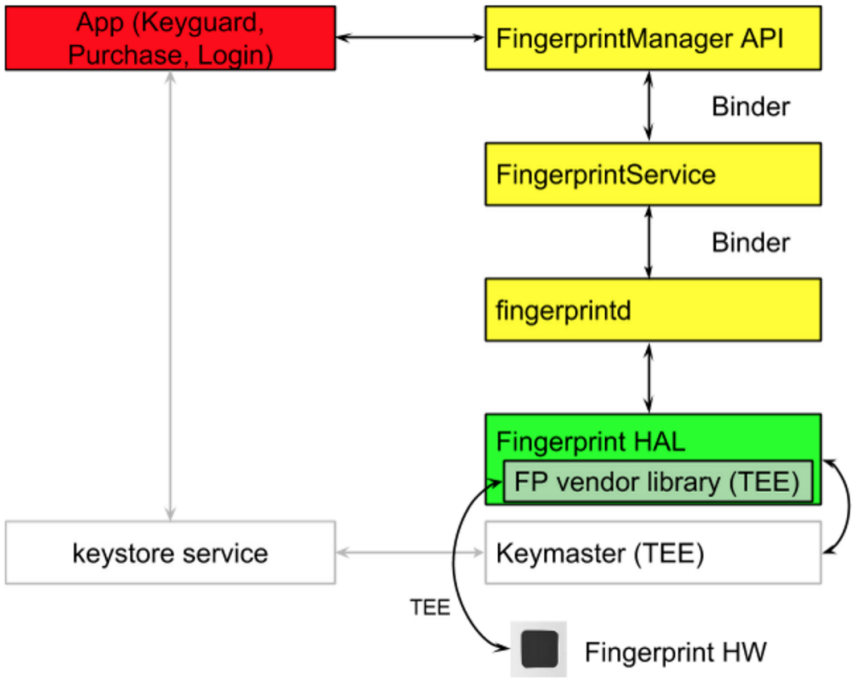

Figure 1: This diagram shows how the different Android components and services interact with eachother.

vendors are free to implement their own library that can communicate with the fingerprint daemon. Applications use the FingerprintManager API to access the services of the Fingerprint vendor library through the HAL.

3.2.2 Android Fingerprint Verification Process

An authenticate or enroll function call will trigger the fingerprint sensor to listen for a touch. The user then places their finger on the sensor and the vendor library converts the sensed print into a template. If this was caused by an enroll function call the template will be stored in a Trusted Execution Environment (TEE). If this was caused by an authenticate function call the template will be compared against the enrolled templates stored in the TEE. The results of this comparison will be passed to the Fingerprint HAL by notifying the fingerprintd (the fingerprint daemon) [5]. Applications can access the results of the fingerprint operation through the FingerprintManager API. The actual implementation and execution of enrollment and authentication is vendor specific.

3.2.3 Trusted Execution Environment

The Trusted Execution Environment (TEE) is a partitioned software environment with higher security. It is composed of an operating system running on a processor that supports TEE, drivers for the Android (Linux) kernel for applications running on Trusty OS, and libraries that allow for the communication

between applications inside and outside of the TEE [6]. TEE is the Android equivalent of Apple's Secure Enclave. Both the TEE and the Secure Enclave work by isolating the authentication and storage of sensitive information away from the rest of the general system. However, because Android is an open source software platform there are variations in the actual implementation of TEEs. The TEE OS is not required to run on a physically separate coprocessor. The Android specifications allow for the TEE OS to run on a sandboxed virtualization of the main processor. This can work assuming that the memory, registers, and fuses that the TEE OS can access are not accessible by general Android processes. Any process running in the general Android OS that tries to access these secret memory locations must delegate the operation to the TEE OS. Allowing both the TEE and Android to run on the same physical hardware presents a risk that improper implementation could lead to memory leaks.

Apple's Secure Enclave provides a coprocessor specifically for fingerprint authentication. Android's TEE is more general and can conceivably run any application that is developed for it. The TEE OS provides API's for trusted applications to run on the TEE processor and general applications to use services provided by the trusted applications. While there may be increased functionality, this also presents the risk for software in the TEE to have access to privileges that it doesn't strictly need. The TEE should also be encrypted, but the choice of TEE OS and implementation is determined by the vendor so there are no guarantees on correct/consistent implementation.

One security feature that Android's TEE has that Apple's secure enclave lacks is a validation of the TEE on every system boot. The bootloader signs and verifies the TEE image on every boot. This prevents the modification of the TEE between boots unless the bootloader is also modified.

3.2.4 Vulnerabilities

Despite the security features in Android's Fingerprint specification there are several vulnerabilities. The first are for timing attacks. The Android specifications note that there will be time variations depending on the number of templates stored on a single device [5]. There may also be time variations based on how far through the matching process a template gets. This is likely device specific as the actual implementation is vendor specific.

