Royal Holloway, University of London

Doctoral Thesis

On Enhancing the Security of Time Constrained Mobile Contactless Transactions

Author:
Iakovos Gurulian

Supervisor:
Prof. Konstantinos Markantonakis

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Information Security Group
Department of Mathematics

September 11, 2018

Declaration of Authorship

This doctoral study was conducted under the supervision of Prof. Konstantinos Markantonakis.

The work presented in this thesis is the result of original research carried out by myself, or in collaboration with others, whilst enrolled in the Department of Mathematics as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Signed:

Date:

List of Publications

Directly Related to the Thesis

[1] I. Gurulian, R. N. Akram, K. Markantonakis, and K. Mayes. Preventing Relay Attacks in Mobile Transactions Using Infrared Light. In *Proceedings of the Symposium on Applied Computing*, SAC '17, pages 1724–1731, New York, NY, USA, 2017. ACM.

[2] I. Gurulian, G. P. Hancke, K. Markantonakis, and R. N. Akram. May the force be with you: Force-based relay attack detection. In T. Eisenbarth and Y. Teglia, editors, *Smart Card Research and Advanced Applications*, pages 142–159, Cham, 2018. Springer International Publishing.

[3] I. Gurulian, K. Markantonakis, R. N. Akram, and K. Mayes. Artificial Ambient Environments for Proximity Critical Applications. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ARES '17, pages 5:1–5:10, New York, NY, USA, 2017. ACM.

[4] I. Gurulian, K. Markantonakis, L. Cavallaro, and K. Mayes. You can't touch this: Consumer-centric android application repackaging detection. *Future Generation Computer Systems*, 65:1 – 9, Dec. 2016. Special Issue on Big Data in the Cloud.

[5] I. Gurulian, K. Markantonakis, E. Frank, and R. N. Akram. Good Vibrations: Artificial Ambience-Based Relay Attack Detection. In *The 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, TrustCom '18. IEEE, Aug. 2018.

[6] I. Gurulian, K. Markantonakis, C. Shepherd, E. Frank, and R. N. Akram. Proximity assurances based on natural and artificial ambient environments. In P. Farshim and E. Simion, editors, *Innovative Security Solutions for Information Technology and Communications*, pages 83–103, Cham, 2017. Springer International Publishing.

[7] I. Gurulian, C. Shepherd, E. Frank, K. Markantonakis, R. N. Akram, and K. Mayes. On the Effectiveness of Ambient Sensing for Detecting NFC Relay Attacks. In *2017 IEEE Trustcom/BigDataSE/ICESS*, pages 41–49, Aug. 2017.

[8] C. Shepherd, G. Arfaoui, I. Gurulian, R. P. Lee, K. Markantonakis, R. N. Akram, D. Sauveron, and E. Conchon. Secure and Trusted Execution: Past, Present, and Future - A Critical Review in the Context of the Internet of Things and Cyber-Physical Systems. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 168–177, Aug. 2016.

[9] C. Shepherd, I. Gurulian, E. Frank, K. Markantonakis, R. N. Akram, K. Mayes, and E. Panaousis. The Applicability of Ambient Sensors as Proximity Evidence for NFC Transactions. In *Mobile Security Technologies, IEEE Security and Privacy Workshops*, MoST '17. IEEE, May 2017.

Other Publications

[10] G. Haken, K. Markantonakis, I. Gurulian, C. Shepherd, and R. N. Akram. Evaluation of Apple iDevice Sensors As a Potential Relay Attack Countermeasure for Apple Pay. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, CPSS '17, pages 21–32, New York, NY, USA, 2017. ACM.

[11] D. Jayasinghe, K. Markantonakis, I. Gurulian, R. N. Akram, and K. Mayes. Extending EMV Tokenised Payments to Offline-Environments. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 443–450, Aug. 2016.

[12] H. Mansor, K. Markantonakis, R. N. Akram, K. Mayes, and I. Gurulian. Log Your Car: The Non-invasive Vehicle Forensics. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 974–982, Aug. 2016.

[13] H. Mansor, K. Markantonakis, R. N. Akram, K. Mayes, and I. Gurulian. Log Your Car: Reliable Maintenance Services Record. In K. Chen, D. Lin, and M. Yung, editors, *Information Security and Cryptology: 12th International Conference, Inscrypt 2016, Beijing, China, November 4-6, 2016, Revised Selected Papers*, pages 484–504, Cham, 2017. Springer International Publishing.

[14] A. Umar, I. Gurulian, K. Mayes, and K. Markantonakis. Tokenisation Blacklisting Using Linkable Group Signatures. In R. Deng, J. Weng, K. Ren, and V. Yegneswaran, editors, *Security and Privacy in Communication Networks: 12th International Conference, SecureComm 2016, Guangzhou, China, October 10-12, 2016, Proceedings*, pages 182–198, Cham, 2017. Springer International Publishing.

"So the problem is not so much to see what nobody has yet seen, as to think what nobody has yet thought concerning that which everybody sees."

Arthur Schopenhauer

Abstract

Relay attacks are passive man-in-the-middle attacks during which an attacker is extending the communication distance of two genuine devices by relaying communication messages between them, without the legitimate user's consent. In the field of smart cards, distance bounding protocols have been proposed as an effective countermeasure. For smartphones, distance bounding protocols may not work due to the multitude of hardware vendors and background processes. Instead, sensing the natural ambient environment has been proposed. However, previously proposed solutions may not be applicable in scenarios where industry imposed time constraints apply, e.g. in the cases of EMV contactless payments and transport ticketing, where a transaction should typically complete within 300–500ms.

In this thesis, the applicability of the natural ambient sensing as a Proximity and Relay Attack Detection (PRAD) mechanism in time-restricted contactless transactions (up to 500ms) is initially investigated. The use of an Artificial Ambient Environment (AAE) is proposed as a potentially more effective PRAD mechanism. Infrared light and vibration have been examined as AAE actuators.

Furthermore, two PRAD techniques that are not based on the ambient environment are proposed. First, comparing subsequent button press and release timings, performed by the genuine user on the smartphone during a transaction, and recorded by both transaction devices simultaneously. Finally, in order to protect against relay attacks initiated through a malicious application, installed on the user's smartphone, a repackaged application detection technique is proposed, based on similarity comparison of application names and icons. Repackaged applications have been found in the past to be responsible for the distribution of as much as 86% of all Android malware.

The effectiveness of all the proposed techniques has been empirically evaluated through field trials. Analysis of the collected data, using threshold- and/or machine learning-based techniques, indicates the high effectiveness of all the proposed solutions as PRAD mechanisms.

Acknowledgements

It would not have been possible to complete this thesis without the support of a number of people.

First, I would like to thank my supervisor, Prof. Konstantinos Markantonakis, as well as Dr Raja Naeem Akram for all their help, support, and guidance throughout the years. Along with them, I would like to express my deepest gratitude to my family and Ana for their love and support. It really meant a lot!

My very special thanks to Carlton Shepherd for the cooperation, the long working days, and the missed flights, and also to Dr Eibe Frank, Dr Gerhard Hancke, Dr Lorenzo Cavallaro, Dr Assad Umar, Dr Danushka Jayasinghe, Dr Hafizah Mansor, and Prof. Keith Mayes.

Thanks also to the rest of the people of the ISG Smart Card and IoT Security Centre and all the volunteers who took part in the experiments that were carried out as part of this work.

Last but not least, a very big thank you to Maria Giakoumatou, Daniela Katramada Kamarieri, Constantine Skourlis, Harris Tamvakis, and all of my friends who stood by my side and didn't stop motivating and advising me.

Contents

List of Figures

List of Tables

List of Abbreviations

2FA	**2** (Two) **F**actor **A**uthentication
AAC	**A**rtificial **A**mbient **C**hannel
AAE	**A**rtificial **A**mbient **E**nvironment
AES	**A**dvanced **E**ncryption **S**tandard
AID	**A**pplication **ID**
AOSP	**A**ndroid **O**pen **S**ource **P**roject
APDU	**A**pplication **P**rotocol **D**ata **U**nit
API	**A**pplication **P**rogramming **I**nterface
ART	**A**ndroid **R**un**T**ime
AUC	**A**rea **U**nder **C**urve
AUROC	**A**rea **U**nder the **R**eceiver **O**perating **C**haracteristic
C-Corr	**C**ross-**C**orrelation
CBC	**C**ipher **B**lock **C**haining
Coh	**C**oherence
CSV	**C**omma **S**eparated **V**alues
DVM	**D**alvik **V**irtual **M**achine
ED	**E**uclidean **D**istance
EER	**E**qual **E**rror **R**ate
EMV	**E**uropay **M**astercard **V**isa
FA	**F**alse **A**ccept
FAR	**F**alse **A**cceptance **R**ate
FFT	**F**ast **F**ourier **T**ransform
FR	**F**alse **R**eject
FRR	**F**alse **R**ejection **R**ate
FSR	**F**orce **S**ensitive **R**esistor
GHz	**G**iga**h**ertz
GPIO	**G**eneral **P**urpose **I**nput **O**utput
GPS	**G**lobal **P**ositioning **S**ystem
GRV	**G**eomagnetic **R**otation **V**ector
GUI	**G**rapgical **U**ser **I**nterface
HCE	**H**ost-based **C**ard **E**mulation
HW	**H**ard**w**are
IoT	**I**nternet **o**f **T**hings
JVM	**J**ava **V**irtual **M**achine
kHz	**K**ilo**h**ertz
LED	**L**ight-**E**mitting **D**iode
m	**M**etres
ms	**M**illi**s**econds
MAE	**M**ean **A**bsolute **E**rror
MHz	**M**ega**h**ertz
NFC	**N**ear **F**ield **C**ommunication
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
OS	**O**perating **S**ystem

OTP	**O**ne **T**ime **P**assword
PCC	**P**earson's **C**orrelation **C**oefficient
PIN	**P**ersonal **I**dentification **N**umber
PKCS5	**P**ublic **K**ey **C**ryptography **S**tandard **#5**
POS	**P**oint **O**f **S**ale
PRAD	**P**roximity (and) **R**elay **A**ttack **D**etection
RAM	**R**andom **A**ccess **M**emory
RBF	**R**adial **B**asis **F**unction
RFCOMM	**R**adio **F**requency **Comm**unication
RFID	**R**adio-**F**requency **Id**entification
RPi	**R**aspberry **Pi**
RSA	**R**ivest–**S**hamir–**A**dleman
SDK	**S**oftware **D**evelopment **K**it
SID	**S**ystem **ID**
SE	**S**ecure **E**lement
SGS4	**S**amsung **G**alaxy **S4**
SGS5 mini	**S**amsung **G**alaxy **S5 mini**
SGX	**S**oftware **G**uard e**X**tensions (refers to Intel SGX)
SHA	**S**ecure **H**ash **A**lgorithm
SMO	**S**equential **M**inimal **O**ptimisation
SMS	**S**hort **M**essage **S**ervice
SVM	**S**upport **V**ector **M**achine
T-FD	**T**ime-**F**requency **D**istance
TA	**T**rue **A**ccept
TAR	**T**rue **A**cceptance **R**ate
TR	**T**rue **R**eject
TRR	**T**rue **R**ejection **R**ate
TI	**T**ransaction **I**nstrument (Malicious or Genuine)
TI′	**T**ransaction **I**nstrument (Malicious or Genuine)
TT	**T**ransaction **T**erminal (Malicious or Genuine)
TT′	**T**ransaction **T**erminal (Malicious or Genuine)
TTP	**T**rusted **T**hird **P**arty
UDP	**U**ser **D**atagram **P**rotocol
UID	**U**nique **ID**
USB	**U**niversal **S**erial **B**us
μ**s**	**M**icro**s**econds

In memory of my grandfather
Iakovos Gurulian (1923–2018)

In memory of Avgi Tzifopoulou
(1918–2019)

Chapter 1

Introduction

Relay attacks [54, 55, 159] are passive man-in-the-middle attacks, aiming to extend the physical distance of devices involved in a transaction beyond their operating environment. Contactless smart cards [71–73, 97], as well as mobile phones [54, 55, 108, 130–132], are susceptible to relay attacks. Using such attacks, an attacker can gain unauthorised access to services and facilities that a genuine user is eligible for, like payments and access to buildings.

As an example, in a relay attack on a payment transaction, one way of extending the operating distance between two transaction devices is by using equipment that masquerades as legitimate devices to both the payment terminal and instrument device (i.e. the client device — e.g. a smart card or a mobile phone that is used for the completion of the payment transaction), as shown in Figure 1.1. At present, additional security mechanisms like fingerprint scanning and Personal Identification Number (PIN) entry may also be required in order to perform a contactless mobile transaction for a payment, transport ticketing, and similar services. However, even the use of PINs and biometrics cannot prevent relay attacks (see the Mafia fraud attack [36]).

Figure 1.1: Overview of a Relay Attack

For contactless smart cards, distance bounding protocols [57, 128, 151] and variants of such [72] have been proposed as an effective countermeasure against relay attacks. This is still an active research domain, with new attacks and countermeasures emerging [19, 36, 74]. At the current state of the art, however, these are not easily transferable to Near Field Communication (NFC)-enabled phones, due to their high sensitivity to time delays [34, 68, 156].

In recent years, a number of proposals suggest the use of ambient sensing as an alternative Proximity and Relay Attack Detection (PRAD) mechanism against the off-the-shelf attacker [68, 106, 139, 152, 157, 158]. Such proposals rely on the collection

of data over a period of time from both transaction devices, using ambient sensors (e.g. the temperature sensor), and subsequently comparing the collected data for similarity.

Specific scenarios susceptible to relay attacks, like EMV contactless payments and transport ticketing, have industry-imposed time limits regarding the completion of a contactless transaction. In the case of EMV the limit is 500ms [50, 81, 149], and in the case of transport ticketing typically between 300 and 500ms [49, 150]. It has been forecasted that by 2020 the amount of NFC payments performed using services such as Android Pay and Apple Pay will exceed 760 million per year [92], while by that time NFC-based mobile ticketing users will reach 375 million [93]. In addition to these, other applications exist that depend on proximity and transaction time, particularly in the realm of the Internet of Things (IoT), such as taking medical equipment inventories in operating theatres. The domain of sensor networks is another closely-related area where the communication time and the proximity of sensors can be of paramount importance.

Certain types of relay attacks on mobile devices may not rely on the use of masqueraded equipment on both ends of the transaction (i.e. the transaction terminal and the transaction instrument). For example, a relay attack that uses a malicious application has been demonstrated [131]. Such relay attack scenarios require different prevention approaches. Researchers have found that approximately 86% of all Android malware is distributed through repackaged applications [171], thus, preventing the installation of such applications can act as an effective countermeasure.

The main goal of this thesis is to evaluate the effectiveness of previously proposed PRAD techniques for mobile devices and propose alternatives and/or enhancements that can potentially improve the detection/prevention of relay attacks. The aim is to prevent the opportunistic, off-the-shelf attacker, from being capable of performing relay attacks on such devices. In the course of this thesis, both parties involved in a transaction are regarded as trusted (the transaction devices and/or operators do not cooperate with an attacker who is attempting a relay attack).

1.1 Motivation and Challenges

Previously proposed PRAD techniques on mobile devices do not take into consideration industry-imposed time restrictions, therefore may not be suitable in such scenarios. Moreover, the best Equal Error Rate (EER) (i.e. the optimal trade-off between false positives and false negatives) reported by previous works is approximately 9%, meaning that 9% of genuine transactions would be detected as relay attacks and 9% of relay attacks would be accepted as genuine transactions. Therefore, they may not be suitable for security-critical applications, or applications that require a high level of reliability (e.g. mobile payments, which have to complete fast in order for the next customers to be served). Furthermore, in the presence of a context-manipulating attacker, many of the previously proposed techniques may fail [137]. Finally, many

of the previously proposed solutions do not focus on relay attacks that are initiated through a malicious application (similar to [131]).

Previously proposed solutions were initially evaluated (Chapter 3) under industry-specific timing restrictions (up to 500ms transaction time). High EERs that may not be sufficient for security-critical applications were perceived. More specifically, evaluating only the effectiveness of sensing the natural ambient environment as a means of establishing proximity evidence between two communicating devices, the best (lowest) EER was 9.2%, using the Pressure sensor (see Chapter 3). In the presence of an attacker, therefore evaluating the effectiveness of sensing the natural ambient environment as a relay attack detection mechanism, the best EER was 17.9%, using the Gyroscope sensor (Chapter 3). Similar results have also been demonstrated on iOS-based devices in the presence of an attacker [66]. Alternative solutions, suitable for such scenarios, are proposed as part of this thesis (Chapters 4, 6, 7, 8). The aim of the proposed techniques is to improve the performance of PRAD techniques in the field of smartphones against the off-the-shelf attacker, while being effective in time-constrained scenarios (like EMV payments).

1.2 Contributions

The main contributions of this thesis are:

- Evaluation of the natural ambient environment as a PRAD mechanism for transactions with a limited time-frame (up to 500ms):
  - A test-bed was implemented to evaluate various sensors on Android devices.
  - The ambient sensor measurement data analysis was evaluated under industry specifications.
  - The effectiveness of ambient sensors as a proximity detection mechanism was empirically evaluated.

  The experimentation and analysis carried out as part of this thesis showed that none of the sensors were individually suitable to be deployed as a proximity detection mechanism for NFC-based mobile transactions.

- Artificial Ambient Environment (AAE)-Based PRAD: A novel approach of PRAD is proposed, based on the generation of an AAE through peripherals of the devices involved in a transaction. One or both transaction devices are responsible for generating and/or measuring the AAE. The AAE is based on randomly generated sequences or streams. The captured data of one transaction device is compared for similarity against the captured or generated data of the other transaction device. A generic framework for generation and measurement of the AAE is proposed. In addition, this generic framework defines the

methodology of how two devices can ascertain whether they are in proximity to each other. Two AAE actuators have been investigated in terms of effectiveness and applicability, namely, infrared light and vibration:

  - Infrared light as an AAE actuator:
    * Taking the generic framework, an evaluation test-bed was designed and built using infrared as an AAE actuator.
    * The effectiveness of using infrared as an AAE actuator was empirically evaluated against six distinct relay attack scenarios. The experimental results indicate a high success rate in both proximity and relay attack detection.
  - Real-world applicability of the infrared light-based AAE PRAD:
    * Potential applications of infrared light as a PRAD mechanism for contactless transactions in mobile payments, physical access control, and transportation are described.
    * A scheme for relay attack resilient continuous Two-Factor Authentication (2FA) for host-based services authentication is proposed.
    * Protocols that follow industry standards (where applicable) for the integration of the PRAD technique are proposed. The security of all the protocols has been verified using the Scyther tool [39].
    * The performance of each of the aforementioned cases has been evaluated through practical implementation. The results indicate that the performance cost is low.
  - Vibration as an AAE actuator:
    * Vibration as an AAE actuator (Section 6.2): The sensing of short (up to 500ms) random vibration sequences, generated by one or both transaction devices, is proposed as a means of PRAD.
    * Evaluation of the proposed solution: An evaluation test-bed was implemented in order to assess the effectiveness of the proposed solution. A total of 36,000 transactions were analysed using different vibration modalities, ambient sensors, and potential attack techniques. The results indicate high effectiveness of the proposed solution and resilience against relay attacks when using certain sensors (namely, gyroscope, linear acceleration, and rotation vector).

- Force Sensing-Based PRAD:
  - A novel approach for PRAD, based on sensing the sequence and timings of subsequent virtual button presses and releases by the transaction devices, is proposed. Virtual buttons are displayed by the smartphone involved in the transaction. The user is called to press on them and both transaction devices are responsible for recording the order of the pressed buttons,

as well as the duration of: (a) each button press and (b) between two subsequent button presses. The recorded values from the two transaction devices are then compared for similarity in order to establish proximity assurances.

  - Two evaluation frameworks were deployed for evaluating the proposed solution as a proximity detection mechanism and as a relay attack detection mechanism, by subjecting it against a set of volunteers that tried to attack the scheme.

- Host-Based PRAD: In order to prevent repackaged applications from being installed on a target smartphone that could potentially be used to perform a host-based relay attack (similar to [131]):

  - A method for detecting repackaged applications is proposed, initiated on the client side prior to an application's installation. The target application is being checked against a database of legitimate applications hosted by a Trusted Third Party (TTP). If the target application does not exist in the legitimate application's database, its name and icon are compared for similarity against those of other, legitimate applications. These two elements characterise an application to the user prior to its installation, so an attacker may not alter them, in order to deceive the user.
  - Experimental evaluation demonstrated that the proposed mechanism is both effective and efficient in detecting repackaged applications.

- The source code and the collected data for all the work presented in this thesis has been made available where necessary and possible. It can be found at `https://www.dropbox.com/sh/6tqbw2xjuqznmf5/AADPRaEO2cWr-UpS2ALeFwBra?dl=0`.

1.3 Thesis Outline

The outline of this thesis is as follows:

Part I: Background

- Chapter 2: In this chapter, background information and the literature review are being discussed.

Part II: Natural Ambient Environment-Based Relay Attack Detection

- Chapter 3: In this chapter, the effectiveness of previously proposed PRAD solutions for smartphones is being evaluated in time-restricted scenarios of up to 500ms (e.g. EMV).

Part III: Artificial Ambient Environment-Based Relay Attack Detection

- Chapter 4: In this chapter, the concept of the AAE is introduced. The use of infrared light as an AAE actuator is being evaluated.
- Chapter 5: In this chapter, the applicability of infrared light as an AAE actuator is discussed through evaluation on different simulations of real-life scenarios.
- Chapter 6: In this chapter, the use of vibration as an AAE actuator is being evaluated.

Part IV: Non-Ambient Environment-Based Relay Attack Detection

- Chapter 7: In this chapter, an alternative approach to using the ambient environment as a PRAD technique is proposed. Instead, timed touches and releases on the smartphone's display, performed by the user, are used. The touch durations and their sequence are also measured by the transaction terminal through force-sensitive resistors.
- Chapter 8: In this chapter, the focus is on mitigating the risk of relay attacks that might be initiated by a malicious application. The aim is to detect and prevent the installation of potentially repackaged applications that attempt to resemble genuine applications (e.g. banking applications) to attract more victims and deploy this type of attack on a larger scale. The detection technique is based on application name and icon similarity with genuine applications.
- Chapter 9: Concluding remarks, discussion, and comparison of the proposed techniques are presented in this chapter.

Part I

Background

Chapter 2

Relay Attacks in Contactless Transactions

2.1 Introduction

The main goal of this chapter is to introduce the reader to the concept of relay attacks, summarise the related literature, and provide an overview of technologies that are used throughout this thesis. This thesis focuses mainly on relay attack detection on smartphones, where traditional distance bounding protocols that are applicable in the field of smart cards may not be as effective. Android devices are used to evaluate the proposed solutions, since the Android Operating System (OS) is open-source and provides rich Application Programming Interfaces (APIs) for controlling various hardware and other components. Thus, initially, the basics of the Android OS are described (Section 2.2).

An overview of relay attacks is provided afterwards (Section 2.3). Three are the main types of relay attacks that are commonly discussed in the literature (*Mafia Fraud Attack*, *Distance Fraud Attack*, and *Terrorist Fraud Attack*). The three types are explained and the most prominent countermeasures in the fields of smart cards and smartphones are explained. It is worth noting that in the field of smartphones the focus of the solutions that have been proposed is mainly on the Mafia Fraud Attack. The relevant literature regarding relay attack detection mechanisms for smart cards and mobile devices is also provided (Section 2.4).

Another type of relay attacks that does not fit the exact definition of the other three types and is relevant to smartphones has also been demonstrated [131]. This type of relay attack relies on malware capable of hijacking information from a user's smartphone without the user's consent/knowledge. It has been reported that approximately 86% of malware distribution is performed through repackaged applications [171]. In Chapter 8, a repackaging detection approach is proposed that can assist in the mitigation of this type of relay attack. The specifics of Android application repackaging are discussed and the related literature is listed (Section 2.5).

2.2 The Android Operating System

In this section, an overview of the Android OS is provided. It is followed by a short overview of Android applications and Host-based Card Emulation (HCE).

2.2.1 What is Android

Android is an OS developed by Google and is mainly targeting mobile devices (smartphones). As of June 2018, it is the most popular OS, holding over 41% of the whole OS market share [142] and more than 85% of the mobile device OS market share [143].

Android is open-source and uses the Linux kernel at its core [12]. Other major components of the OS are depicted in Figure 2.1. Android provides APIs for accessing a multitude of hardware components (e.g. Bluetooth, WiFi). Device vendors can choose the hardware components that an Android device uses. Different devices may be using different components (e.g. processors), and certain components may not be available on certain devices (e.g. certain ambient sensors). However, the Android guidelines require certain components to be present on all devices (e.g. a touchscreen), and strongly recommend others [5].

Applications are also supported by the Android OS. A user is capable of downloading and installing them on their devices, using application stores and other methods. The next section provides an overview of Android applications.

2.2.2 Android Applications

Android applications come in `.apk` or `.xapk` containers (basically `.zip` files). These containers include the application's bytecode, resources, assets, and libraries, as well as a folder (named `META-INF`) that holds the signature(s), generated by the developer, on different elements (e.g. classes and resources) of the respective application. Each application has to be signed using the private key(s) of the developer. The developer needs to safeguard the private key(s), since in order for an application to be updated it has to be signed using the same private key(s) as the already installed version's.

Each Android application has a package name that is used by the operating system as the differentiator factor between applications. If an application that is about to be installed shares the same package name with an already installed one, Android will perceive it as an update attempt on the last.

Android applications until Android 4.4 used to run in the Dalvik Virtual Machine (DVM), which is the equivalent of the Java Virtual Machine (JVM). Dalvik was replaced by Android Runtime (ART) in Android 5.0 [11, 12]. The languages that Android officially supports are Java, Kotlin, and C++ [62], however using C++ is discouraged by Google, except in special cases, like game engines [63].

Various sources for installing applications on a device exist. For example, applications can be downloaded from the Google Play Store, or a third-party application store. A user can also install an application file directly on their device via USB, if the

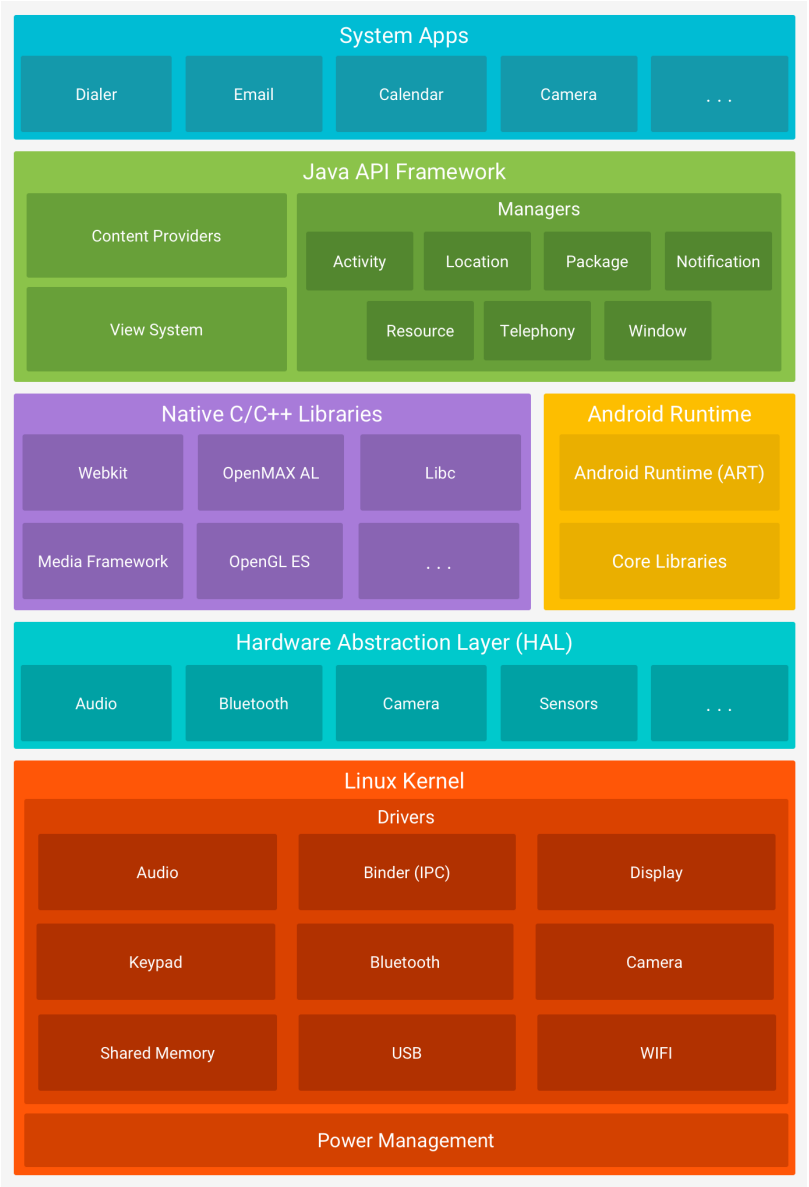


FIGURE 2.1: The Android Software Stack[1]

appropriate setting is enabled by the user, or by selecting an application file that is located in the device's storage.

Updating Installed Applications

As already mentioned, in order for an application to be installed on a device, it has to be signed with the developer's private key(s). The differentiation factor being used by the operating system in order to distinguish applications is the application's package name. If an application that is about to be installed shares the same package name with an already installed one, Android will perceive it as an update attempt of that application. If the private key(s) used to sign the new version do not match the already installed one's, the operating system will refuse to install it. However, no

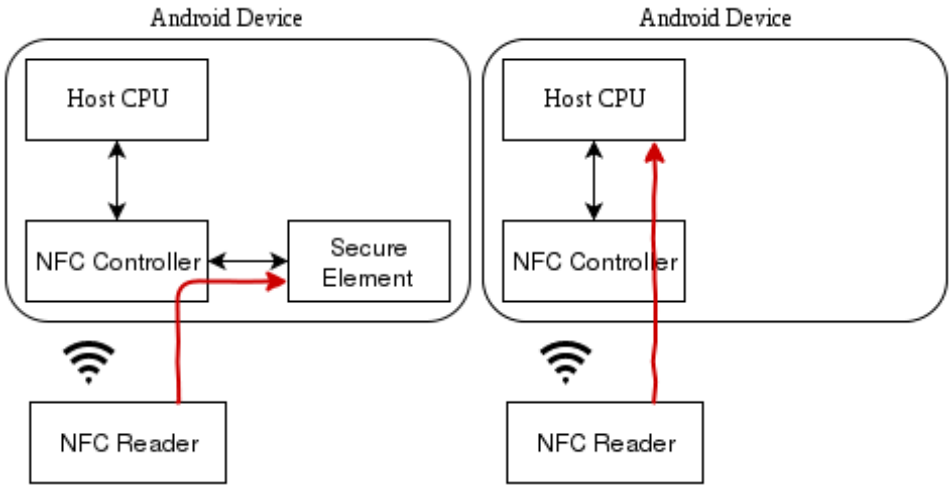
[1]Source: https://developer.android.com/guide/platform/

actions are being taken to prove the authenticity of an application when it is installed for the first time.

2.2.3 Android Host-based Card Emulation

The Android OS includes support for NFC [34]. NFC is a short-ranged contactless communication channel (up to 10cm), based on the ISO-14443 standard [87]. NFC has opened smartphone platforms to a range of application domains, particularly those previously based on smart cards.

Android's HCE allows NFC-enabled devices to act as contactless smart cards without relying on a dedicated Secure Element (SE). It was first introduced in Android 4.4 (API 19) [7] and is used by mobile applications to perform transactions through NFC, such as banking and transport-related. Its ability to emulate contactless smart cards facilitates interoperability with existing card—reader infrastructures based on NFC [80]. Using HCE, application developers do not have to have write access to an SE (which is typically not publicly available) in order to make their application widely available.



(A) SE-based (B) HCE-based

FIGURE 2.2: Card Emulation Using SE and HCE

When an SE is used, typically all of the data received by the NFC controller is routed directly to the SE (Figure 2.2a). In the case of HCE, however, data is directed to the device's CPU, which is responsible for handing it to the corresponding application via the operating system (Figure 2.2b). In both SE- and HCE-based deployments, data is passed to an application based on an Application ID (AID), as stipulated in the ISO/IEC 7816-4 specification [23].

Naturally, because HCE is software-based, it cannot provide the level of tamper-resistance that a hardware SE can [113]. Device storage may only be protected by the security mechanisms of the operating system, such as application sandboxing [2]. Therefore, it is not recommended that critical transactions, like mobile payments and

high-security access control, to be based entirely on HCE [80, 123]. In such scenarios an SE- or TEE-based solution would be more appropriate for safeguarding critical information.

Android devices do not always contain an SE or TEE. In order to make HCE more widely accessible, cloud-based storage that acts as a remote SE is one proposed solution. However, Internet connectivity is typically required and achieving authentication via a remote cloud platform can be challenging. Various proposals have been made to enhance the security of HCE, some of which are being used by stakeholders. Such methods include verification of the user and/or the hardware (e.g. using PINs or biometrics), limitations to transaction amounts, tokenised payments, operating system checks, and white-box cryptography [78].

In case an SE is available on the device, a combination of HCE and SE can be used. In this scenario, HCE can allow developers to build applications that run on the Android space instead of the SE, while using the additional security that the SE provides to store secrets and/or perform sensitive tasks (e.g. encryption/decryption). In this approach, the security is improved compared to a solely HCE-based solution, but lacks when compared against an SE-only approach. However, application developers would not require write access to the SE, as various cryptographic functions of the SE are available to the developers through API calls [9].

The use of an SE may be able to protect against many types of software and hardware attacks better than HCE. Relay attacks, however, can be effective against both, unless the appropriate countermeasures have been applied. This type of attack is discussed in the following section.

2.3 Relay Attacks

As already discussed, relay attacks aim to extend the operating distance of communicating devices beyond their intended operating environment. A wide range of applications can be affected by relay attacks, like contactless transactions and access control.

Typically, some form of relay equipment is required by an attacker to perform such attacks, but this depends on the type of the relay attack. For example, in the case of an NFC-based contactless payment using a smartphone as a payment instrument, an attacker may use a pair of a malicious (masqueraded) payment terminal and a malicious payment instrument as the relay equipment. The malicious payment terminal would have to be presented to a legitimate user (smartphone owner) and the malicious payment instrument to a distant (e.g. located at a different city) legitimate payment terminal operator. The legitimate payment terminal will initiate a transaction when the malicious payment instrument appears in its vicinity (<3cm distance in the case of an EMV payment [51]). The malicious payment instrument would then have to passively relay the first Application Protocol Data Unit (APDU) (i.e. the communication unit between a card and a reader) to the malicious payment terminal,

which is co-located with the legitimate payment instrument at a distant location (i.e. >3cm distance). The APDU would then be passed to the legitimate payment instrument through NFC, leading to the initiation of a transaction between the two legitimate devices. The relaying process should be repeated by both malicious devices for all the APDUs that are exchanged during the course of the transaction. Thus, the purpose of the relay equipment used by the attacker is to passively relay the exchanged communication messages between the legitimate devices, without being detected. For the attack to be successful, all data communicated during a transaction should be relayed between the malicious devices, as shown in Figure 1.1. If the attack is successful, the attacker may have paid for goods, using the legitimate user's account. This type of relay attack is known as the Mafia Fraud Attack.

Three types of relay attacks are commonly discussed in the literature [36]:

1. Mafia Fraud Attack
2. Distance Fraud Attack
3. Terrorist Fraud Attack

In all three types, a transaction is considered to be performed between a prover $\mathcal{P}$, and a verifier $\mathcal{V}$. In a payment transaction, the prover would be the device used for the payment (e.g. a smart card), and the verifier the Point Of Sale (POS).

In the case of the Mafia Fraud Attack, both parties involved in a transaction are considered to be honest. During this attack, an attacker relays the communication between the two parties, without their consent. For example, masqueraded relay equipment may be used by the attacker in order to relay the messages (as described above). This type of relay attack was introduced by Desmedt et al. [44] and is based on the Chess Grandmaster problem [32]. Figure 2.3 demonstrates this attack. $\mathcal{V}_{\mathcal{A}}$ denotes the malicious verifier and $\mathcal{P}_{\mathcal{A}}$ the malicious prover.

FIGURE 2.3: Mafia Fraud Attack

In the case of the Distance Fraud Attack (also known as 'Wormhole Attack' in the field of sensor networks [83]), the prover is at the same time the attacker. The goal is for the malicious prover to convince the verifier that the distance between the two parties is shorter than in reality [20]. This can be achieved, for example, by the prover transmitting the communication messages ahead of time. Figure 2.4 depicts this attack.

Finally, in the case of the Terrorist Fraud Attack [43], the prover is again also the attacker, however the communication is purposely relayed to a remote malicious prover, which is in proximity to the verifier. Figure 2.5 depicts this type of relay attack.

Another type of relay attack that has been demonstrated (see [131]) assumes a trusted verifier, but the prover is unknowingly untrusted. For example, an attacker

FIGURE 2.4: Distance Fraud Attack

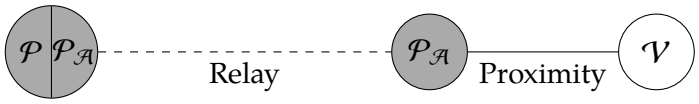


FIGURE 2.5: Terrorist Fraud Attack

may have compromised the prover device. This attack contains elements of both the Mafia Fraud Attack and the Terrorist Fraud Attack and can be considered to be a separate type. This type of attack is discussed in greater detail in Section 2.5.1 (under *'Threats to the User'*).

Relay attacks on Radio-Frequency Identification (RFID) and NFC have been widely studied and demonstrated in the literature [54, 55, 72, 97, 99, 120, 130–132, 159, 161]. In [161] for example, the authors demonstrate that NFC-enabled Android devices can be used as off-the-shelf relay devices, without even requiring firmware and/or hardware modifications.

Since the main focus of this thesis is on contactless transactions like contactless payments, in the following chapters the prover and verifier will be referred to as Transaction Instrument (e.g. the contactless smart card or smartphone that is used for the payment) and Transaction Terminal (e.g. the POS), respectively. In the course of this thesis, the term *'relay attack'* mainly refers to the Mafia Fraud Attack, unless stated otherwise. The attacker is considered to be of opportunistic nature, with access to off-the-shelf equipment and components, and is attempting to successfully relay the communication between two transaction devices without being detected. Even though some of the proposed solutions might be able to protect against other types of relay attacks, this is not the main scope of this thesis (except in Chapter 8).

2.4 Relay Attack Detection

Relay attack detection and prevention requires information regarding the coexistence of the devices involved in the transaction. In the field of smart cards, distance bounding protocols have been proposed as an effective countermeasure. However, distance bounding protocols that are available for smart cards may not work in the field of smartphones. Several alternative methods have been proposed regarding relay attack prevention in the field of smartphones. Many of these methods have demonstrated that using the environmental (ambient) sensors that are present in modern smartphones can provide sufficient information regarding the co-existence of the devices.

2.4.1 Smart Card Distance Bounding

The first distance bounding protocol was proposed by Brands and Chaum in 1993, and was based on timing the response of a challenge bit sent between the communicating parties (timed challenge-response) [20]. In their proposal, the verifier and the prover first generate a random bit-string each. The verifier transmits a random bit to the prover and the prover responds with the corresponding random bit that it has generated. The time required for the round-trip (time required to send the random bit and receive the response from the prover) is measured by the verifier. Moreover, both the verifier and the prover store the received bits. After the completion of the distance bounding protocol, the prover transmits a Message Authentication Code (MAC) or a digital signature of the received and transmitted bit-strings to the verifier. The verifier can detect whether the two communicating devices are within some acceptable distance by checking whether the correct response bits were received within some acceptable time-frame. This technique focuses on Mafia Fraud Attacks, and both the verifier and the prover are considered to be trusted.

The authors also provide in the same work a variation of their protocol for scenarios where the verifier does not trust the prover, therefore they also focus on Distance Fraud Attacks. In their second protocol, the prover first transmits the hash of a random bit-string to the verifier and replies to the verifier's challenge with bits that are produced by XOR-ing each challenge bit with the corresponding bit of the prover's bit-string. Upon completion of the challenge-response process, the prover transmits the bit-string to the verifier. An untrusted prover will not be able to transmit a correct response bit to the verifier before a challenge bit has been received with higher probability than guessing it. With this work, the authors set the theoretical basis for distance bounding protocols.

In 2005, Hancke and Kuhn proposed the first distance bounding protocol specific for RFID devices, based on ultra-wideband pulse communication [57]. As in the work of Brands and Chaum, timing of the round-trip of bits is used in this work as well. Their protocol assumes that the two communicating devices (verifier and prover) are trusted, therefore it focuses on Mafia Fraud Attacks.

In [70], Hancke used improvised wideband pulses to achieve distance bounding for RFID. A prototype implementation was presented and the effectiveness was measured to be 1m against Mafia Fraud Attacks and 11m against Distance Fraud Attacks.