The second is for fingerprint backdoors [13]. The service that reports how many templates are enrolled is separate from the authentication service in the TEE. It is possible to display N enrolled templates to the user when in reality greater than N templates are enrolled. The third is for fingerprint data storage. While the Android specification has all sensitive data encrypted in the TEE, in practice vendors may not implement encryption and store data in plain text and outside the TEE [13].

The fourth is for fingerprint sensor exposure. The actual implementation of sensing fingerprints, converting them into templates, and authenticating them against enrolled templates is vendor specific. The Android specification has all of this hidden in the TEE. However in practice vendors have used insecure implementations with public API's that allow general applications to scrape data that should be hidden [13].

While Android has specifications for a secure system, ultimately the implementation is up to the vendor which has been shown to result in insecure

systems [11].

4 Security and Attack Models

To help understand potential threats for a biometrically-secured device we have devised four different games that describe the different capabilities of an attacker. In each game the attacker is considered successful if the biometrically-secured device is easier to open than a device secured by a four digit passcode.

The first game we consider is the Physical Access Game. The Physical Access Game is a two-stage game in which the adversary has physical access to a finger that the fingerprint reader accepts. In the first stage, the examiner provides physical access to the finger and the adversary is able to examine the finger and make molds or take pictures. In the second stage of the game, the examiner removes access to the finger. The adversary is allowed to use any models or molds that were created in stage one. The adversary wins the Physical Access Game if they are able to unlock the device with a higher probability than randomly guessing a four digit passcode.

The next game we consider is the Full Visual Access Game. In the Full Visual Access Game, the examiner provides high-resolution visual access to a finger or access to high-quality fingerprints that can unlock the device. The adversary can use any software (such as photo editing software or fingerprint databases) or hardware (such as 3D or 2D printers) to create molds from visual imagery of the finger. The adversary wins the Full Visual Access Game if given full visual access, they are able to unlock the device with a higher probability than randomly guessing a four digit passcode.

The third game we consider is the Partial Visual Access Game. In the Partial Visual Access Game, the examiner provides partial fingerprints of a finger that can unlock the device. The adversary can again use any software or hardware to create molds from the partial visual imagery of the finger. The adversary wins the Partial Visual Access Game if given access to partial fingerprints, they are able to unlock the device with a higher probability than randomly guessing a four digit passcode.

The last game we consider is the No Access Game. In the No Access Game the adversary is given the device without any additional information about any fingers that can unlock the device. The adversary wins the No Access Game if they are able to unlock the device using the biometric sensor with a higher probability than randomly guessing a four digit passcode.

We think that the appropriate game to choose depends on the application. For example a company considering whether iPhones with biometric security are secure enough for corporate phones with confidential data might only consider devices that are unexploitable in the Partial Visual Access Game. On the other hand consumers who might have financial information (such as Apple Pay) may consider devices that are secure in the No Access Game satisfactory for their needs.

5 Attacks and Vulnerabilities

5.1 Adversarial Strategies to Proposed Games

We now describe potential attacks and strategies for the adversary to take in each of the games that we described above. While the games described can apply to any biometrically-secured device, in our discussion of attacks and vulnerabilities we focus on Apple and TouchID.

We first consider the Physical Access Game. In the physical access game there are many documented attacks using molds using flour, wood glue, or play-doh. We made several attempts to reproduce these attacks on both the iPhone 6 and the iPhone 6s but only our attempt on the iPhone 6s using wood glue without a heat gun was successful. Current devices are definitely susceptible to attacks where the adversary uses molds, and these attacks can be conducted by an adversary at low cost. The refinement of checks to determine whether a real finger is unlocking the phone, such as a temperature check, a finer conductivity check, or an opacity check would limit the ability for an adversary to use common materials as a mold.

We now consider potential attacks an adversary can make in the Full Visual Access Game. While we were not available to procure conductive ink or similar materials used in practice, there have been several documented attacks done by security researchers. In order to fake a fingerprint from a high-resolution image researchers are able to print the image using either conductive ink or graphene dust to make a conductive cast that can be used to trick the sensor.

Depending on the quality of the partial prints, in the Partial Visual Access Game an adversary can perform similar attacks that are possible in the Full Visual Access Game.For example an adversary might be able to stitch together partial imagery of a fingerprint using photo editing software to create an image of a complete fingerprint. However, if this is not the case we consider new vulnerabilities based on the underlying nature of human fingerprints.

Apple reports that with one fingerprint stored on a device, there is a 1 in 50,000 chance that a random fingerprint that is not the stored fingerprint will pass verification by Touch ID [4]. However, there are many different types of fingerprints, with the major three types being arch, loop, and whorl. Additionally, according to Apple the categorization of fingerprint scan as one of these three basic types is a part of the analysis of a fingerprint by Touch ID [3]. Thus, we propose that the security of Touch ID is reduced when the type of the stored fingerprint is known, as it would be if we had access to partial fingerprints such as in the Partial Visual Access Game.