In [126], the authors proposed an RFID distance bounding system that allows the implementation of distance bounding protocols. Practical implementation demonstrated effectiveness against all three types of relay attacks (Terrorist Fraud Attack up to 4.5m, and Mafia and Distance Fraud Attacks up to 0.41m).

In the field of smartphones, distance bounding might not be an effective countermeasure. In [59], the authors implemented distance bounding protocols on Android smartphones. However, they noticed irregularities in the protocol execution time (the authors report this observation in [101]). The exact causes could not be identified by

them, but it is reported that, for example, irregularities existed between when a device had just booted and when it was left running for some time. The authors state that these irregularities could possibly exist due to background processes that are using the device's resources. A similar observation was made by Umar et al. [156], who demonstrated that the round-trip timing of the same APDU, transmitted multiple times using NFC, is not constant on smartphones. The authors also noticed that in the case of smart cards the round-trip time deviation is much shorter. This could also be related to the multi-process nature of smartphones, since the processing power is not evenly distributed. Moreover, different devices use different processors, versions of the operating system, and may contain a different set of installed applications. Depending on these attributes, the round-trip time of a challenge-response in a distance bounding protocol may vary not only among different devices, but also among the same device model, based on the amount of installed applications and the resources required by those. Also, it may vary on a single device, based on the amount of background processes and the tasks that these are performing. For these reasons, distance bounding protocols may not be applicable in the field of smartphones. Sensing the natural ambient environment in order to gather proximity evidence information is an alternative approach that has been proposed by a number of researchers.

2.4.2 Natural Ambient Environment

Using this approach (the natural ambient environment) to prove the co-existence of two transaction devices, first, both transaction devices have to measure some element of the natural ambient environment that is surrounding them, using some ambient sensor (e.g. the temperature sensor) over some period. The captured values are then compared for similarity either by a TTP or by one of the transaction devices. If the similarity between the two measurements is within some certain acceptable threshold, the devices are considered to be within close proximity.

An ambient sensor measures a particular environmental property of its immediate surroundings, such as temperature, light, humidity, and sound; a wealth of such sensors are deployed in modern smartphones and tablets. Figure 2.6 illustrates a generic approach for deploying ambient sensing as a proximity detection mechanism for mobile payments (between a smartphone and a payment terminal), with the following variations:

1. **Independent Reporting.** Both the smartphone and payment terminal collect sensor measurements independently of each other and transmit these to a trusted authority (depicted as solid lines in Figure 2.6). The authority compares the sensor measurements, based on some predefined comparison algorithm with set margins of error (threshold), and decides whether the two devices are within proximity to each another.

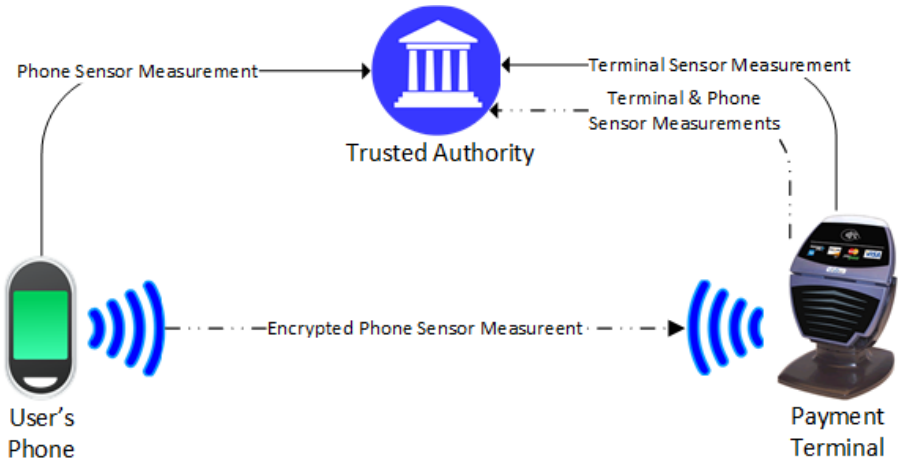


FIGURE 2.6: Generic Deployment of Mobile Sensing for Proximity Detection

2. **Payment Terminal Dependent Reporting.** This set-up involves the smartphone encrypting the sensor measurements with a shared key between smartphone and trusted authority, and transmitting the encrypted message to the payment terminal (shown as double-dot-dash lines in Figure 2.6). The payment terminal sends its own measurements and the smartphone's to the trusted authority for comparison.

3. **Payment Terminal (Localised) Evaluation.** The smartphone transmits its measurement to the payment terminal, which compares it with its own measurements locally; the payment terminal then decides whether the smartphone is in proximity.

It is worth noting that using the natural ambient environment to detect relay attacks does not guarantee the same level of security as distance bounding protocols. The aim of using the natural ambient environment is mainly to protect against the off-the-shelf, opportunistic attacker. This approach, however, mostly concerns transactions performed using mobile devices, on which distance bounding protocols might not be applicable. Moreover, mobile devices are vulnerable to a multitude of other attacks [164], therefore their use in security critical scenarios should be avoided.

Related Works

Bellow are identified and summarised key pieces of related work that have suggested using natural ambient sensing as a PRAD mechanism.

Ma et al. [106] explored the use of Global Positioning System (GPS) location data for determining the proximity of two mobile phones. A ten-second recording window was used in which GPS data was collected every second, which was subsequently compared across various devices. The work reported a high success rate in identifying co-located devices.

Halevi et al. [68] demonstrated the use of ambient sound and light for proximity detection. The authors analysed sensor measurements —collected for 2 and 30 seconds duration for light and audio respectively— using a range of similarity comparison metrics. Extensive experiments were performed in different physical locations with a high success rate in detecting proximate devices.

Varshavsky et al. [158] used the shared radio environment of devices —the presence of WiFi access points and associated signal strengths— as a proximity detection mechanism for secure device pairing. The approach was considered to produce low error rates and, while it did not focus on contactless mobile transactions, their techniques and methodologies may still be applicable.

Urien et al. [157] suggested using ambient temperature with an elliptic curve-based RFID/NFC authentication protocol to determine whether two devices are co-located before creating a secure channel. The proposal combines the timing channels in RFID, used traditionally in distance bounding protocols, in conjunction with ambient temperature. The work, however, was not implemented and has no experimental data to evaluate its effectiveness.

Mehrnezhad et al. [116] proposed the use of an accelerometer to provide proximity assurances of a mobile device with a payment terminal. The scheme requires the user to tap the terminal twice in succession, before comparing the sensor data from the device and the terminal for similarity. It is difficult to deduce the total time it took to complete a transaction entirely, but the authors provide a recording time range of 0.6–1.5 seconds.

Truong et al. [152] evaluated four different sensors using a recording duration of 10–120 seconds. While the results were positive, the long sampling duration renders it unsuitable for various smartphone-based contactless transactions, like contactless payments and transportation ticketing-related transactions.

Additional work by Jin et al. [89] showed that a smartphone's magnetometer can be used to establish proximity assurances. This approach requires more than 500ms; the authors do not claim that magnetometers can provide an effective relay attack detection mechanism.

Shrestha et al. [139] used a number of ambient sensors within specialised hardware, known as *Sensordrone*, for proximity detection. The work was not evaluated using the ambient sensors available on commodity handsets, did not provide the sampling duration, and states that data from each sensor was collected for a few seconds.

As mentioned in Chapter 1, industry-imposed requirements apply on certain applications, like EMV payments and transportation ticketing, that require transactions to complete within a 300–500ms time-frame. Table 2.1 summarises past work, using sensor sampling durations to determine their suitability for NFC-based mobile phone transactions in banking and transportation. *'Unlikely'* proposals have sample durations so large that they may not be adequate for mobile-based services that substitute contactless cards, while those with reasonably short durations are labelled *'More Likely'*. However, even schemes denoted as *'More Likely'* may not be suitable, as

Table 2.1: Related Work in Sensor-based PRAD Mechanisms.

Paper	Sensor(s) Used	Sample Duration	Contactless Suitability
Ma et al. [106]	GPS	10 seconds	Unlikely
Halevi et al. [68]	Audio	30 seconds	Unlikely
	Light	2 seconds	More Likely
Varshavsky et al. [158]	WiFi (Radio Waves)	1 second	More Likely
Urien et al. [157]	Temperature	N/A	–
Mehrnezhad et al. [116]	Accelerometer	0.6 to 1.5 seconds	More Likely
Truong et al. [152]	GPS Raw Data	120 seconds	Unlikely
	WiFi	30 seconds	Unlikely
	Ambient Audio	10 seconds	Unlikely
	Bluetooth	12 seconds	Unlikely
Jin et al. [89]	Magnetometer	4.5 seconds	More Likely
Shrestha et al. [139]	Temperature (T)	Few seconds	Unlikely
	Precision Gas (G)	Few seconds	Unlikely
	Humidity (H)	Few seconds	Unlikely
	Altitude (A)	Few seconds	Unlikely
	HA	Few seconds	Unlikely
	HGA	Few seconds	Unlikely
	THGA	Few seconds	Unlikely

no proposal is evaluated under the time constraints stipulated by the banking and transport sectors. In these sectors, the goal is to serve people as quickly as possible to maximise customer throughput; time is critical in determining whether a transaction is successful and, indeed, permitted. Here, an optimal transaction duration is up to 500ms rather than seconds.

Ambient sensing has also been used in various user-device authentication, key generation and secure channel schemes [94, 136]. These applications typically measure the environment for longer periods of time (>1 second) and, generally speaking, their primary goal is not proximity detection of a device with a terminal. As such, these are omitted from the discussion.

In Chapter 3, the effectiveness of using the natural ambient environment as a PRAD mechanism in short transactions of up to 500ms is investigated. Ambient sensors widely available on Android devices were examined. Using threshold and machine learning-based analyses, it was concluded that the proximity evidence provided by data collected in a 500ms time-frame might not be sufficient for use with applications like EMV payments. Similarly, Haken et al. [66] demonstrated that using sensors available on the Apple iOS platform, insufficient evidence regarding the co-existence of two devices can be collected in 500ms. Finally, in [138] and [137], Shrestha et al., raised concern that many of the proposed solutions might be insecure in the presence of an attacker with context manipulating capabilities.

Relay attacks, however, do not always assume that the communication parties are trusted. A relay attack, for example, has been demonstrated, which is performed through a malicious application installed on a user's device [131]. In this occasion, assuming that a user is genuine and has no interest in performing a relay attack on their device, the challenge for the attacker is to compromise that user's device.

A popular method that attackers may use to distribute malicious functionality is through repackaging genuine applications. This topic is discussed in the following section.

2.5 Application Repackaging

Repackaged applications pose a serious threat to the Android ecosystem and can be used in order to perform various types of attacks, including a variation of relay attacks (discussed in previous sections). In the past, they have been found to be responsible for approximately 86% of the Android malware distribution [171].

In the context of this thesis, repackaged applications are defined as applications that impersonate a genuine application by slight modifications/variations to the genuine application's artwork and/or changes to its source code in a way that the repackaged application looks and/or feels like the genuine application. The main objective of a repackaged application is to mimic a genuine application so it can target novice users that gravitate towards the popularity/functionality of the genuine application. In this definition, applications that infringe the potential intellectual property of the original application and present themselves as unique/different applications, are not included.

2.5.1 Threats to the Android Ecosystem

Major threats posed by application repackaging can be separated into those concerning the application developer and those concerning the user.

Threats to the Developer

Developers are given the opportunity to gain financial profit through their applications. Repackaged applications pose a threat to them in various ways.

- *Unauthorised redistribution:* An attacker can redistribute an application, signed using their own private key, and possibly sell it in an application store, without the original developer's permission.

- *Advertisements:* Potential revenue from an application might be redirected to the attacker, as they might have altered the developer's account with their account, thus receiving the advertisement revenue. More specifically, Android applications can include advertisements that are linked to a user's account, typically the developer's. The linked account is credited based on the advertisement clicks or impressions. An attacker can alter the advertisement package and replace it with theirs. In this way, the profit from users who install the repackaged application will be going to the attacker instead of the original developer [61].

- *Cracking (Piracy):* Application repackaging might distribute pirated copies of a genuine paid application, by bypassing any implemented verification and validation mechanisms.

Threats to the User

Repackaged applications that aim to harm the user may achieve their goal in different ways.

- *Trojan horse:* A repackaged application could act as a Trojan horse. Malicious code can be implanted in the original application, capable of intruding the platform's security and/or the user's privacy. For example, repackaging an application that requires permissions to access the Internet and the device's storage can allow an attacker to steal the user's images.

- *Denial of upgrade:* An Android application should be signed using the same private key as the version that is already installed in order to get updated. A repackaged application will not be updated by a genuine application's update, since the signatures will not match. Therefore, the user will not be able to upgrade once a repackaged application has been installed. Not being able to update an application can pose a threat. An attacker, for example, can take advantage of a known vulnerability in an application by re-signing and distributing the unpatched version of the application. This way, the user will be unable to upgrade to the newer, patched version, without uninstalling the already installed application first.

- *Variation of Mafia/Terrorist Fraud Attack:* An untrusted user (prover) is usually described in the definition of a Terrorist Fraud Attack, who is cooperating with the attacker either willingly, or after being threatened. In the case of a Mafia Fraud Attack, the attacker typically uses some relay equipment, while the user (prover) is considered to be trusted. In the field of smartphones, an attack is also applicable that combines elements from both, in which the user is considered to be legitimate but the device has been compromised by the attacker. While the architecture of the attack, as well as the attacker's access to the user's resources, remain the same as in a traditional Terrorist Fraud Attack, the user is unknowingly cooperating. A malicious application is used in order to retrieve the relevant information from the user's device. Such an attack has been demonstrated in [131]. Repackaged applications can be used for the distribution of such applications.

2.5.2 Related works

AndroGuard was proposed by Desnos and Gueguen [45]. Their proposal is based on Control Flow Graphs, used to measure the similarity between decompiled applications.

Crussell et al. proposed *DNADroid* that uses Program Dependency Graphs to detect similar applications [41]. The authors claim that their system produces a low false positive rate. However, the authors state that advanced obfuscation techniques can go undetected. To increase the system's performance, they only compare the application in question against applications with similar names.

Zhou et al. used Fuzzy Hashing for repackaging detection [170]. This method is based on generating a hash of the application by breaking it down into small chunks and combining their hashes. They also remove string operands that can easily be altered from the instructions prior to hashing, in order to prevent common obfuscation techniques. They have created an application called *DroidMOSS*. They state that although their system is very robust, detection may fail if big chunks of code have been added to the original application.

Another solution, proposed by Hanna et al., introduces the idea of Feature Hashing for the detection of similar applications [75]. The main goal of this approach is to detect pirated applications and applications that use already known malicious code. For this purpose, a tool called *Juxtapp* was created, that according to the authors is resilient to some amount of obfuscation.

Hu et al. [82], proposed MIGDroid, which makes use of method invocation graph-based static analysis as a means for repackaging detection. The aim of their proposals is to detect repackaged applications injected with malware, complementing prior solutions. High effectiveness of the solution was reported, with 1,260 malware-contaminated repackaged applications being detected.

Zhauniarovich et al. [167] proposed detection of repackaged applications based on the contents of the `.apk` file. Their method showed results similar to those of techniques that involve code analysis. SHA1 is used for the comparison of all the files between different applications. A repackaged application with slight changes on many files might go undetected. The authors propose a combination of their method with code analysis in order to overcome this issue.

In [48], the authors statistically extracted data-flow features based on API calls triggered by user interactions with the application's interface. According to the authors, very high detection rate was achieved, comparative to the state-of-the-art at the time of publication. Even though the primary goal of this technique is malware detection rather than repackaging detection, the authors reported some effectiveness in detecting repackaged applications as well.

Other methods have also been developed that do not have as primary goal repackaging detection, but may detect it in many occasions [24, 91, 160, 168]. A framework capable of measuring the obfuscation resilience of algorithms used in repackaging detection programs was also proposed in 2013 by Huang et al. [84]. Other researchers have also proposed methods for detection of repackaged applications as well, or have investigated the subject [40, 135, 166].

In [134], the authors used the Kullback-Leibler Divergence metric to detect repackaged applications. Based on smali opcode [65] (i.e. disassembled Android Java

code) that can be common on both genuine and repackaged applications, a population distribution is built for genuine and potentially repackaged applications. The authors report high detectability of repackaged applications.

Aldini et al. [3] proposed *PICARD*, which is a collaborative approach to repackaging detection, by generating contracts based on the analysis of the behaviour of an application, at runtime. Certain application behaviours are not only categorised in terms of being known or unknown, but also in their likeliness to occur. Their approach is suggested to be complementary to static analysis tools.

In [147], the authors base their detection technique on code heterogeneity analysis. Machine learning is used to classify code regions based on their behavioural features. The authors report a very low false negative rate and a low false positive rate.

Due to the computational complexity of many methods that have been proposed, these methods are only meaningful on an application store level. For this reason, Zhou et al. [169] proposed client side initiated repackaging detection, with *AppInk*. Their method uses application watermarking in order to confirm the authenticity of an application. In order to achieve that, a few additional steps have to be taken by the developer in order to embed the watermark.

Sun et al. [145] proposed detection of visually similar Android applications by comparing Android layout files of genuine and potentially repackaged applications. Two approaches were proposed, a server-based and a client-based. The server-based approach calculates the distance between the tree of the layout files of genuine and potentially repackaged applications. The client-based approach is based on generation of hashes of the layout tree. The authors state that it is not as accurate as the server-based approach, but its aim is to provide protection for users who do not rely on application stores for installing applications on their devices. High detectability is reported, however the authors do not take into consideration layouts that are generated dynamically, for example through Java code.

In order to tackle this issue, Malisa et al. [110] used Graphical User Interface (GUI) crawling techniques to capture screenshots of applications, which were then compared for similarity. High detectability rate of perceptually similar applications was detected in this case as well. Their approach was later evaluated through a large-scale field trial, and they implemented a detection system capable of estimating the deception rate [109].

Chen et al. [26] proposed *MassVet*. Their technique combines repackaging and malware detection by first looking for applications with similar user interface as the suspected application and then looking for differences between the subject and the visually similar applications. The effectiveness was found to be higher than popular anti-malware software and was even capable of detecting potential zero-day malware.

In [100], the authors use text and image similarity detection techniques to detect visually similar applications based on their names, icons, descriptions, and screenshots. Their aim is to detect camouflaged applications, meaning applications that try to imitate genuine applications in terms of style. The authors report high effectiveness of

their solution, which was capable of detecting potentially camouflaged applications on the Google Play Store.

In order to achieve both accuracy and scalability, Chen et al. [25] proposed the use of the centroid of dependency graphs to calculate the similarity between fragments of code of two applications. The authors report high effectiveness in providing both accuracy and scalability.

2.6 Summary

This chapter's main goal is to introduce the reader to concepts and technologies that are discussed in forthcoming chapters. Initially, an overview of the Android OS is provided, with focus on Android HCE and Android applications. The most commonly discussed types of relay attacks are then listed and described. Relay attack detection and prevention methods in the fields of smart cards and smartphones are then discussed. Finally, Android application repackaging is explained and the related literature is given.

Part II

Natural Ambient Environment-Based Relay Attack Detection

Chapter 3

Natural Ambient Sensing for Detecting NFC Relay Attacks

3.1 Introduction

In the previous chapter it was mentioned that NFC has opened smartphone platforms to a range of application domains, particularly those based previously on smart cards. Through card emulation, users may use their smartphones in a range of payment, transport and access control applications — leading to the deployment of services such as Google Pay and Apple Pay. The Android platform also provides HCE, which enables any application to take advantage of card emulation mode via NFC. Deloitte estimated that 5% of the 600–650 million NFC-enabled mobile phones were used at least once a month to make a contactless payment globally in 2015 [31]. In the same year, 12.7% of smartphone users in the USA were actively using contactless mobile payments according to Statista, while the value of such transactions is projected to grow to $114 billion (USD) by the end of 2018 [144]. Similar trends are being observed in other domains where mobiles are used to deliver smart card-type services, like transportation and access control [1].

However, contactless transactions are vulnerable to relay attacks. As discussed in the previous chapter, such attacks on contactless smart cards have been extensively studied in related literature [47, 53, 72, 97] and were quickly shown to be applicable to NFC-enabled smartphones [54, 55, 159]. In a range of past proposals, the sensors found on smartphones have been suggested as a strong countermeasure against relay attacks [68, 106, 139, 152, 157, 158].

In this chapter, an empirical study that evaluates smartphone sensors as a PRAD mechanism under the practical time constraints stipulated by EMV (500ms time-frame) is performed in two stages. Initially, the focus is on proximity detection, by using data from the natural ambient environment. Legitimate data was collected from two devices which were in proximity (<3cm) with each other. Both devices were recording data from some predefined sensor over a period of 500ms. The goal of this part of the study was to attempt, based on the collected data, to establish proximity evidence, so that the co-location of the devices is implied.

During the second stage of the study, the focus was on assessing the effectiveness of the natural ambient environment for relay attack detection. Legitimate data was collected from two devices that were in close proximity (<3cm) to each other, while two devices, placed 1.5m apart, assisted in the collection of illegitimate data.

In both parts, two-fold evaluation was conducted, firstly, based on similarity threshold analysis and, secondly, on machine learning. It is neither evaluated nor claimed that similar results will be produced in other deployment scenarios where distance bounding and transaction time limits are not as stringent. It should be considered that the transaction time limit and operating distance are not set arbitrarily, but rather in compliance with industry-wide requirements, as stipulated by EMV and transport specification bodies.

3.2 Approaches and Evaluation Metrics

In previous work (e.g. [68, 116, 139, 152, 158]), two approaches have been used predominately for sensing-based PRAD mechanisms:

- *Threshold-based Similarity*: the use of time and frequency domain similarity metrics, such as Mean Absolute Error (MAE), Pearson's Correlation Coefficient, and Coherence. A single threshold is generated, that aims to separate all legitimate transactions from illegitimate ones using a particular similarity metric. The transaction is accepted if the metric result falls within this pre-set threshold of the maximum allowed dissimilarity.

- *Machine Learning*: the use of well-known classification algorithms, such as Naïve Bayes, Support Vector Machines (SVMs), and Random Forests. The classifier is trained on a set of feature vectors with corresponding binary labels (legitimate or relayed transaction), which are collected beforehand. The trained model is used to classify subsequent transaction data streams as legitimate or relayed.

Standard binary classification evaluation metrics have been applied to measure the effectiveness of a particular scheme, namely classification accuracy [68], f-scores[2] [139, 152] and EER [116]. F-scores and EERs involve the computation of *false positives/acceptances* (the number of relayed transactions accepted erroneously) and *false negatives/rejections* (the number of legal transactions rejected). F-scores account for *precision*, the correct positive results divided by the number of all positive results, and *recall*, the number of correct positive results as proportion of the number of positive results that should have been identified (see Eq. 3.1). The EER —used extensively in biometrics, e.g. fingerprint recognition [111]— is found by calculating the False Acceptance Rate (FAR) and False Rejection Rate (FRR), shown in Eq. 3.2, over a range of thresholds, and finding the rate at which $FAR = FRR$. Alternatively, some authors have opted to present the FAR and FRR results alone [158]. Finally, accuracy

[2]Also known as the F1-score or F-measure.

represents the correct identification of positive and negative transactions in the test set (Eq. 3.3), but does not clearly illustrate the number of false positives and negatives.

F-scores and accuracy have been used to primarily evaluate machine learning-based relay attack detection, e.g. [139, 152], while EERs have been employed for threshold-based similarity approaches [116] to find an acceptance threshold that, broadly speaking, balances usability (false rejection rate) with security (false acceptance rate). The EER is used as a common evaluation metric for assessing the performance of machine learning and threshold-based approaches across a variety of similarity metrics.

$$F_{score} = \frac{2TP}{2TP + FP + FN} \tag{3.1}$$

$$FAR = \frac{FP}{FP + TN} \qquad FRR = \frac{FN}{FN + TP} \tag{3.2}$$

$$Accuracy = \frac{TP + TN}{P + N} \tag{3.3}$$

3.3 Effectiveness for Proximity Detection

In the first stage of the study, the effectiveness of ambient sensors to determine whether two devices are in proximity to one another (irrespective of whether a relay attack is in action) was evaluated. A field trial was conducted in which sensor data from 1,000 transactions per sensor was collected from 252 users at four different locations on the university campus. Two devices were used for the data collection: a Transaction Terminal (TT), and a Transaction Instrument (TI). Data was collected for 500ms upon the initiation of the NFC-based transaction (i.e. when the two devices were brought within proximity to each other), and stored locally (in an SQLite database) for later evaluation (Figure 3.1). More detailed information is provided in the following sections.

The collected data was subjected to the two analyses discussed in Section 3.2 —threshold-based similarity and machine learning-based— to determine whether data from legitimately co-located devices can be distinguished from non-proximate pairs.

3.3.1 Test-bed Architecture

To empirically evaluate each sensor, an experimentation test-bed was developed. Two applications were implemented and installed on a pair of Android devices: one emulating the transaction terminal TT and the other acting as the transaction instrument TI (a smartphone). When the devices come sufficiently close, an NFC connection is established and both begin recording data using a specified sensor. After collecting measurements for 500ms, each device stores the recorded data in a

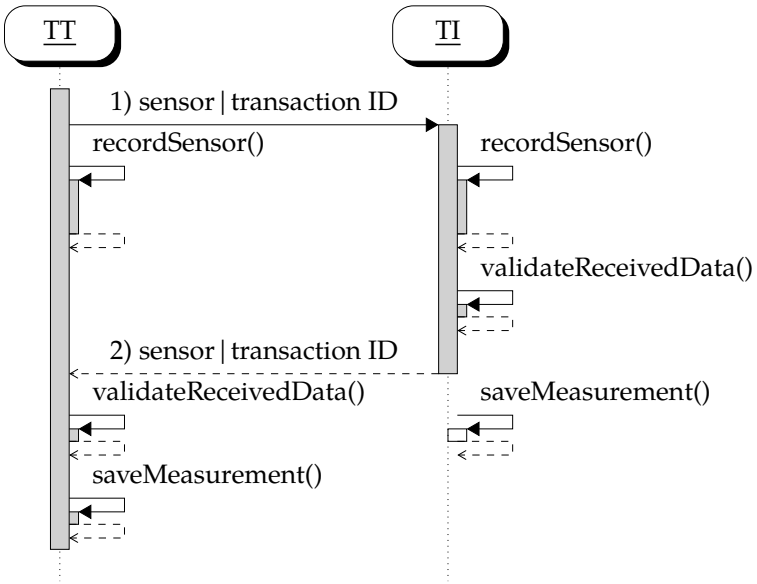


FIGURE 3.1: Measurement Recording Overview (Proximity)

local database. During field trials, one mobile phone was fixed in a position, as a terminal, and the second mobile phone was free of any restriction.

Figure 3.1 shows this in more detail. Bringing the two devices together (< 3cm) causes the TT application to send the first message to TI over NFC, stating which sensor it uses in the transaction and a unique transaction ID. After this message is received by TI, both applications initiate the process to record a sensor for 500ms.

After collecting the measurements, TI validates the data it received from the terminal —whether the transaction ID and chosen sensor match that of the terminal (shown in message one in Figure 3.1)— and returns an acceptance or rejection message accordingly. This validation process ensures that both devices were recording data from the same sensor. Finally, TT performs the same process, ensuring that both devices used the same transaction ID and recorded from the same sensor. The measurement is rejected in the event that devices recorded data for differing transaction IDs or sensors. Upon validation, the devices save the measurements in their local databases. The database is designed to hold measurements for each transaction, which are used in the off-line analysis of each sensor.

Data Collection Framework

Each sensor was tested in four different locations around the university —the lab, cafeteria, dining hall and library— to account for the influence of different physical locations on sensor measurements. A field trial was conducted in each location with 252 participants that carried out a varying number of transactions. Each participant used TI and was given free reign with how they interacted with TT for each transaction, i.e. they could tap it once, hold it extremely close without touching, or tap and hold it to the device. This is to closely replicate the conditions in which they would conduct a regular contactless transaction. The data collection at each of the locations was

collected over an eight hours period (0900–1700hours) with irregular gaps between transactions over the course of four days.

Four devices were used in the experiments, forming two TT–TI pairs. The first pair consisted of two Nexus 9 tablets, while the second pair comprised two Android smartphones: a Nexus 5, assuming the role of the transaction terminal, and a Samsung Galaxy S5 mini (SGS5 mini), which acted as the transaction instrument. The availability of the sensors on each device is listed in Table 3.1.

TABLE 3.1: Sensor Availability

Sensors	Nexus 9 (1)	Nexus 9 (2)	Nexus 5	SGS5 mini
TI–TT Pair: Nexus 9 (1) → Nexus 9 (2)				
Accelerometer	✓	✓	✓	✓
Bluetooth	*	*	*	*
GRV†	✓	✓	*	✓
GPS	*	*	*	*
Gyroscope	✓	✓	✓	✓
Magnetic Field	✓	✓	✓	✓
Network Location	✓	✓	✓	✓
Pressure	✓	✓	✓	✗
Rotation Vector	*	*	*	*
Sound	✓	✓	✓	*
WiFi	*	*	*	*
TI–TT Pair: SGS5 mini → Nexus 5				
Gravity	○	○	✓	✓
Light	*	*	✓	✓
Linear Acceleration	○	○	✓	✓
Proximity	✗	✗	✓	✓
Unsupported				
Relative Humidity	‡	‡	‡	‡
Ambient Temperature	‡	‡	‡	‡

✓: Working properly. ✗: Not present on device. *: Technical limitations.
‡: Evaluated using Samsung Galaxy S4. ○: Returned only zero-values.
† Geomagnetic Rotation Vector.

Initially, 100 transactions were performed for each sensor, in order to assess the quality of the sensor data. Some sensors, such as Bluetooth, GPS, Rotation Vector, and WiFi —although present on the devices— returned no or very few data points within the 500ms time-frame (≥99% sensor failure). Two sensors, for humidity and temperature, are relatively uncommon among Android devices and none of the tested devices contained them. For completeness, two Samsung Galaxy S4 (SGS4) smartphones were used, containing these sensors, to include in the evaluation. However, for these sensors, measurements were recorded for less than 6% of the transactions, in the 500ms time-frame. Consequently, having failed to record any data points in such a large number of cases, these sensors were omitted from subsequent analysis. A minimum of 1,000 transactions were recorded for each sensor — comprising measurement pairs, for which both TT and TI have valid sensor data.

Listing 3.1: Sample XML Record

```
<?xml version=‘‘1.0’’ encoding=‘‘UTF-8’’?>
<measurements>
  ....
  <measurement>
    <id>6</id>
    <timestamp>60</timestamp>
    <data0>-0.32935655</data0>
    <data1>6.2396774</data1>
    <data2>-7.558329</data2>
  </measurement>
  ....
</measurements>
```

The Android OS returns data captured by a sensor in time intervals set by the application. To prevent unnecessary power consumption, however, the OS returns sensor values to the application only when the values have changed from the past measurement. Note that the sound sensor (microphone) captures data in a continuous, uninterrupted stream; in this instance, the applications converted the recorded amplitudes into sound pressure levels (in decibels) before storing the values in their respective databases. For Bluetooth, the OS returns data every time a new Bluetooth device is discovered nearby; with WiFi, this is after the device has scanned and detected the presence of nearby access points. Details regarding each of the sensors is available in Appendix A [8].

The recorded sensor measurements were stored in XML form in each database. A new child element was created, containing the sequence ID of the measurement, the timestamp (initialised to zero at the start of the transaction), along with the data for each returned measurement. A sample XML record can be found in Listing 3.1. The sequence ID of each measurement, the date and time the transaction occurred, the location in which it was captured, and the transaction ID were also stored in the database.

The transaction ID is a random 7-byte string, generated by the terminal, and is used to link the measurements of each device, in order to produce a TT–TI pair. Occasionally, the NFC connection was disrupted, primarily when the devices were moved apart before the transaction was completed. To address this, the transaction ID was used in conjunction with the sensor type to detect and exclude these measurements prior to analysis. After completing the field trials, the databases were extracted from the devices via USB and transferred to the computer running the analysis software.

3.3.2 Analysis Approach

The analysis approach that was followed is described in this section. As already mentioned, both threshold- and machine learning-based analyses were performed.

Threshold-based Analysis

Initially, it was ensured that data with the same transaction ID and sensor used are regarded. Transactions that did not exist on both devices (due to failure of one of the transaction devices to validate the $\{ID|sensor\}$ tuple) were discarded. This occurred in some occasions, as already explained, for example when a user was pulling the devices before the transaction could be completed. In this occasion, device TI could not send message 2 (see Figure 3.1) to device TT, leading the former to not storing the transaction data, due to an incomplete transaction fault. To compare (TT_i, TI_i), the similarity of, or distance between the two was measured. For each sensor, the EER and associated threshold, t, were computed using six time- and frequency-domain similarity measures, including those used in previous work. These are listed forthwith. Each metric was applied directly onto the sensor data collected during the field trials.

- *Time domain metrics*:
  - Mean Absolute Error (MAE), Eq. 3.4
  - Pearson's correlation coefficient [116], Eq. 3.5
  - Maximum cross-correlation [68, 152], Eq. 3.6
  - Euclidean distance [152], Eq. 3.7
- *Frequency domain*: coherence [116], Eq. 3.8.
- *Both domains*: time-frequency distance [68, 152], Eq. 3.9.

$$MAE(A,B) = \frac{1}{N}\sum_{i=1}^{N}|A_i - B_i| \tag{3.4}$$

$$corr(A,B) = \frac{\sum_{i=1}^{N}((A_i-\mu_A)(B_i-\mu_B))}{\sqrt{\sum_{i=1}^{N}(A_i-\mu_A)^2\sum_{i=1}^{N}(B_i-\mu_B)^2}} \tag{3.5}$$

Where μ_A represents the arithmetic mean of A.

$$M_{corr}(A,B) = \max(cross\_correlation(A,B)) \tag{3.6}$$

$$d(A,B) = \sqrt{\sum_{i=1}^{N}(B_i - A_i)^2} \tag{3.7}$$

$$C_{AB}(f) = \frac{|G_{AB}(f)|^2}{G_{AA}(f)\cdot G_{BB}(f)} \qquad F_{AB} = \sum_{f} C_{AB}(f) \tag{3.8}$$

Where G_{AA} is the auto-spectral density of A and G_{AB} is the cross-spectral density of signals A and B (left). The similarity is found by the sum of the magnitudes of coherence values at all frequencies (right).

$$Diff(A,B) = \sqrt{D_{time}(A,B)^2 + D_{freq}(A,B)^2} \tag{3.9}$$

Where $D_{time}(A,B) = 1 - M_{corr}(A,B)$ and $D_{freq}(A,B) = ||FFT(A) - FFT(B)||$, in which $||FFT(A) - FFT(B)||$ is the Euclidean norm of the FFTs of signals A and B.

Certain sensors —the accelerometer, gyroscope, magnetic field, rotation vector, and GRV sensors— produce a vector of values comprising x, y and z components. In these instances, the vector magnitude (Eq. 3.10) was used in order to produce a single, combined value prior to the computation of the metrics.

$$M = \sqrt{x^2 + y^2 + z^2} \tag{3.10}$$

The similarity metrics listed above were computed for each successful transaction. As legitimate pairs (those collected during field trials) were considered (TT_i, TI_i), while illegitimate pairs were considered (TT_i, TI_j). An ideal similarity metric, V, would produce $V(TT_i, TI_i) < t$ (where t is the threshold) and $V(TT_i, TI_j) > t$ for all possible pairs. The illegitimate pairs were constructed by exhaustively matching each TT_i with every TI_j measurement belonging to another transaction ($i \neq j$).

For the analysis, a Python application was developed for threshold-based similarity learning using the Numpy [162], SciPy [90], Matplotlib [85] and Pandas [115] Python packages for metric implementations, graph plotting and CSV data processing.

Machine Learning-based Analysis

For machine learning, the Weka [56] package was employed. Simple distance functions give equal weight to each measurement obtained from the sensors. To address this, machine learning is considered to perform the discrimination between legitimate and illegitimate transactions more effectively, through modelling non-linear interactions between measurements. The conducted experiments apply the same evaluation strategy as the first method, but the threshold is based on the probability estimate output by the learned classification model, i.e. the estimated probability that a transaction is legitimate. Moreover, to avoid optimistic bias in the error estimate when applying machine learning, it is necessary to perform a train-test experiment. More specifically, the full set of transaction pairs is split into a training set and a test set. The machine learning algorithm is applied to the training set to build a classification model that can output class probability estimates. Once the model has been built, it is applied to obtain probability estimates for the test set. Moreover, instead of using a single train-test split, 10-fold cross-validation repeated 10 times is used. This is a standard estimation technique from machine learning that generates 100 different train-test splits based on shuffled versions of the data. The learning algorithm is run 100 times on the 100 training sets, to build 100 models, and these 100 models are

evaluated on the corresponding test sets. Performance estimates from the 100 test sets are averaged to obtain a final performance estimate.

Five different machine learning algorithms that induce different types of classification models from data were used, namely, Random Forest, Naïve Bayes, Decision Tree, Logistic Regression, and SVM. A Random Forest [21] is an ensemble classifier comprising a large number of decision trees induced in a semi-random manner from bootstrap samples of the original dataset. In the experiments, the default parameters for the *RandomForest* classifier in the Weka software were used. It generates an ensemble of 100 decision trees from the training data. The other machine learning algorithms that were evaluated are (a) Simple Naïve Bayes classifier, which assumes conditional independence of the attributes given the classification, and models the data for each class using a Gaussian distribution with a diagonal covariance matrix, (b) Logistic Regression, which assumes that the log-odds of the class probabilities are linearly related to the attributes, (c) Decision Trees grown using the C4.5 decision tree learning algorithm [125], and (d) SVMs optimised with the Sequential Minimal Optimisation (SMO) algorithm [124]. For the SVMs, the complexity parameter C and the width of the Radial Basis Function (RBF) kernel γ were tuned using a grid search by optimising Area Under the Receiver Operating Characteristic (AUROC) as estimated using internal cross-validation on the training data.

It is worth noting that additionally, as part of the analysis in Chapter 7, Weka was benchmarked against the popular Python library `scikit-learn` [133]. Almost identical results were produced by both Weka and `scikit-learn`, when using the same algorithms and configuration parameters.

Pre-processing

In the rare event that the sensor values exceeded the maximum permitted transaction time (500ms), such values were discarded prior to computing the similarity, for both analyses. Moreover, any sensor values on device A (can be either TI or TT) that were recorded after the maximum time recorded by device B (denoting the communicating device of A) were also discarded. This was to prevent undefined results when computing metrics such as MAE; calculating $|A_{i,j} - B_{i,j}|$ is undefined when $A_{i,j}$ has no corresponding datapoint on device B, i.e. $B_{i,j}$.

Related to this was the issue of irregular sampling rates; usually, sensor values of device A did not map to exactly the same times as those recorded on B. That is, $A_{i,3}$ may have been collected at 15ms from the start, while $B_{i,3}$ was recorded at 22ms. This time discrepancy will have some impact on the results, particularly if the number of recorded datapoints is small and the time difference is large. Hence, to mitigate this, linear interpolation was performed on both sets of datapoints to establish a consistent sampling rate of 10ms before computing either similarity metric.

TABLE 3.2: Threshold-based EERs for each sensor with Mean Absolute Error (MAE), Pearson's Correlation Coefficient (PCC), Maximum Cross-Correlation (C-Corr), Euclidean Distance (ED), Coherence (Coh) and Time-Frequency Distance (T-FD). Best result for each sensor shown in bold.

Sensor	MAE	PCC	C-Corr	ED	Coh	T-FD
Accelerometer	**0.434**	0.458	0.501	0.498	0.542	0.501
GRV†	**0.384**	0.486	0.500	0.442	0.524	0.498
Gravity	0.429	**0.424**	0.498	0.501	0.506	0.498
Gyroscope	0.443	**0.441**	0.493	0.498	0.548	0.499
Light	0.488	0.496	0.545	0.502	**0.471**	0.546
Linear Acceleration	0.496	**0.426**	0.494	0.507	0.507	0.500
Magnetic Field	**0.323**	0.384	0.537	0.337	0.568	0.536
Pressure	**0.270**	0.492	0.601	0.283	0.503	0.601
Rotation Vector	0.498	0.466	0.501	0.278	0.500	**0.273**
Sound	0.417	0.488	0.481	**0.338**	0.518	0.481

Proximity excluded due to insufficient unique values.
† GRV: Geomagnetic Rotation Vector sensor.

3.3.3 Results

The results for the threshold-based and machine learning analyses are presented in Tables 3.2 and 3.3 respectively. Note that the proximity sensor was excluded from the analysis. On some Android devices, proximity sensors return the precise distance at which an object is located from the sensor, whereas others return a binary value for whether an object is close to/far from the sensor (within 5cm)[3]. The test-bed devices returned only binary values. Virtually every transaction contained 'far' values, as the devices were tapped back-to-back and the sensor was located on the front of

TABLE 3.3: Estimated EERs for Machine Learning Algorithms (obtained by repeating stratified 10-fold cross-validation 10 times. Best result for each sensor shown in bold.)