In the global population, it is estimated that 41% of people have a whorl fingerprint, 53% of people have a loop fingerprint, and 6% of people have an arch fingerprint [1]. Thus if the chance of a random fingerprint passing verification is 1 in 50,000, then reducing the candidates to randomly select from to only be the fingerprints that are of the same type as the stored fingerprint will increase the chance of the selected fingerprint passing verification. If the stored fingerprint is an arch, then there is a 1 in 3000 chance of a random arch candidate passing verification. Similarly, whorl candidates for a whorl stored fingerprint have a 1 in 20500 chance of passing verification, and loop candidates for a loop stored fingerprint have a 1 in 26500 chance of passing verification. In all three cases, the security of Touch ID is greatly reduced by knowing the type of fingerprint. For

stored fingerprints of type arch, the chance of a random guess passing verification is greater than the chance of a random guess passing verification for a four digit passcode.

5.2 Other Types of Attacks

We now consider new attacks that can be used to weaken the security of biometrically-secured devices.

5.2.1 Proposal for a Timing Attack to Determine Fingerprint Type

We propose an attack that could potentially reveal information about a stored fingerprint given no knowledge of the fingerprint is provided to the attacker, the same scenario as the No Access Game. For the purposes of this attack we assume there is only one stored fingerprint. The fingerprint verification by Touch ID uses the categorization of a fingerprint into one of the three major types of arch, loop, and whorl, to determine if a scanned fingerprint is a viable match to a stored fingerprint [3]. The full details of the verification algorithm are private, as Apple has not released the verification algorithm to the public. However, given that categorizing a fingerprint to a type is likely less computationally difficult than matching a full print, we propose that it may be possible to determine the fingerprint type of the stored fingerprint by examining the verification return times for failed matches. The core idea of this timing attack is that if a fingerprint scan is not of the same fingerprint type as the stored fingerprint, then the verification process will return a failure faster than if the scan matches the type of the stored fingerprint. Given that an attacker has five attempts to pass verification using Touch ID, it is feasible for the attacker to test each of the three major types of fingerprints as this would require only three attempts. Additionally, if the timing attack is successful in revealing the type of the stored fingerprint, then the attacker's chances of randomly selecting a match go up significantly, as discussed above.

5.2.2 Timing Attack Results and Discussion

In attempting this timing attack, we used an iPhone 6 with the latest software version, version 9.3.1, always with one stored fingerprint. We had three volunteers each with one of the three major different fingerprint types. We ran three rounds of testing such that each fingerprint type was used for the stored fingerprint in a round. In each round we placed a non-matching print of each type of fingerprint on the scanner and recorded the phone display at thirty frames per second. The phone displays a visual indicator of four dots filling with color when the fingerprint is scanned, and the words "Try Again" appear when the verification process returns as a failure. In every test in all of the rounds seven frames, or approximately 0.233 seconds, elapsed from the visual indicator of fingerprint scanning to the visual indicator of verification failure. Our proposed timing attack was unsuccessful with our setup. Future attempts at this timing attack could be performed by recording at a higher frame rate to determine if there is a noticeable difference in return time at the higher frame rates, which may be possible.

5.2.3 Shared Memory Attack

The security of the memory shared by the secure enclave and the application processor is unspecified. While the full communication structure between these two processors is not fully disclosed by Apple, it is stated that the two processors only communicate through this shared memory and an interrupt-driven mailbox [4]. Given that Touch ID can be used by applications on the phone for verification, such as by banking applications, and that these applications do not have direct access to the Secure Enclave, it must be that the Touch ID verification result is communicated to the application processor via this shared memory. A malicious third party program could potentially gain access to this shared memory and overwrite the actual verification result such that Touch ID verification would behave improperly. Given that the security of the shared memory is unspecified, it is even theoretically possible that navigating to a web page using a mobile browser on the device could allow for an attacker to manipulate the shared memory, bypassing the need for the Apple account and password of the owner of the phone to install new software. This attack was not implemented as part of this paper as it would require modifying a device to the point of voiding the warranty in order to fully explore the security of the shared memory.

5.2.4 Third Party Vulnerability

Third party applications are allowed to use an API provided by Apple for using Touch ID to allow the phone's owner to sign in to their accounts with the third party application. However, the option of limiting the number of unsuccessful fingerprint verification attempts before requiring a password is left to the developers of the third party application. This creates a potential vulnerability in Touch ID's security as now the security promised by Touch ID's limited number of guess attempts is now dependent on the choice of a third party developer. Additionally, if an adversary were to gain access to an unlocked device for a brief period of time with a third party application that did not limit the number of Touch ID attempts already installed on the device, then the attacker would be able to brute force test many random fingerprints until finding a print that matches the device owner's. This would then provide the attacker with full access to the phone at all times, even after the initial window of permitted access ended.