Sensor	Random Forest	Naïve Bayes	Logistic Regression	Decision Tree	Support Vector Machine
Accelerometer	0.626 ± 0.024	0.509 ± 0.026	0.526 ± 0.023	0.500 ± 0.0	**0.498** ± 0.025
GRV	**0.435** ± 0.021	0.447 ± 0.024	0.474 ± 0.031	0.500 ± 0.0	0.489 ± 0.036
Gravity	0.874 ± 0.018	0.579 ± 0.020	0.579 ± 0.024	**0.500** ± 0.0	**0.500** ± 0.026
Gyroscope	0.683 ± 0.027	**0.499** ± 0.024	0.543 ± 0.024	0.500 ± 0.0	0.511 ± 0.025
Light	0.576 ± 0.026	0.515 ± 0.024	0.533 ± 0.025	**0.500** ± 0.0	0.508 ± 0.024
Linear Acceleration	0.603 ± 0.025	0.507 ± 0.027	0.543 ± 0.023	**0.500** ± 0.0	**0.500** ± 0.021
Magnetic Field	**0.292** ± 0.021	0.319 ± 0.020	0.322 ± 0.020	0.415 ± 0.015	0.398 ± 0.046
Pressure	0.103 ± 0.010	0.107 ± 0.010	0.287 ± 0.013	**0.092** ± 0.054	0.319 ± 0.045
Proximity	0.499 ± 0.031	0.537 ± 0.069	**0.476** ± 0.188	0.500 ± 0.0	0.543 ± 0.254
Rotation Vector	**0.276** ± 0.046	0.563 ± 0.243	0.596 ± 0.233	0.500 ± 0.0	0.513 ± 0.243
Sound	**0.288** ± 0.019	0.314 ± 0.022	0.310 ± 0.021	0.347 ± 0.136	0.411 ± 0.041

[3] Android Developers (Use the proximity sensor): http://developer.android.com/guide/topics/sensors/sensors_position.html#sensors-pos-prox

the device. Consequently, this returned identical values in almost all cases when applying the similarity metrics described previously, e.g. $MAE = 0$, which impeded threshold-finding. Machine learning was able to capture the rare times in which the sensor returned 'close' values, like when the user covered the device with their hand during the transaction. While the machine learning results are included, the issues identified mean they should be treated with caution. Other technical challenges existed elsewhere; the Rotation Vector sensor, for example, returned significant numbers of zero values on the test-bed devices, which likely distorted the results of the analysis, while sound was capable of capturing values for only half of the permitted 500ms time-frame.

The results indicate that no sensor in either analysis can satisfactorily distinguish between proximate and non-proximate device data pairs. Some sensors provide virtually no discrimination and perform similarly to a random classifier, e.g. the accelerometer (43.4–49.8% EER) and the linear acceleration (42.6–50.0%). Other sensors provide better discrimination, e.g. the magnetic field (29.2–32.3%) and the pressure (9.2–27.0%), but still fall short of acceptable performance. Even in the best case —the pressure sensor using the Decision Tree classifier— the EER was 9.2%. By definition of the EER, this implies that approximately 9.2% of both legitimate and illegitimate transactions would be rejected and accepted respectively. Rejecting almost 1-in-10 legitimate transactions in a high throughput scenario, such as mobile ticketing in a subway system, is likely to cause user annoyance in practice. As such, it is difficult to recommend any single sensor in the analysis as an effective proximity detection method.

3.4 Effectiveness for Relay Attack Detection

The first evaluation focused only on proximity detection, rather than using data from relay attacks. Further field trials were conducted in order to expand the analysis, in which data was collected from two devices that were genuinely in proximity and a third device that was located 1.5m/5ft away. The third device was co-located with a fourth device, which was not collecting data. This replicated a relay attack in which an adversary launches the attack on a nearby victim, such as in a shop queue. The aim was to determine whether sensor data from the illegitimate device pair —the terminal and the device 5ft away— could be distinguished from a genuine (legitimate) pair, i.e. the terminal and the device in proximity.

3.4.1 Test-bed Framework: Theoretical Model

A test-bed environment was designed and developed, that captures both legitimate and illegitimate pairs of sensor measurements under real-world conditions. A genuine pair of measurements comprises two devices that are physically in close proximity to each other (<3cm), while an illegitimate pair is from two devices that are not physically in close proximity, which for the course of this study was considered to

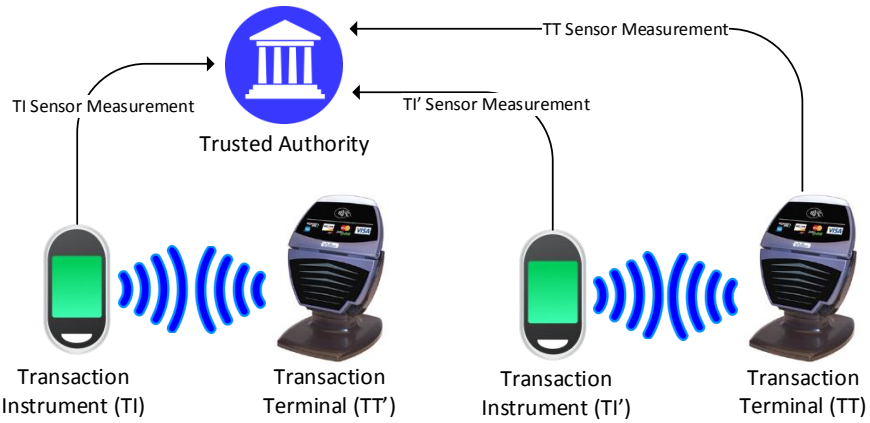


FIGURE 3.2: Overview of Test-bed — Trial Data Collection Platform

be 5ft (1.5m) — emulating a reasonable distance for pickpocketing, for example, at a busy supermarket. To achieve this, a test-bed of four devices was developed, as per Figure 1.1, that records sensor measurements on both the legitimate terminal and device, and an emulated victim phone at distance (all three devices measure the ambient sensor values at approximately the same time). To avoid any discrepancies introduced because of the dependence of ambient sensors on location and time, these pairs were collected concurrently.

Figure 3.2 shows the data collection setup. The Transaction Terminal (TT), as in the previous stage (Section 3.3), is a static device and is used as a reference point for the two pairs. The Transaction Instrument (TI') is a mobile phone in close proximity to TT. The ambient sensor measurement pair TT–TI' is referred to as the genuine pair. Another mobile phone, at a 5ft (1.5m) distance from TT, is referred to as the Transaction Instrument (TI), and is co-located with the Transaction Terminal (TT'). The ambient sensor measurement pair TT–TI is referred to as the illegitimate pair. The rationale for setting up the test environment in this manner is to collect ambient sensor values from proximate and distant devices almost simultaneously. If, for a transaction 'T_i', the genuine pair is uniquely identified (and accepted) then the ambient sensor is considered effective. However, if the illegitimate pair is accepted (whether uniquely or along with the genuine pair) then the relay attack on that 'T_i' is successful. The reasoning is that if two devices at 5ft (1.5m) apart measure ambient sensors independently and the illegitimate pair is indistinguishable from a genuine pair, then the attacker can successfully relay messages between these two devices without being detected.

At a point in time a user taps TI' to TT; at approximately this point, TI is also tapped against TT' and the ambient sensor measurements are initiated. Thus, at approximately the same time, three separate ambient sensor measurements for the three devices are collected (TT' does not record any sensor values). Overall, for an ambient sensor to be effective, a Trusted Authority (e.g. a TTP) responsible for the

PRAD should be able to distinguish the genuine pair from the illegitimate pair. To evaluate each ambient sensor's effectiveness for proximity detection and to detect relay attacks, the collected data were analysed using threshold- and machine learning-based analyses, as in the previous stage (Section 3.3).

3.4.2 Test-bed Architecture and Data Collection Platform

As mentioned in Section 3.4.1, four devices were used in the data collection phase, TT, TI', TT', and TI. During the experimental phase, the devices TT and TI' were placed at a distance of 5ft from the devices TT' and TI. When TI' was brought in close proximity to TT, an NFC connection between the two devices would be established, initiated by TT, indicating the beginning of a transaction. As already mentioned, according to the EMV standard, TT and TI' should be in proximity, less than 3cm apart. During the analysis process, the pair TT–TI' represented the genuine devices, where no relay attack was involved (genuine pair). The pair TT–TI represented the genuine devices, where a relay attack was active (illegitimate pair — the two devices were not in proximity). So, device TI' had a double role, acting as both a genuine, and a relay device. Figure 3.3 depicts the experimental setup.

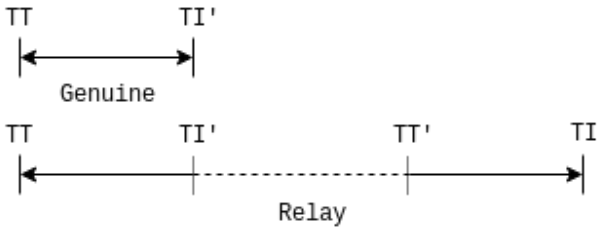


FIGURE 3.3: Test-bed Scenarios

Three SGS4 (GT-I9500) Android devices running Android 5.0.1 were used in the experimental phase for data collection. This specific device was found to include a large variety of sensors, covering the majority of sensors supported by the Android platform [8], excluding the Geomagnetic Rotation Vector. A Nexus 5 Android device was used as TT', which was not collecting sensor data. Table 3.4 lists the available sensors on the SGS4 device, the ones used in the experiments, and the rational behind excluding some. For the majority of excluded sensors, no values could be captured in the 500ms time-frame in the initial tests. Based on initial testing, for less than 5% of the 500ms transactions any data is being returned by the Bluetooth, GPS, Network Location, and WiFi sensors on the SGS4 device. Furthermore, during the initial evaluations of Ambient Temperature and Relative Humidity sensors, it was discovered that insufficient data was returned by these sensors during 500ms for any meaningful proximity analysis. The proximity sensor on the SGS4, like in the case of the devices used in the proximity evaluation stage, returns only a true or false value when the sensor, located on the front of the device, is covered or uncovered. Hence, it could not be used effectively in the experimental phase. Lastly, the sound sensor, i.e. the microphone, of the specific device was tested; it was found that this could

Table 3.4: Sensor Availability for the Samsung Galaxy S4

Sensor	Supported	Used	Reason
Accelerometer	✓	✓	-
Gravity	✓	✓	-
Gyroscope	✓	✓	-
Light	✓	✓	-
Linear Acceleration	✓	✓	-
Magnetic Field	✓	✓	-
Rotation Vector	✓	✓	-
Ambient Temperature	✓	✗	Insufficient values in time-frame.
Bluetooth	✓	✗	Insufficient values in time-frame.
GPS	✓	✗	Insufficient values in time-frame.
Relative Humidity	✓	✗	Insufficient values in time-frame.
Network Location	✓	✗	Insufficient values in time-frame.
Pressure	✓	✗	Insufficient for data collection.
Proximity	✓	✗	Insufficient values in time-frame.
Sound	✓	✗	Insufficient values in time-frame.
WiFi	✓	✗	Insufficient values in time-frame.
GRV†	✗	✗	Not present. Used Rotation Vector instead.

†Geomagnetic Rotation Vector

not initiate and record values within 500ms. Finally, after the initial analysis, seven sensors were selected, and the others were discarded.

An overview of the measurement recording process across the four devices is presented in Figure 3.4. Three Android applications were developed. Devices TI′ and TI were running the same application, and HCE was used to achieve NFC communication with TT and TT′ respectively. The application for device TT included two connection interfaces. For the first interface —used for the NFC connection with the device TI′— device TT was set to NFC reader mode, allowing it to interact with discovered NFC tags. For the second interface —used for connection with the device TT′ over WiFi— device TT would broadcast a UDP packet in the local network. In both, the message transmitted on the NFC or wireless channel included the sensor to

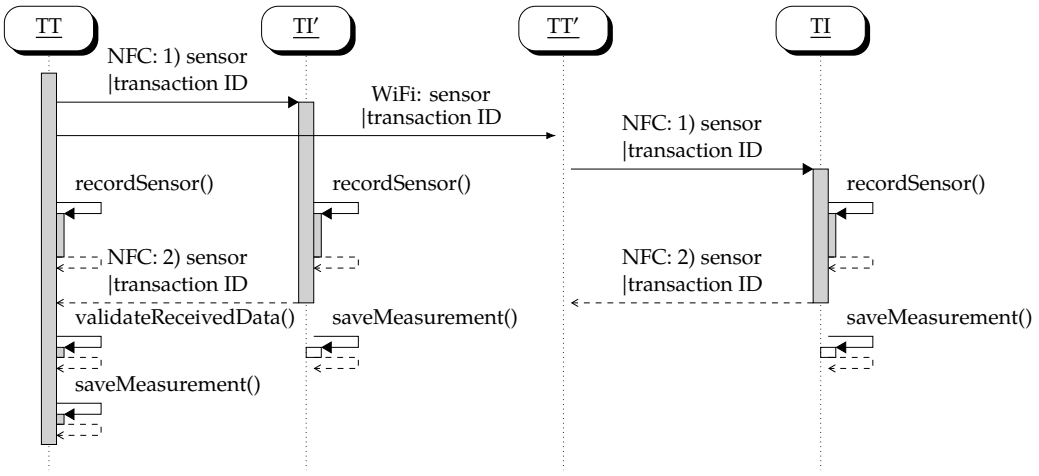


Figure 3.4: Measurement Recording Overview (Relay Attack)

be measured in that transaction and a random 7-byte transaction ID. The transaction ID was generated by device TT and used in the analysis phase in order to uniquely identify each transaction across the devices. In real-world scenarios, device TI' would act as the device communicating with TT'. However, since the scope of this chapter is to evaluate the effectiveness of the ambient environment as an anti-relay mechanism, device TT was responsible for sending the information across the WiFi channel for greater transaction synchronisation.

Lastly, the application running on device TT' featured a broadcast listener for UDP packets from TT. Devices TT and TT' were connected to the same wireless hotspot, created for the requirements of the experiment. Upon receiving a packet from device TT, device TT' would be able to initiate a transaction with device TI, upon tapping the latter to the former. After the initiation of a transaction, devices TT, TI', and TI would start recording data using some predefined sensor for 500ms. As already mentioned, on the Android operating system, data captured by a sensor is returned to an application in time intervals set by the application. The rate at which data was polled from the sensors was set at the highest available, which was the same across all three devices.

Following the recording time period, devices TI' and TI would send a response message, containing the transaction ID and sensor used to TT and TT', respectively. Device TT would then validate the received data. In case of inconsistencies, data would not be stored for the specific transaction. The three devices would store the recorded data in a local SQLite database, along with the transaction ID, sequence number, timestamp and the pre-defined location in which the recording took place. Separate database tables were used for each sensor. The recorded data was stored in XML format (as shown in Listing 3.1). During the data analysis phase, only transactions that existed in all three databases, based on their transaction ID and the ambient sensor that was used, were considered. The rest of the transactions were disregarded. A total of 400 transactions were collected for each sensor, distributed in 2 distinct locations on the university campus.

3.4.3 Transaction Data Analysis

In line with the framework discussed previously, a two-fold evaluation was conducted, using the sensor data collected during field trials.

Analysis Approach and Results

Once more, the EERs were computed to determine optimal similarities/thresholds where the rate of false acceptances (FAR) is equal to the rate of false rejections (FRR) for each tested sensor. The following notions of true accepts (TAs) and rejects (TRs), false accepts (FAs) and rejects (FRs) are defined for each evaluation (threshold- and machine learning-based):

Evaluation 1 (threshold-based). TA: the legitimate transaction instrument–terminal pair (i.e. TI′–TT) is accepted correctly. TR: the distant instrument–terminal pair (i.e. TI–TT), is rejected correctly. FA: the distant instrument–terminal pair (i.e. TI–TT) is wrongly accepted as a legitimate transaction. FR: the legitimate transaction instrument–terminal pair (i.e. TI′–TT) is rejected incorrectly.

Evaluation 2 (machine learning-based). For the machine learning experiments, the same definitions as in Evaluation 1 are used. As already indicated above, the estimated probability of being legitimate is used instead of a similarity score when the threshold for EERs is determined. When training and testing each machine learning model, the individual differences $|A_{i,j} - B_{i,j}|$ are used as attributes (also called features or independent variables) that describe each pair of transactions. Each example for training and testing the machine learning model thus has 49 numeric features (corresponding to 490ms sampled at 10ms intervals). An example is labelled as positive if it corresponds to a legitimate transaction and as negative otherwise.

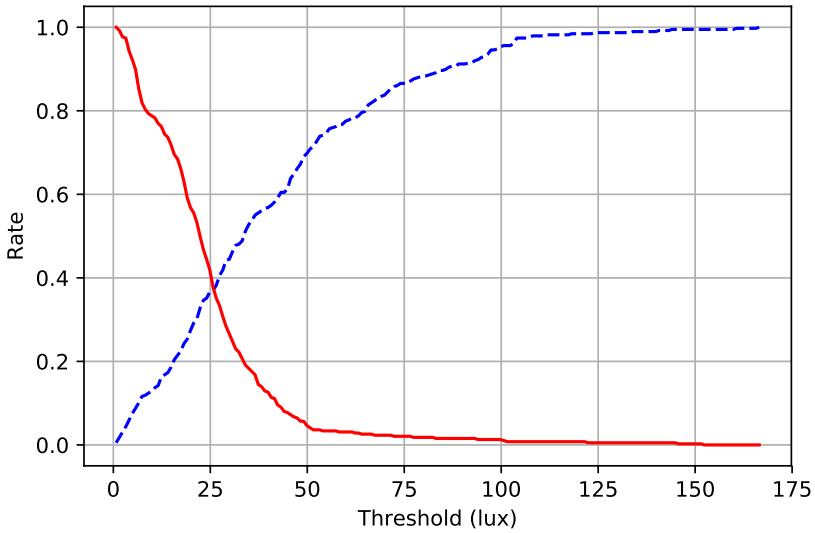


Figure 3.5: Light Sensor FAR-FRR Curves With MAE

In this occasion as well, a threshold is the maximum permitted difference in similarity before a transaction is rejected; conversely, a transaction is accepted if the similarity between sensor measurements is within this threshold. Ideally, a chosen threshold should reject all illegitimate transactions, namely those between the distant instrument (TI) and the terminal (TT), while accepting all legitimate transactions between the legitimate transaction instrument (TI′) and terminal (TT). The EERs and associated thresholds for each sensor are calculated for all of the similarity metrics described in Section 3.3.2. Figure 3.5 shows the FAR/FRR graph produced for 100 different thresholds using the Light sensor with MAE. By inspection, the point at which FRR and FAR intersect (FAR=FRR) has a corresponding EER of approximately 37% for this sensor and metric.

TABLE 3.5: Threshold-based EERs (using the metric abbreviations in Table 3.2)

Sensor	MAE	PCC	C-Corr	ED	Coh	T-FD
Accelerometer	0.494	0.477	0.590	**0.468**	0.507	0.590
Gyroscope	0.521	**0.455**	0.535	0.495	0.528	0.489
Magnetic Field	0.444	0.473	0.470	**0.433**	0.487	0.470
Rotation Vector	0.330	0.472	**0.327**	0.670	0.534	0.509
Gravity	0.521	0.490	0.401	**0.289**	0.503	0.362
Light	**0.367**	0.488	0.444	0.372	0.505	0.437
Linear Acceleration	0.482	0.536	0.503	0.506	**0.443**	0.493

In this part of the study, the sensor selection was limited to the best performing sensors from the first analysis (Section 3.3), i.e. those which successfully and consistently captured values within the 500ms time limit over 1,000 transactions, which were also present on the SGS4 devices that were used during the experiments. Additionally, based on the initial investigations, sensors that are largely uncommon on commodity handsets in the current market, like the pressure sensor, were eliminated. The same analysis techniques were used, as described in Sections 3.2 and 3.3 for proximity detection —threshold-based analysis with the same similarity measures, and the same machine learning algorithms— using the EER evaluation metric. In the machine learning-based analysis, when the data is split into training and test sets, it is ensured that two transactions with the same ID (i.e. a legitimate and a fraudulent transaction that were recorded simultaneously) are either both in the training set or both in the test set, to avoid potential bias. The results for the threshold-based and machine learning analyses of this study can be found in Tables 3.5 and 3.6 respectively.

Similar to the previous part of the study (Section 3.3), some sensors provided poor discriminatory power; the magnetic field sensor, for instance, gave EERs of 36.1% and 43.3% in the analyses. Some sensors provided greater discriminatory power, e.g. the gyroscope (17.9% EER with Random Forest) and the rotation vector sensor (27.7%, also with Random Forest). These EERs, however, are still too high to recommend as an effective PRAD in high throughput situations. Based on this evaluation, the tentative conclusion was reached, that sensing the transaction devices' natural ambient environment may not be a suitable PRAD mechanism under industry-specified time constraints.

3.5 Analysis Equipment

The analysis of both parts of the study (proximity detection and relay attack detection) was conducted on a Fedora machine with a quad-core Intel i5–4690k (3.7 GHz) and 16GB of RAM. The analysis application was developed in Python, using the Pandas and NumPy libraries for data loading and numerical computation. The Weka machine learning software was used to run the machine learning experiments on a cluster

Table 3.6: Estimated EER for Machine Learning Algorithms (obtained by repeating 10-fold cross-validation 10 times)

Sensor	Random Forest	Naïve Bayes	Decision Tree	Logistic Regression	Support Vector Machine
Accelerometer	**0.277**±0.052	0.474±0.047	0.358±0.059	0.483±0.050	0.454±0.126
Gyroscope	**0.179**±0.041	0.354±0.059	0.228±0.049	0.356±0.055	0.288±0.045
Magnetic Field	**0.361**±0.055	0.400±0.053	0.389±0.063	0.421±0.061	0.385±0.053
Rotation Vector	**0.285**±0.052	0.327±0.055	0.317±0.073	0.353±0.050	0.325±0.050
Gravity	0.499±0.046	0.488±0.043	0.494±0.057	**0.484**±0.043	0.486±0.156
Light	0.361±0.059	0.369±0.058	**0.293**±0.149	0.407±0.054	0.351±0.054
Linear Acceleration	**0.307**±0.050	0.484±0.048	0.392±0.057	0.502±0.049	0.397±0.058

of 10 Ubuntu Linux computers with Intel Core i7–2600 CPUs and 16 GB of RAM. Multi-threading was used for training random forests and performing parameter optimisation for support vector machines. Generating all performance estimates using 10-times 10-fold cross-validation took less than two hours. The vast majority of this time was used to obtain performance estimates for the support vector machines.

3.6 PRAD Evaluation Outcome

The presented results provide a basis for quantifying the effectiveness of various mobile sensors as a PRAD mechanism in NFC-based mobile transactions. In the first part, the effectiveness of natural ambient sensing as a device proximity indicator was assessed, while in the second the focus was on relay attack detection. Threshold- and machine learning-based evaluation was performed for both parts.

For the first part (proximity detection), using threshold-based analysis, the Pressure sensor performs relatively better with an EER of 0.27, while the Rotation Vector has a 27.3% possibility of a success — one of the highest in the analysis. Similarly, for the second part (relay attack detection), again using threshold-based analysis, the Gravity outperformed the rest of the sensors, with an EER of 0.289. Not only does the EER imply the potential success of the attack, but also the potential of a legitimate transaction being denied. The Gravity EER indicates that almost 29% of relay attacks would be accepted *and* 29% legitimate transactions would be denied. Denying more than 1-in-4 legitimate transactions would invariably cause annoyance issues for users in practice. From this analysis, it is difficult to recommend any of the sensors for a single-sensor deployment for high security applications, such as banking, using the similarity detection techniques that were used in this chapter. Such sensors might be appropriate for low-security access control, but it is recommended that a thorough analysis of the sensors and their performance in the chosen domain is performed prior to deployment.

The next evaluation applied machine learning classifiers. For the first part (proximity detection), the Pressure sensor with Decision Tree performs significantly

better, with an estimated EER of 0.092. This illustrates that the Pressure sensor may have potential as a proximity detection mechanism, but still the detection accuracy (more than 9% of legitimate transactions would be rejected) is not high enough to provide sufficient security in a real-world deployment. In the second part (relay attack detection), Gyroscope presented the best results, with an EER of 0.179. Again, it is difficult to recommend using this or any other of the evaluated sensors for relay attack detection, with any of the threshold- and machine learning-based algorithms that were used in this chapter.

One reason that past work achieved different results could be due to the significantly larger sampling durations (see Section 2.4.2). The sampling duration imposed in the experiments was in line with the performance requirements of an EMV application, i.e. 500 milliseconds. Transportation is another major application for contactless smart cards; in this domain, the recommended transaction duration is far lower, between 300–500 milliseconds. The 500 millisecond limit in the experiment was thus an upper bound of the recommendations of two significant application areas where contactless mobile phones may be utilised.

3.7 Summary

Proximity and Relay Attack Detection (PRAD) is an important element for many contactless and wireless technologies. In this chapter, it has been illustrated that the viability of PRAD mechanisms can be largely dependent on the time constraints mandated by industry requirements. Contactless payment transactions for example —whether smart card- or smartphone-based— must adhere to <3cm for proximity and up to 500ms for transaction duration, as stipulated by the EMV specifications. The claim that natural ambient environments can provide a robust PRAD, as stated by some previous literature, was evaluated under industry-specified time constraints. This was evaluated for both proximity detection (Section 3.3) and as a relay attack detection mechanism using a test-bed that reflected an actual attack (Section 3.4). The results of the two-part evaluation are presented using six similarity metrics, used previously and several widely-used machine learning classifiers. The investigation aimed to analyse and evaluate a range of sensors present in modern off-the-shelf mobile devices, and to determine which sensors, if any, would be suitable as an anti-relay mechanism for NFC-based smartphone transactions. A total of 17 sensors accessible through the Google Android platform were shortlisted, before limiting the investigation to those which are widely-available and displayed promise in the initial trails. In existing literature, only 13 sensors have been suggested as an effective proximity detection mechanisms, as listed in Table 2.1. Some sensors are only available in specialised ambient sensor hardware and, in almost all instances, no relay attack data was collected to determine their effectiveness against such attacks. In the first part of the study (proximity detection), 10 sensors were successfully evaluated through analysis of collected data, representing a genuine transaction and a malicious

transaction. In the second part (relay attack detection), 7 sensors were successfully evaluated based on data collected from an actual relay attack emulation. In many cases the results were far from what was claimed in past literature; the initial results indicate that natural ambient environments might provide a poor PRAD for time-critical domains such as banking and transport. As such, it is strongly recommend that any PRAD proposal should be evaluated based on the operating restrictions of the suggested deployment application.

Part III

Artificial Ambient Environment-Based Relay Attack Detection

Chapter 4

Preventing Relay Attacks Using Infrared Light

4.1 Introduction

As discussed in the previous chapters, the use of the natural ambient environment as a means of PRAD may not be sufficient for various time-constrained (up to 500ms) contactless transactions. Protecting such transactions against relay attacks, however, is crucial, since time-constraints exist in monetary transactions (e.g. EMV and transport-related transactions), as well as in access control. Failing to protect against relay attacks may have consequences such as money loss and unauthorised access to buildings and facilities.

In this chapter, a framework for generating and measuring an Artificial Ambient Environment (AAE) as a means of PRAD is initially proposed. According to the framework, both transaction devices are responsible for generating (based on the device's peripherals) and/or measuring the AAE. Comparison of the captured values should provide adequate proximity information to the devices for the AAE to be effective, in case these are genuinely in proximity to each other. At the same time, it should be hard for an attacker to reproduce/relay the AAE at a distant location.

The use of infrared light as an AAE actuator is also investigated. This approach requires one of the transaction devices to emit a random sequence over an infrared blaster (emitter) and the other device to capture this sequence. The transaction devices only emit or capture the infrared signals for certain time after the initiation of the transaction, such that an attacker would have to relay the sequence at a remote location in near real-time, otherwise bits would be lost, leading to the detection of the attack. Experimental evaluation demonstrated high effectiveness of using infrared as an AAE actuator for PRAD. Using six distinct relay attack approaches (based on off-the-shelf equipment), relaying the sequence in a timely manner could not be achieved.

Initially, the generic framework for the generation of an AAE is discussed (Section 4.2). The use of infrared light as an AAE actuator is then investigated in Sections 4.3 and 4.4. The effectiveness of the proposed PRAD mechanism, using infrared light as an AAE actuator, is also evaluated in the same sections through the six distinct test-beds that were deployed.

4.2 Artificial Ambience as a means of PRAD

In this section, the theoretical foundation of the AAE framework is described and analysed.

4.2.1 Proposed Framework

As already discussed, ambient sensors in smartphones measure a particular environmental or physical property of the immediate surrounding of the smartphone, like the light intensity, the ambient sound, or the device's acceleration. Existing literature, listed in Chapter 2, argues that two devices, measuring some attribute of their surrounding environment over a period of time through the use of some ambient sensor(s), can provide proof of proximity.

The majority of previous work however does not comply with the EMV transaction requirement to complete in 500ms [51, 112, 146, 149]. In Chapter 3, it was concluded that data captured in up to 500ms by any ambient sensor commonly found on off-the-shelf smartphones may not provide sufficient proof of proximity information to be used in critical applications, like EMV contactless payment transactions.

As an alternative approach, the generation of AAEs by using the smartphone's peripherals, like the speaker or the phone's vibration, is proposed. The aim is to increase the irreproducibility of the ambient environment, in order to thwart the relay attacks and establish strong proximity proof.

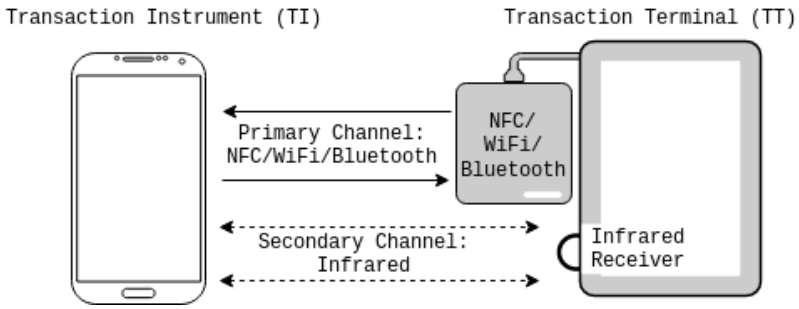


FIGURE 4.1: Framework Architecture

The AAE works as follows. During an NFC (or other — e.g. WiFi, Bluetooth, etc.) transaction, one (unidirectional) or both (bidirectional) devices involved in the transaction are responsible for generating (be actuators) and/or measure (be sensors) the AAE, forming a second communication channel (Figure 4.1 depicts an example of using infrared as an AAE actuator). This second channel, referred to as Artificial Ambience Channel (AAC), should be operating in parallel to the primary channel (i.e. NFC, WiFi, Bluetooth, etc.) for some predefined time.

Smartphones available on the market are not the same in performance capabilities, OS, and/or available ambient sensors. Depending on a device's capabilities, the two devices involved in a transaction can negotiate the tuple $\{\mathcal{A}, \mathcal{S}, \mathcal{T}_{Sent}, \mathcal{M}\}$ to be used in the transaction. $\mathcal{A}$ denotes the AAE actuator to be used in the transaction. $\mathcal{S}$

denotes the sensor which should measure the AAE. $\mathcal{T}_{Sent}$ (where $\mathcal{T}_{Sent} \leq 500$ms in the case of EMV contactless payments, for example), denotes the time for which the AAE should be active. Finally, $\mathcal{M}$ denotes whether the AAE generation should be unidirectional or bidirectional.

Upon the completion of the AAE measurement by the devices involved in the transaction, a comparison between the measured and/or the transmitted data should take place either by one or both devices, or by a TTP. Transmission of the captured measurement should be done in a secure manner, for example integrated in the transaction protocol running on the NFC channel (for on-device verification), or through a secure network channel (for verification by third party). Such an integration is beyond the scope of this chapter.

During the comparison phase, only data exchanged through the AAC while the AAE was active should be regarded ($\mathcal{T}_{Received}$, where $\mathcal{T}_{Received}$ is equal to $\mathcal{T}_{Sent}$ plus, potentially, some acceptable threshold). Thus, during a relay attack, AAE information should have to be successfully relayed by the attacker within the $\mathcal{T}_{Received}$ time-frame, or they will be discarded from the comparison phase.

For an AAE generation method to be successful, the result of the comparison should provide adequate proof of proximity. Therefore, the AAE should be difficult to be accurately replicated by an attacker during a relay attack.

In this chapter, only the properties of the AAC channel are investigated. This is to empirically establish whether AAE can provide an effective proof of proximity as a relay attack countermeasure. To investigate the proposed solution, test-beds were developed using off-the-shelf and custom-built hardware.

Architecture Requirements

In order for a generated AAE to be suitable for relay attack detection, the following requirements should be met:

- The generation of the AAE should be based on some random streams/sequences.
- The AAE generation and the establishment of proximity evidence should be easy for a genuine pair.
- The generated AAE should be hard to relay to/reproduce at a remote location, without detection.

Candidate Architecture

Peripherals widely available on smartphones that can potentially be used to generate an AAE are:

1. Infrared,
2. Camera's flash light,

3. Vibration,
4. Speaker (sound),
5. Bluetooth,
6. Device's display, and
7. Camera

In this chapter, the focus is on infrared emitters, that are widely available on different smartphones[4], mainly for allowing users to control home equipment, like televisions [6].

4.3 Infrared as an AAE Actuator

Infrared is a form of electromagnetic radiation with a wavelength outside of the visible spectrum for the human eye. Infrared emitters exist on a wide range of modern smartphones and their main purpose is for controlling home equipment (e.g. televisions). During the evaluations, Android handsets were used, as Android provides an API for emitting infrared signals [6].

In this section, the architecture and deployment of the test-bed for the evaluation of infrared as an AAE actuator and its effectiveness against six different relay attack techniques are described.

4.3.1 Test-bed Architecture — Infrared

Figure 4.2 depicts the architecture of the test-bed for the evaluation of infrared as an AAE actuator. Two scenarios were regarded. First, a transaction between the transaction instrument (smartphone) TI, and the transaction terminal TT' (referred to as TI–TT'). This scenario aims to evaluate the proximity detection capabilities of the framework, and no relay is involved between the pair. Second, a transaction between the mobile device TI, and the transaction terminal TT (referred to as TI–TT). In this scenario, the pair TT'–TI' is attempting to relay the infrared-based AAE. Figure 4.3 shows a more clear representation of the devices involved in the two scenarios.

Proximity Scenario

In the first scenario, upon the initiation of the transaction, TI begins emitting a sequence consisting of 500 random pre-generated bits through its infrared emitter. The transaction terminal, TT', captures the emitted sequence. After the completion of the transaction, the emitted and captured bits are compared for similarity by a TTP (during the course of the experiments, a laptop running the comparison software acted as the TTP). The comparison step can also potentially be performed by one of

[4]Available devices with infrared emitters: http://www.akshatblog.com/list-of-android-smartphones-with-ir-blaster/

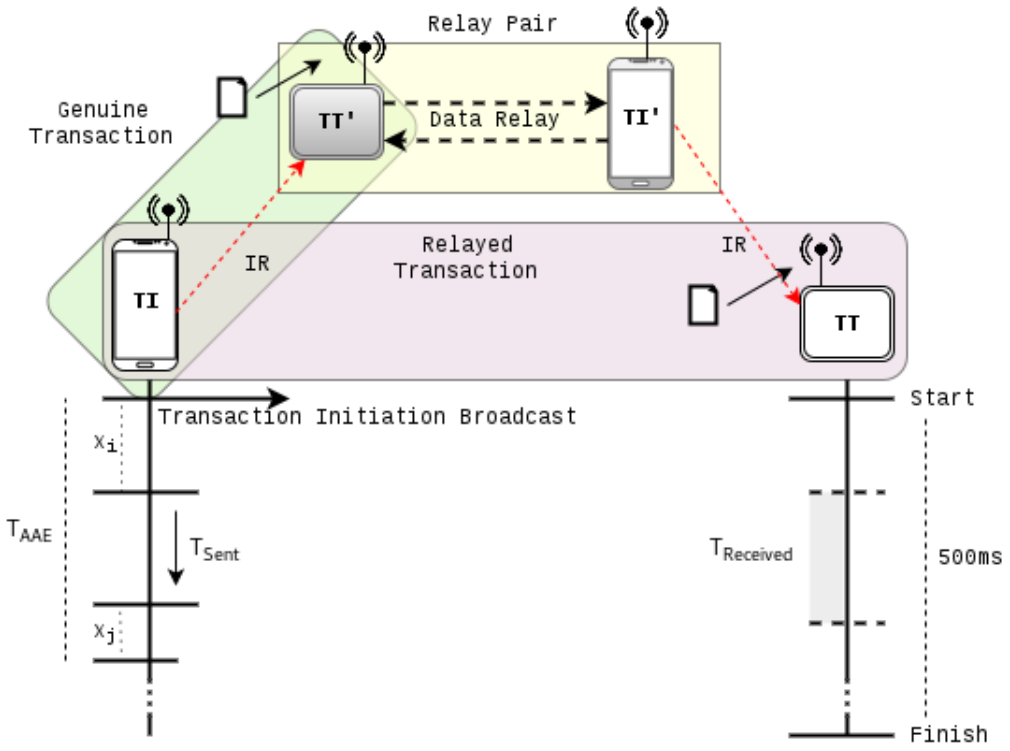


FIGURE 4.2: The Evaluation Framework

the devices involved in the transaction, as already mentioned in Section 4.2, but this is out of the scope of this chapter (further investigation in Chapter 5).

The infrared emitter is technically a Light-Emitting Diode (LED) that emits electromagnetic radiation at the infrared spectrum. In order to emit the 500 bits through it, the bits first have to be converted to pulses and pauses. After experimentation, one bit was set to be represented by a $200\mu s$ pulse (emission) or pause (no emission). Although infrared at 38.4kHz frequency (used during the experiments) can theoretically emit one bit per $13\mu s$, the available mobile devices used in the experiments could effectively emit one bit in no less than $200\mu s$.

Sub-sequences of *ones* and *zeros* from the random 500-bit sequence are encoded into *pulse* and *pause* durations (e.g. a sub-sequence of two consecutive *ones* would be encoded to a $400\mu s$ pulse, a sub-sequence of three consecutive *zeros*, to a pause of $600\mu s$, and so on). The 500-bit sequence requires 100ms of emission through

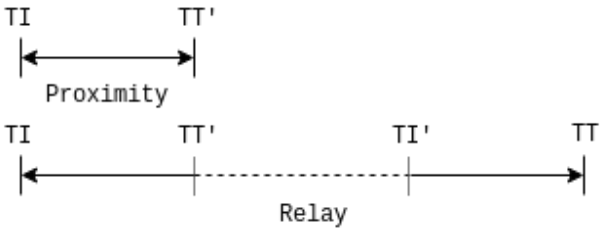


FIGURE 4.3: Test-bed Scenarios of the Infrared-based AAE PRAD Evaluation Framework

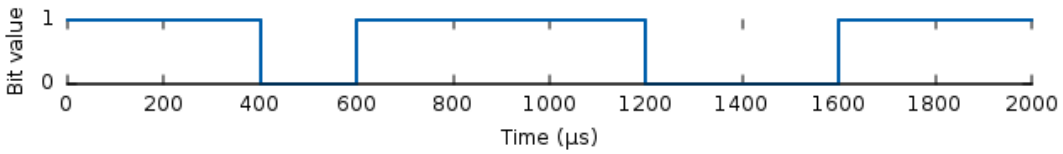


FIGURE 4.4: Time Representation of Bit-Sequence '1101110011'

the infrared emitter ($\mathcal{T}_{Sent}$ = 100ms). Figure 4.4 represents the conversion of the bit-sequence '1101110011' into pulse and pause timings.

An infrared receiving diode capable of sensing infrared radiation is attached to the transaction terminal and accessed through software in order to time the pulse and pause sequences. The terminal listens for infrared signals for the whole period of the transaction. Intercepted pulse and pause timings are then decoded into a bit sequence. In case the data comparison should take place on one of the devices involved in the transaction, the infrared listening period can be reduced according to the requirements of the desired transaction protocol. However, integrating the proposed framework in a particular protocol is beyond the scope of this chapter, and is investigated in the next chapter.

As mentioned in Section 4.2, some time may be required before an AAE actuator can be initialised. In the case of the Android devices that were used in the experiments, it was concluded through analysis of experimental results that time x_i was required before a device could start emitting infrared, and time x_j was required for the process to shut down, after the emission had stopped. Since low level access through API calls is not possible (it is not provided by the Android infrared API), these delays could not accurately be calculated. Modifying the Android OS was also not an option, as after careful inspection of the OS's source code it is believed that both delays are caused by the proprietary infrared drivers of the devices. Because of the infrared Android API, it was only possible to know the total time of the infrared emission process (referred to as $\mathcal{T}_{AAE}$). However, after analysis of 150 runs of the infrared emission process, x_j was calculated to be equal to 28 ± 2ms on an SGS5 mini device (SM-G800F, running Android 5.1.1). Therefore, since $\mathcal{T}_{AAE}$ can be measured by an application, and $\mathcal{T}_{Sent}$ and x_j are known, Eq. 4.1 provides x_i.

$$x_i = \mathcal{T}_{AAE} - \mathcal{T}_{Sent} - x_j \tag{4.1}$$

More specifically, in order to measure x_j, the infrared emitting smartphone was connected through a USB cable to the infrared receiving computer. Immediately after the infrared emission process, the smartphone would log an event message, which was received by the computer through the USB interface. Through a shell script that was written, the computer would calculate the time passed between receiving the last infrared bit through an infrared receiver that was attached to it, and receiving the log event from the smartphone. This way, a single clock was used (that of the computer), so the time could accurately be calculated (no clock synchronisation between the

two devices —emitting smartphone and receiving computer— was required). Even though x_i could also be estimated in a similar fashion, it was found to be more variable between transactions, possibly due to background processes and system load (similar issue to NFC-based distance bounding on smartphones, as described in [101]).