6 Conclusions

While in recent years Apple and other companies have embraced biometric security as a more usable password, we believe there are serious security concerns that inhibit biometric security from being deployed in a widespread manner as the systems are implemented currently. Users can be compromised if an attacker that has a high resolution camera and can get within three feet of a target. Unlike traditional passwords once a user's biometric credentials are compromised, they are compromised forever.

Apple's approach to biometric security introduced Touch ID. With Touch ID, Apple began requiring longer passwords of at least six digits instead of the default four digit passcode length that existed previously. The use of touchID

maintains ease of access to the device for the device owner, because users can unlock their phones with a quick tap of the finger; however, this introduction of fingerprint verification opened up new potential vulnerabilities that weaken security for Apple devices, to the point where the device may be less secure than if it was guarded by a four digit passcode.

Having any knowledge of a fingerprint has the potential to allow an attacker to gain access to the phone or at least have a better chance than 1 in 50,000 of unlocking the phone through fingerprint verification. Additionally, the designs published by Apple are very vague at best and lack specifics, thus there are potentially unaddressed security concerns such as the security of the shared memory between the Secure Enclave and application processor. Given the available information on Touch ID and the vulnerabilities proposed and tested in this project, it is clear that the inclusion of Touch ID poses security risks and opens up the potential for new attacks.

Apple attempts to mediate these risks via protocols for disabling Touch ID, disabling access through fingerprint verification after 48 hours of the phone being locked, after five failed access attempts, or after a remote signal to disable Touch ID that a user can send if they have lost their phone. Full analysis on the security provided by Apple's Touch ID faces challenges due to Apple's restriction of information such that key parts of the design are private to Apple, such as the actual fingerprint verification algorithm. Full disclosure of these system design details would allow for a thorough evaluation of the actual security provided by Touch ID.

Android devices have also turned to using biometric checks in many of the phones currently on the market. Like their Apple counterpart, the systems used to implement these biometric checks are vulnerable to a user with any knowledge of the device owner's finger, and, even without this, Android's own specs note the time variations in their system opening the door for timing attacks.

Biometric security, at its best, allows for mobile device users to quickly and securely access their device without the constant hassle of having to input a long password. In current systems, however, this best case just simply doesn't happen. With the myriad of vulnerabilities seen in both Apple and Android devices related to how their biometric checks are implemented, current mobile systems only accomplish the bare minimum of what is possible with this technology, but perhaps they pave the way for the best case to be something we see in the future.

References

[1] The Fingerprints World Map, May 2016.
URL:"http://www.handresearch.com/news/fingerprints-world-map-whorls-loops-arches.htm".

[2] Andy Adler and Stephanie Schuckers. *Encyclopedia of Biometrics*, chapter Biometric Vulnerabilities, Overview, pages 160–168. Springer US, Boston, MA, 2009.

[3] Apple. About Touch ID security on iPhone and iPad, May 2016.
URL:"https://support.apple.com/en-us/HT204587".

[4] Apple. iOS Security, May 2016. URL:"https://www.apple.com/business/docs/iOS_Security_Guide.pdf".

[5] Google. Fingerprinthal, May 2016. URL:"https://source.android.com/security/authentication/fingerprint-hal.html".

[6] Google. Trusty TEE, May 2016. URL:"https://source.android.com/security/trusty/index.html".

[7] Alex Hern. Hacker Fakes German Minister's Fingerprints Using Photos of her Hands, December 2014. URL:"https://www.theguardian.com/technology/2014/dec/30/hacker-fakes-german-ministers-fingerprints-using-photos-of-her-hands".

[8] Keenan. Hidden Risks of Biometric Identifiers and How to Avoid Them. Black Hat, 2015.

[9] Andrew Liszewski. How to easily hack a smartphone with an inkjet printer and conductive ink. *Gizmodo*, mar 2016. http://gizmodo.com/how-to-easily-hack-a-smartphone-with-an-inkjet-printer-1763261331.

[10] Jeff John Roberts. This guy unlocked my iphone with play-doh. *Fortuen*, apr 2016. http://fortune.com/2016/04/07/guy-unlocked-iphone-play-doh/.

[11] Di Shen. Exploiting Trustzone on Android. Technical report, KeenLab, 2015.

[12] Frank Stajano. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication. Technical report, University of Cambridge, 2012.

[13] Yulong'Zhang, Zhaofeng'Chen, Hui'Xue, and Tao'Wei. Fingerprints On Mobile Devices: Abusing and Leaking. Technical report, FireEye Labs, 2015.