During the comparison phase, any data received by the transaction terminal prior to time x_i (compared to the initiation of the transaction) or after time $x_i + \mathcal{T}_{Sent}$ should be discarded (however some extra threshold might be required in some occasions). This time-frame is referred to as $\mathcal{T}_{Received}$. This aids towards the avoidance of relay attacks through replaying chunks (an attacker having captured the whole sequence, or large chunks, emitted by the legitimate mobile device and replaying it towards a remote transaction terminal). Taking into account the 4ms window (equivalent to 10 bits), introduced by x_j:

$$x_i - 2ms \leq \mathcal{T}_{Received} \leq x_i + \mathcal{T}_{Sent} + 2ms \tag{4.2}$$

The runtime of $\mathcal{T}_{Received}$ is thus $\mathcal{T}_{Sent}$ + 4ms. The 4ms window can potentially be eliminated by enhancements in the Android API and the device drivers. As mentioned earlier, this is not possible without access to the driver source code. Figure 4.5 depicts more clearly the basic concept of the process on the two channels.

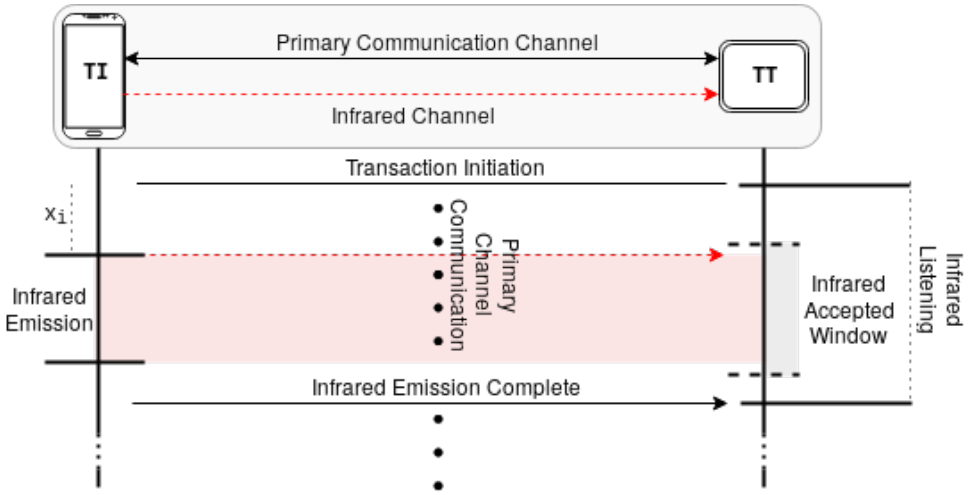


FIGURE 4.5: Infrared as an AAE Actuator

Relay Scenario

In the second scenario shown in Figure 4.3, the same principles apply. The difference is that in this case the malicious pair TT'–TI' is introduced. The malicious terminal TT' is capturing infrared signals emitted by the genuine mobile TI, forwards them to TI', which re-emits them towards TT. The challenge for the attacker is to be able to correctly relay bits in no more than 200μs (the length of one bit) from the time of receiving them. Delay in emission of a few bits will lead to introduction of new bits, since pauses or pulses will last longer. Also caching for longer than the 4ms window

explained above will cast the attack detectable, since not enough bits will be captured within the time frame $\mathcal{T}_{Received}$. Caching however may cause an overhead, since it requires data to be pushed in and popped from a buffer, that may render it inefficient for such short time-frame. Moreover, as already mentioned, the 4ms time-frame may be reduced by enhancing the Android API and device drivers, making the attack even harder for the attacker.

4.3.2 Data Collection

During the experiments, four devices were used. Device TT' was acting as both a legitimate and a relay terminal. Thus, at the same time it was saving the captured infrared signals and relaying them towards device TI' (except in Test-bed 6, described in Section 4.3.3, where it was only capturing data and not relaying). A more thorough comparison was possible this way, by juxtaposing data captured by TT' and TT against the same bit-sequence, emitted by device TI during a single transaction. On both devices, captured infrared pulse and pause timings (in μs) were saved in different log files for each transaction, along with the time (in μs) at which the first bit was received, compared to the initiation of the transaction. Device TI was appending the random bit stream, along with time $\mathcal{T}_{AAE}$, in a `.csv` file. Calculation of x_i (as per Eq. 4.1), and subsequently of the bounds of $\mathcal{T}_{Received}$ were thus possible. These `.csv` files were used during the evaluation phase, described in Section 4.4. After the completion of a transaction, device TI would generate a new random-bit sequence, to be used in the next transaction.

All the devices involved in the experiments were connected to a single, dedicated WiFi network. The indication of the transaction initiation was performed through a network broadcast, transmitted by TI. Assessment of the primary channel (e.g. NFC) is out of the scope of this chapter. However, in certain scenarios, like in an NFC initiated relay, further tapping on both TT' and TT simultaneously might be required in order to ensure the maximum efficiency of the attacker. Such task may be hard to accomplish, even in a controlled environment. Initiating the transaction through a network broadcast satisfies this requirement.

In the next section, different test-bed variant deployments that were used to evaluate the proposal are discussed.

4.3.3 Deployed Test-beds with Relay Attack Variants

In order to assess the effectiveness of the framework against relay attacks, six different variants of the test-bed were deployed, based on potential relay attack configurations.

In all experiments, an SGS5 mini (SM-G800F, running Android 5.1.1) was used as device TI. Raspberry Pi 3 computers (4×1.2GHz CPU and 1GB RAM), running Raspbian Jessie were used as devices TT' and TT. The Raspberry Pi (RPi) was a good candidate for acting as a terminal device, as it is equipped with 40 General Purpose Input Output (GPIO) pins, making it possible to easily build prototypes. In the case

of the device TT', except from listening for infrared signals, it was also relaying them towards device TI' (except in Test-bed 6 described below, where no data was being relayed between the devices TT' and TI').

Test-bed 1: Mobile Based

In the first scenario, a mobile device (SGS4 — GT-I9505, running Android 5.0.1) was used as device TI'. Timings of detected infrared pulses and pauses were transmitted from TT' to TI' through the WiFi channel. In the case of a pulse, TI' would use the `ConsumerIrManager` class (Android infrared API) [6]. In the case of a pause, TI' would wait for the indicated time before emitting the next available pulse.

Test-bed 2: RPi Based (no caching)

Using a mobile device for infrared emission was proven to be inefficient, due to severe delays at emitting any data via the infrared emitter (further discussion in Section 4.4). A custom-built RPi-based relay device was used as device TI' in this scenario instead. An infrared emitting LED was connected to the GPIO pins of the RPi, and controlled by a program written in the C language (for performance reasons), using the GPIO access C library *WiringPi* [79]. Device TT' was used for relaying infrared data to TI' through the wireless network. Unlike in the previous scenario, whenever an infrared state alteration was perceived (switch from pulse to pause, or the opposite), TT' would send a UDP packet to TI'. The application built for TI' would listen for UDP packets from TT', and upon packet arrival it would switch the state of the attached infrared LED, attempting to replicate the sequence emitted by TI.

The same libraries and programming tools as in this scenario were also used in test-beds 3 and 4.

Test-bed 3: RPi Based (caching — TI' side)

Because there are inconsistencies in the packet delivery time through the network, using the method described in Test-bed 2, extra bits might be introduced or subtracted from the infrared sequence emitted by TI', depending on the delay of subsequent packets. In order to reduce the number of packets that have to be transmitted, therefore reducing the number of potential added/removed bits, an attacker can cache data and relay in chunks. As mentioned in Section 4.3.1, due to x_j, an attacker has a window of approximately 4ms, during which data can be cached.

In this scenario, TT' would transmit pulse and pause timings upon alteration of the perceived infrared state. Upon receiving by TI', the pulse–pause timings would be pushed in a stack (array), and 4ms after the arrival of the first packet TI' would initiate emitting them. Using a timer, TI' then re-emits the received infrared sequence by altering the state of the infrared emitter as per the timings received by TT'.

Delays in the network may cause inconsistencies in the delivery time between two packets. In order to minimise the amount of inconsistencies in the bit stream,

delays were handled. In case the next packet arrives while the data received by the previous packet is still being emitted, its data was cached by TI' and emitted after the completion of the current instructions. In case the next packet is delayed, the state of the infrared would not change until new instructions arrived.

Test-bed 4: RPi Based (caching — TT' side)

In this scenario, TT' cached infrared state alterations for 4ms before forwarding to TI'. Although this scenario is similar to the previous, the amount of generated network traffic, which was concluded to be a bottleneck to the relay process, was reduced, since data packets were relayed in 4ms buckets instead of per infrared state alteration. The cached information forwarded to TI' indicate the pulse–pause sequence timings that should be emitted by TI', in μs. The same method as in the previous scenario was used for emitting the infrared sequences, caching data that arrived early and altering the state of the infrared LED when the stack is empty.

Test-bed 5: Infrared Extender Based

Wireless remote control extenders are available on the market. These are devices that come in pairs and convert infrared signals to radio-waves (receiver side) and back to infrared (emitter side). The typical use of such devices is to control infrared remote controlled home equipment from a distance (e.g. from another room). This is achieved by repeating or relaying infrared sequences emitted by the remote control.

Marmitek Powermid XL, a relaying wireless remote control extender, was used during the evaluation phase. According to the manufacturer, the range of the product is up to 100m (free field) or up to 25m (through walls). After inspection of the product's circuit, the receiving device appears to be applying current to an antenna through an infrared receiving diode. Therefore, the circuit remains open when no infrared is detected, and it closes, making it emit at 433.39MHz frequency, when pulses at the infrared spectrum are detected. On the other side, when the receiving antenna captures a signal at 433.39MHz, a circuit is closed, powering three infrared emitters. The product was successfully tested for controlling home appliances (desktop speakers and television) from a distance.

Test-bed 6: Random Generation Based

In the last test-bed scenario, a different random sequence was generated and emitted by a second device. The same device as in Test-bed 1 was used as TI', which was listening for broadcasts regarding transaction initiation as well. Upon receiving a broadcast, it would initiate emission of a pre-generated infrared sequence, in a similar way as TI.

Although in the course of this thesis it is assumed that Android's random number generator is secure, this scenario was assessed mainly for reference reasons.

4.4 Results and Evaluation

The evaluation methodology, the results, and the discussion of the results are presented in this section.

4.4.1 Data Analysis Methodology

The evaluation was conducted in two phases. During the first phase of evaluation, 50 transactions were performed in the lab for each of the test-beds. After the completion of each experimental phase, data from the RPis and the mobile device used as TI was transferred to a laptop (2×2.7GHz Intel i7 CPU and 8GB of RAM, running Fedora 24 OS) for analysis. The transfer was accomplished through SSH and USB cable, respectively. A program, written in Java, would then convert the captured pulse–pause sequence timings, recorded by TT' and TT, of each transaction into streams of bits. Only bits captured within time $\mathcal{T}_{Received}$ were regarded. Translation of pulse–pause timings into bits was accomplished by inverting the methodology described in Section 4.3.1. For this, time $\mathcal{T}_{AAE}$ of each transaction was also given as an argument to the program. The `dwdiff` [69] program was used for the comparison of the bit streams. This tool uses the *GNU diff* utility in order to detect the longest common sub-sequences between two lines (in this scenario each bit stream corresponded to one line), and based on that, calculates the similarity percentage.

Prior to selecting `dwdiff` as the comparison method, 72 well known algorithms that are effective for binary similarity were examined (listed in [28, 30, 60, 76, 77, 86, 141, 148, 154]). A Python program was developed, capable of running the Binary Similarity Calculator found in [118], and storing the corresponding results for later evaluation. After exploring the effectiveness of the algorithms with 50 measurements from each test-bed, most of the algorithms reported high accuracy in detecting the similarity. Tables 4.1 and 4.2 list the output of the various algorithms for a single measurement (in this example, the measurement is taken from Test-bed 5). Many of the algorithms do not return the percentage similarity, but rather some 'similarity weight'. For this reason, their results might need to be rescaled, if one of these is used. This can be achieved by using methods similar to the one proposed in [148]. In the tables, the similarity between the devices that are in proximity is first listed, and then between the distance devices (where a relay attack is taking place). Finally, in the last column, for reference reasons, since many algorithms do not return the percentage similarity, the comparison of the emitted infrared sequence against itself is reported (i.e. the result when there is a perfect match between two 500-bit sequences). Since the performance of various algorithms was high (genuine and relayed transactions could effectively be distinguished), the `dwdiff` was used for convenience and performance reasons in this chapter. However, this does not imply that this is the only, or the best method.

Table 4.1: Performance of Various Binary Similarity Algorithms

Algorithm	**TI–TT′**	**TI–TT**	**TI–TI**
Ample	—	1.03	—
Anderberg 3	0.48	0.00	0.48
Austin-Colwell	0.94	0.52	1.00
Baroni-Urbani-Buser 1	0.99	0.57	1.00
Baroni-Urbani-Buser 2	0.98	0.15	1.00
Braun-Blanquet	0.98	0.53	1.00
Cohen	0.98	0.03	1.00
Cole 1	1.00	0.02	1.00
Cole 2	0.97	0.09	1.00
Consonni-Todeschini 1	1.00	0.90	1.00
Consonni-Todeschini 2	0.74	0.12	1.00
Consonni-Todeschini 3	0.89	0.87	0.90
Consonni-Todeschini 4	1.00	0.88	1.00
Consonni-Todeschini 5	1.00	0.02	1.00
Cosine, Driver-Kroeber, Otsuka, Ochiai 1	0.99	0.68	1.00
Dennis	10.65	0.23	10.73
Dice, Sorensen, Gleason	0.99	0.66	1.00
Dice, Wallace, Post-Snijders 1	0.98	0.87	1.00
Dice, Wallace, Post-Snijders 2	1.00	0.53	1.00
Dispersion	0.25	0.01	0.25
dwdiff	0.99	0.61	1.00
Eyraud	3.94	0.22	4.01
Fager-McGowan 1	-129.01	-212.32	-129.00
Fager-McGowan 2	-1.01	-99.82	1.00
Fager-McGowan 3	0.96	0.65	0.97
Faith	0.75	0.49	0.76
Forbes 1, Margalef	1.92	1.02	1.92
Forbes 2	1.00	0.09	1.00
Fossum	490.39	227.52	498.08
Gilbert-Wells	0.65	0.02	0.65
Goodman-Kruskal	0.98	-0.50	1.00
Gower	0.01	0.01	0.01
Hamann	0.98	0.06	1.00
Harris-Lahey	492.05	125.27	500.00
Hawkins-Dotson	0.98	0.31	1.00
Inner Product	496.00	264.00	500.00

table continues in Table 4.2

TABLE 4.2: Performance of Various Binary Similarity Algorithms *(continuation)*

Algorithm	**TI–TT′**	**TI–TT**	**TI–TI**
Intersection	256.00	225.00	260.00
Jaccard 2W, Czekanowski, Nei-Li	0.99	0.66	1.00
Jaccard 3W	0.99	0.74	1.00
Jaccard, Tanimoto	0.98	0.49	1.00
Johnson	1.98	1.39	2.00
Kulczynski 1	64.00	0.95	—
Kulczynski 2	0.99	0.70	1.00
Maxwell-Pilliner	0.98	0.04	1.00
McConaughey	0.98	0.39	1.00
Michael	1.00	0.06	1.00
Mountford	0.50	0.01	—
Pearson 1, Chi-squared	484.24	0.77	500.00
Pearson 2	0.70	0.04	0.71
Pearson 3	0.04	0.01	0.04
Pearson-Heron 1	0.98	0.04	1.00
Pearson-Heron 2	1.00	0.09	1.00
Peirce	1.00	0.50	—
Peirce 1	0.98	0.03	1.00
Peirce 2	0.98	0.06	1.00
Rogers-Tanimoto	0.98	0.36	1.00
Russell-Rao	0.51	0.45	0.52
Savage	0.02	0.47	0.00
Scott	0.98	-0.10	1.00
Simpson	1.00	0.87	1.00
Sokal-Michener, Simple Matching	0.99	0.53	1.00
Sokal-Sneath 1, Anderberg 1, AntiDice	0.97	0.32	1.00
Sokal-Sneath 2, Gower-Legendre	1.00	0.69	1.00
Sokal-Sneath 3	124.00	1.12	—
Sokal-Sneath 4(3), Rogot-Goldberg, Anderberg 2	0.99	0.52	1.00
Sokal-Sneath 5(4), Gower 2, Ochiai 2	0.98	0.20	1.00
Sorgenfrei	0.98	0.46	1.00
Stiles	9.57	6.35	9.59
Tarantula	—	1.03	—
Tarwid	0.32	0.01	0.32
Yule 1	1.00	0.11	1.00
Yule 2	1.00	0.06	1.00
van der Maarel	0.98	0.31	1.00

Table 4.3: Similarity Percentage Between Sent and Received Bits

	Test-bed 1	Test-bed 2	Test-bed 3	Test-bed 4	Test-bed 5	Test-bed 6
Genuine Pair	98.86%	98.78%	98.34%	98.20%	98.28%	98.32%
Relay Pair	3.70%	44.72%	22.30%	30.76%	81.94%	69.56%

In the second evaluation phase, an additional 450 transactions were performed, using the best performing test-bed (Test-bed 5). The same evaluation procedure was followed.

4.4.2 Evaluation of the Results

Regarding the genuine transaction pair (TI–TT'), the similarity between the emitted and the received bit streams was in all scenarios, except in nine cases, either 98% (in 17% of all transactions) or 99% (in 81% of all transactions). In these nine cases, a portion of the data was captured by TT' in the $\mathcal{T}_{Received}$ time-frame. The reason for the delay of TI in emitting the data is not clear, but it may be related to the extensive use of the device during the experiments. However, some amount of failure is acceptable by the smart card industry [127].

For the evaluation of the proposed solution, a threshold of 98% similarity between the captured and emitted streams was set, based on the evaluation results of the genuine pair. According to the experimental results, the false negative rate would therefore be less than 2%.

The average similarity percentages for the genuine pair (TI–TT') and the relay pair (TI–TT) for 50 runs for each test-bed are presented in Table 4.3. Test-bed 5 is an exception, as the average occurs from analysis of 500 runs. Figure 4.6 presents the similarity percentage of fifty individual transactions, for all test-beds (randomly chosen in the case of Test-bed 5).

The true negative rate, while maintaining the threshold at 98%, was 100%. None of the assessed relay methodologies were capable of exceeding this threshold during the experiments.

In Test-bed 1, the mobile device was incapable of relaying a significant amount of data in the given time-frame. In many occasions no data was relayed. Similar to the device TI, the relay mobile device required time x_i before it could initiate the infrared emission.

In Test-bed 2, due to inconsistencies in the network packet transmission time, extra bits were frequently introduced or deducted from the bit stream. For example, for two subsequent infrared state alterations (i.e. the infrared emitter switching on while it was off, or the opposite), in case of a delay in the transmission of the second network packet, comparatively to the arrival of the first packet, extra bits would be introduced until the arrival of that packet. Similarly, if the transmission delay of a third packet was shorter than that of the second, bits would be deduced in the time-frame between the second and third packets.

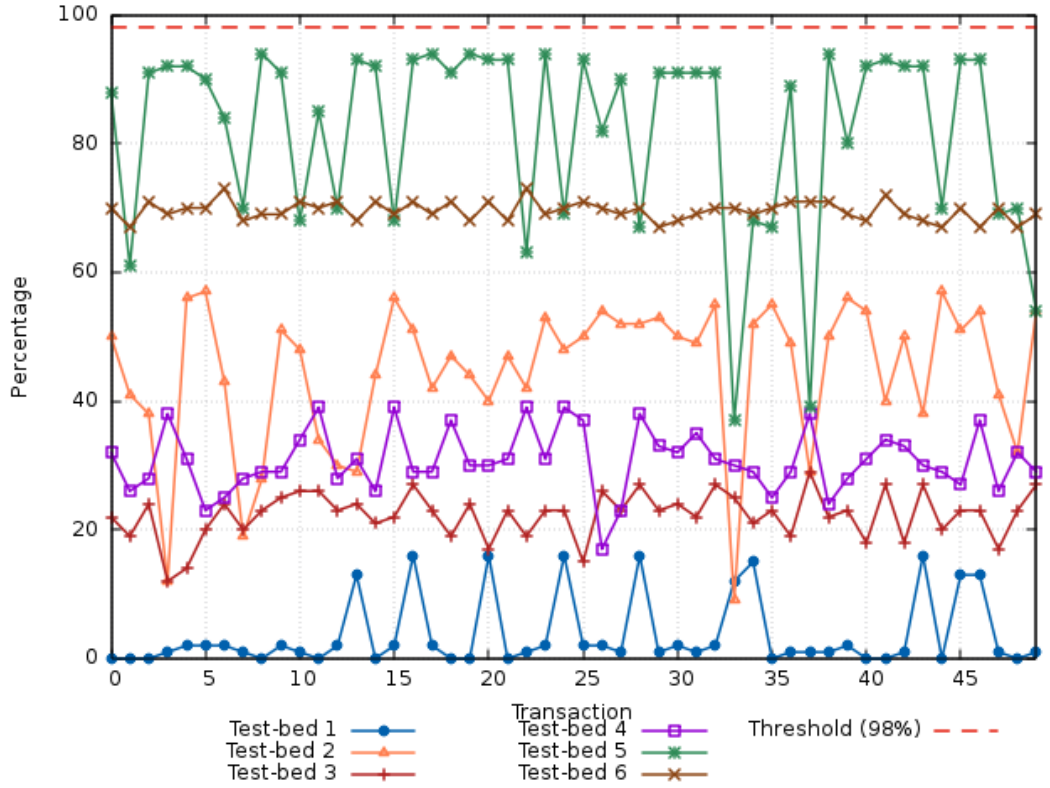


FIGURE 4.6: Test-bed Performance

Higher result consistency was noticed in Test-beds 3 and 4, compared to Test-bed 2, although the overall performance was lower. In these scenarios, caching was causing an overhead. Moreover, in contrast to Test-bed 2, in these scenarios pause to pulse switching (and the opposite) were based on relayed signal timings instead of a single network packet. The overhead imposed due to the previous caused detectable delays in relaying bits in the magnitude of microseconds. Finally, the 4ms caching window proved to be insufficient for caching, relaying and replaying signals. Test-bed 4 performed better than Test-bed 3, but although less than half the network packets were required in comparison to Test-bed 3, Test-bed 4 had to cache data on both sides of the relay, which led to suboptimal increase in the overall performance.

The highest performance was observed in Test-bed 5. Almost 52% of the captured relayed bit streams exceeded 90% similarity upon comparison with the emitted bit stream, almost 10% of which reached the maximum observed similarity of 94%. However, white noise interference captured by the antenna of the infrared extender was causing significant bit irregularities in certain relayed streams, leading to inconsistencies in the reproducibility of the results. Even though this technique produced a relatively high similarity, all the relayed transactions were detected, since the threshold of 98% was not reached in any case.

Finally, Test-bed 6 was the second best performing test-bed, even though in this test-bed, instead of signal relaying, a different random sequence than the one produced by TI was emitted by TI'. In the design of a random number generator

the major concern is the unpredictability of pseudo-random sequences and not the percentage similarity. In any pseudo-random sequence there is a chance that two random numbers have high similarity, but not the same sequence.

Other methods of attacking this system might exist, like relay via fibre optic cable or utilising the infrared range (for few metres), but due to their difficulty in concealment and distance limitations, as well as the high expense associated with them, they were not investigated any further.

4.4.3 Applicability and Feasibility in Real-World Scenarios

In order to integrate this solution in a real-world scenario, extra hardware would be required on the transaction terminal side. An infrared receiver would have to be installed and covered to avoid interference from remote devices. The transaction instrument would also require to be equipped with an infrared emitter. As already mentioned, many smartphones come with such emitters, and Android provides an API for accessing them.

As long as the smartphone's emitter is directed towards the terminal's infrared receiver (within a 30 degree angle), and is not blocked by a phone case or other barriers, the user is not required to perform any additional steps. The lack of such emitters on many devices might affect the feasibility of deploying this solution in a large scale, while it might be more relevant to scenarios where devices are provided to the users (e.g. a company providing devices to its employees). Potential implications that may arise due to the extra positioning requirements (for example on people with motor difficulties) should also be considered before deployment. Moreover, some overhead will be added to the total transaction time. However, as will be discussed in Chapter 5, the proposed solution can effectively be used to establish proximity evidence in various 500 and 300ms transaction scenarios, even when the comparison of the collected data is performed by the transaction devices.

4.5 Summary

Previous works have claimed that sensing the ambient environment and comparing the measurements can be used as a proximity detection method between two devices. However, the effectiveness of this technique in certain types of transactions that require to be completed in short time-frames ($\leq$ 500ms), like banking and transport related transactions, has been argued.

In this chapter, the generation of AAEs as a means of PRAD in smartphone-based contactless transactions with short timing restrictions is proposed. Out of a list of AAEs that was provided in Section 4.2.1, the proposed solution was evaluated by using infrared as an AAE actuator. Six distinct ways of attacking the system were designed, built, and evaluated. The experimental results showed that infrared has a high success rate for relay attack detection, as well as for proximity detection. None of the attack methods were capable of evading the proposed framework.

Chapter 5

Infrared Light for Proximity Critical Scenarios

5.1 Introduction

In the previous chapter, the use of infrared as an AAE actuator was discussed. In this chapter, the applicability of using infrared light as an AAE actuator in real-life scenarios in which relay attacks are relevant, is investigated. Moreover, the architectural enhancements, limitations, and requirements regarding the deployment in a large scale are discussed. The areas that have been investigated are (a) contactless transactions (mobile payments, physical access control, and transportation ticketing) (Section 5.2) and (b) continuous Two-Factor Authentication (2FA) for host-based services authentication (Section 5.3). Further possible scenarios are discussed in Section 5.4.

Protocols that follow industry standards (where applicable), for the integration of the proximity detection technique, are proposed and undergone mechanical formal verification, using Scyther. The performance of each of the aforementioned cases has also been measured through practical implementation.

5.2 Use Case: NFC-based Contactless Transactions

In this section, the potential of using infrared as an AAE actuator to provide PRAD in popular contactless NFC transactions (payments, transportation, access control, and others with similar time restrictions and requirements) is investigated. Modern NFC-enabled smartphones are capable of performing smart card-like contactless transactions. Domains that benefit from this type of transactions include contactless payments, physical access control, and transport ticketing. This way of performing transactions is convenient for the users, but due to security implications, restrictions are usually imposed by the operators. For example, contactless payments are currently limited to £30 per transaction in the UK [46, 155].

These limitations apply in contactless transactions in general, including smart cards. However, smartphones are susceptible to more security threats. They may not contain secure storage, their operating system is much larger than that of a smart card,

therefore more prone to security flaws due to larger amount of coding errors [114], and users have control over the software installed on a device.

Among the possible attacks [164] on contactless transactions is the relay attack. As already mentioned, in order to protect against relay attacks, the coexistence of the devices involved in a transaction should be able to be proven. In the previous chapter, the use of infrared as an AAE actuator was described and assessed. The aim of this chapter is to investigate whether the detection can take place during the course of the transaction.

5.2.1 Threat Model

In a contactless transaction scenario, an attacker performing a relay attack can use a legitimate user's identity without authorisation. Unauthorised access to services that the legitimate user is entitled to can therefore be granted. For example, an attacker can perform a payment using the legitimate user's account, or get unauthorised access to buildings.

For the attack to take place, a relay pair should be operated by an attacker (Figure 1.1). A relay transaction instrument TI', presented as a legitimate transaction instrument, should be tapped to a legitimate transaction terminal (TT) and initiate a transaction. Simultaneously, the attacker should tap the relay device TT' to a legitimate payment instrument (TI). This can be achieved either by masquerading TT' and presenting it to the legitimate user as a legitimate transaction terminal, or by tapping it without the user's consent.

For a successful attack, the communication messages should be relayed between the devices TI and TT, through the relay pair. As already mentioned, according to the EMV standards, a transaction should be completed within a time frame of 500ms. Similarly, in the case of transport related transactions, a time restriction of 300–500ms is usually imposed [150]. Therefore, the relay equipment should be capable of relaying all the communication data between the legitimate device pair (TT–TI) in a timely manner.

In the course of this chapter, the attacker requires no prior interaction or knowledge of neither TT nor TI; the attacker is of opportunistic nature and has only control over the devices TT' and TI'. However, the potential implications if TT or TI are compromised are out of the scope of this chapter, since a relay attack may not be necessary to achieve the same goals under these circumstances. The focus is primarily on the issue of genuine devices requiring proximity assurance in order to conduct a legitimate transaction. Finally, the attacker only has access to off-the-shelf relay equipment.

5.2.2 Candidate Integration Architecture

The anti-relay mechanism should be optimised in order to detect relay attacks during the time-frame of the transaction, as imposed by industry standards. In order for the

proposed solution to be introduced to the existing infrastructure, an infrared receiver should be available on payment terminals. Moreover, payment instruments should be equipped with infrared blasters (emitters).

Since infrared light has a range of a few metres, a denial of service attack can be achieved (for example, an attacker can emit random infrared signals from a distance). Even though this is not a relay attack, the objective of the adversary may still be attained. A protective cover around the infrared sensor can effectively prevent such attacks, by blocking infrared light out of the range required for the payment process from reaching the infrared sensor.

In order for the bit-sequence comparison to take place, three pieces of information are required:

1. The random-bit sequence generated and emitted by TI.
2. The time required for TI to emit the information through infrared (x_i — see Section 4.3.1).
3. The infrared sequence captured by TT.

The second element is required because a delay (initialisation time) was noticed before device TI could start emitting infrared. As mentioned in Section 4.3, after analysis of Android's source code, it was concluded that it is likely for that latency to be induced due to the infrared driver of the device. Since the source code of the driver is not publicly available, this delay is being considered, as it was not possible to reduce it (further discussion in Section 5.2.5). However, since the total emission time (500 bits — 100ms, since each bit is represented by a $200\mu s$ pulse or pause) and the total initialisation time are known or can be calculated, it is possible to find the time (relative to the initiation of the transaction) when device TI started emitting infrared. Any infrared signals received by device TT prior to the initiation of the emission or after its completion are discarded by the comparing party.

The average total infrared emission process time by device TI was measured to be 226.66ms, by performing 300 runs of the experiment, using an SGS5 mini (SM-G800F) device, running Android 5.1.1. In order for the relay attack detection method to be performed during the time of the transaction, the similarity comparison between the transmitted and the captured bit-sequence has to be performed by one of the devices involved in the transaction. The required data should be transferred through a secure channel, in this occasion encrypted through the NFC channel that is running in parallel.

In order to accelerate the relay attack detection technique, TI can encrypt and transfer the random-bit sequence to TT, through the NFC channel, in parallel to the infrared emission process. Since the two devices are considered trusted, the security of the system is maintained. Once the emission of the infrared is complete, the time required for the process to complete is also transferred in a secure manner from device TI to device TT. Device TT discards any bits received outside the emission time-frame

and proceeds with the comparison of the emitted and received bits. The `wdiff` [58] program is responsible for the comparison process. The `wdiff` program was used instead of the `dwdiff`, which was used in Chapter 4, because it produced the exact same results, but its performance was found to be slightly better. If the computed similarity is greater or equal to 98%, device TT responds with an approval message to device TI. Contactless transactions tend to use either public, or symmetric key cryptography, hence the integration of the PRAD mechanism was evaluated against these scenarios.

Public Key-based Protocol

In mobile payments and physical access control, public key cryptography is usually recommended for use in related protocols, according to EMV [52] and NIST [107], respectively. The protocol messages used for the secure exchange of the required data between the two devices are listed in Protocol 1. A mechanical formal verification of the protocol has been performed, using the Scyther tool, against all automatic Scyther claims, except the secrecy of data sent in plain text. The verification script can be found in Appendix B. No attacks were detected through the mechanical formal verification. Note that resilience against relay attacks of the protocol itself cannot be verified through Scyther. The tool is only used to prove that data marked as *Secret* or *SKR* in the scripts, will not be revealed to an adversary, assuming that the cryptographic functions are perfect.

Protocol 1 Public-key Based Protocol

1: $TT \rightarrow TI : Cert_{Auth}(TT)||ID_{TT}||n_{TT}$
2: $TI \rightarrow TT : E_{PK_{TT}}\{ID_{TI}||ID_{TT}||n_{TI}||n_{TT}||K||IRseq\}$
$||S_{SK_{TI}}[K||n_{TT}]||Cert_{Auth}(TI)$
3: $TI \rightarrow TT : E_K\{ID_{TI}||ID_{TT}||n_{TI}||n_{TT}||IRtiming\}$
4: $TT \rightarrow TI : E_K\{ID_{TI}||ID_{TT}||n_{TI}||n_{TT}||approval||S_{SK_{TT}}[n_{TI}]\}$

– **Message 1:** Device TT sends to device TI its public key certificate $Cert_{Auth}(TT)$, its ID ID_{TT}, and a random nonce n_{TT}.

– **Message 2:** TI verifies the received certificate. If it is genuine, device TI encrypts, using the public key of device TT, its ID (ID_{TI}), the ID of device TT (ID_{TT}), the random nonces n_{TI} and n_{TT}, a session key K, and the random-bit sequence that is emitted through the infrared channel ($IRseq$). The aforementioned encrypted message, along with the RSA public key certificate of device TI, and a signature on K and n_{TT} —using the private key of device TI— are sent to device TT.

– **Message 3:** Device TI sends to device TT, ID_{TI}, ID_{TT}, n_{TI}, n_{TT}, and the time required until it was capable of emitting infrared signals ($IRtiming$). The data is encrypted with the session key K. This message is separated from the previous for performance reasons, as $IRtiming$ is available after the infrared sequence is emitted, during which time $IRseq$ can be sent.

– **Message 4:** If the certificate of device TI, as well as the signature from Message 2 are verified, device TT proceeds to compute the emitted and captured infrared bit similarity. Device TT takes a decision and responds with an approval or rejection message. The decision message, along with ID_{TI}, ID_{TT}, n_{TI}, n_{TT}, and a signature on n_{TI} —using the private key of device TT— are sent to device TI. The message is encrypted using the session key K (as in Message 3).

Symmetric Key-based Protocol

In the field of smart ticketing, protocols are often based on symmetric key cryptography. For example, the MIFARE Plus card [117], and the Calypso ticketing system [22]. In the case of the MIFARE Plus card, in order for the security of the system to be maintained in case the key of a card is compromised, each card uses a different key. The terminal contains a master key, from which the symmetric key of a card/smartphone is derived through a process called symmetric key diversification [4]. The card provides the Unique Identification number (UID), the Application ID (AID), and the System Identifier (SID) to the terminal. This information is used as input to the diversification process in order for device TT to generate the key stored in device TI.

In a smartphone-based transaction, a dynamic element should also be included in the diversification process, possibly provided by the scheme operator. Without a dynamic element, a device with a compromised key might not be able to be used for this type of transaction again. Replacing the device in such scenarios is expensive. However, developing a diversification process that takes as input dynamic elements is out of the scope of this thesis.

A symmetric key-based protocol was designed (Protocol 2) and undergone mechanical formal verification, using Scyther, against all automatic Scyther claims. The verification script can be found in Appendix C. No attacks were detected through the mechanical formal verification. For the verification of this protocol, Scyther Compromise was used [37], which supports all the protocols that are available for the regular version of the program. The *SKR* function that is used in the verification script, and is not standard for the regular Scyther version. In the script, it is treated as equivalent of the *Secret* claim of the regular version, but further marks the variables as session keys [38]. In this occasion as well, resilient against relay attacks is not implied.

Protocol 2 Symmetric-key Based Protocol

1: $TI \rightarrow TT : \textit{Key Diversification Information}$
2: $TT \rightarrow TI : E_K\{ID_{TT}||n_{TT}\}$
3: $TI \rightarrow TT : E_K\{ID_{TI}||ID_{TT}||n_{TI}||n_{TT}||IRseq\}$
4: $TI \rightarrow TT : E_K\{ID_{TI}||ID_{TT}||n_{TI}||n_{TT}||hash[IRseq]||IRtiming\}$
5: $TT \rightarrow TI : E_K\{ID_{TI}||ID_{TT}||n_{TI}||n_{TT}||hash[IRseq]||approval\}$

– **Message 1:** Device TI sends the information used by the diversification algorithm in order for device TT to generate the symmetric key K. A symmetric key compromise

in a MIFARE card scenario requires replacement of the card, as the diversification data is static. Replacing a mobile device in such scenarios is very expensive and impractical. Therefore, the diversification information in this scenario includes the Device ID and a dynamic ID that can change in case the original key is compromised.

– **Message 2:** Device TT sends its ID (ID_{TT}) and a random nonce (n_{TT}) to device TI. The message is encrypted using the symmetric key K, which is pre-shared between the two devices.

– **Message 3:** TI responds with its ID (ID_{TI}), the ID of the terminal (ID_{TT}), random nonces n_{TI} and n_{TT}, and the random-bit sequence that is emitted through infrared ($IRseq$). The message is encrypted using key K.

– **Message 4:** Upon completion of the emission of the random-bit sequence through infrared, device TI sends the emission initiation time ($IRtiming$) to device TT. The message also contains the identities and the random nonces generated by the two devices, as well as the hash of $IRseq$. The last element is provided so that an attacker is not capable of flipping the message order of messages 2 and 3. This message is also encrypted using the key K.

– **Message 5:** Using the provided $IRseq$ and $IRtiming$, device TT returns a rejection or approval message to device TI. The approval, along with ID_{TI}, ID_{TT}, n_{TI}, n_{TT}, and a hash on $IRseq$ are sent encrypted with key K to device TI.

5.2.3 Evaluation Framework

In order to evaluate the performance of the protocols and the proposed architecture, working prototypes were built. In all scenarios, a laptop running Fedora 25 was used as device TT, with an i7–2620M CPU at 2.70GHz processor and 8GB of RAM. Device TI was represented by an SGS5 mini (SM-G800F) device, running Android 5.1.1. The device is regarded to be of low specifications. Two applications were built for each device, one for the evaluation of each of the two protocols. Both protocols are relevant in the case of contactless transactions, since public key cryptography is suggested in the cases of contactless payments and physical access control, while symmetric key is usually suggested in the case of transportation ticketing.

An NFC-reader (ACS ACR1281-C1) was attached to device TT. The applications for device TT were developed in Python, using the `pyscard` library [16], in order to control the NFC-reader. Android applications were developed in Java, using HCE in order to control the NFC adapter of the device.

Raw infrared data captured during the experimental phase of Chapter 4 was used for the emulation of infrared emission/capture. In order to further weigh potential impact on the performance of the system, device TI was emitting infrared signals during the time of the transaction. The performance impact of capturing infrared data is minimal. A Raspberry Pi with an infrared receiver connected to its GPIO pins was used to measure the system load when receiving infrared signals. The `liblirc` library [102] was used, and the performance burden on the device was negligible.

Public Key Based

In the first scenario, which applies in domains that require the use of public key cryptography in contactless transactions, 2048-bit RSA was used. Each device contained a public key pair, used for the transaction.

Upon the establishment of the session key, AES 128-bit encryption, using the CBC mode and PKCS5 padding, was used. All the keys and nonces were randomly generated, using the `SecureRandom` Java class on device TI and the `Random` module of the `pycrypto` Python library [104] on device TT.

The bit-similarity comparison between the captured and transmitted infrared sequence was performed by device TT, upon receiving the third message. Bits captured outside the time-frame during which device TI was transmitting were discarded. The `wdiff` tool was then called through the `check_output` module of the `subprocess` Python library. The similarity percentage was returned to the application running on device TT. The *approval* flag was finally sent as part of the last message to device TI, based on that percentage (`0x01` if the similarity percentage was equal or exceeded the threshold of 98%, otherwise `0x00`).

Symmetric Key Based

Similar to the public key-based scenario, 128-bit AES, CBC mode, with PKCS5 padding was used in the symmetric key-based scenario as well. The bit-similarity comparison was performed upon receiving the fourth message, by device TT, in the same way as in the public key-based scenario. The key diversification code, according to the document AN10922 [4], was integrated by modifying the implementation in [17]. Finally, SHA256 was used as the hashing algorithm.

5.2.4 Results

Each of the protocols was executed 100 times in order to estimate the performance of the system and evaluate the applicability in contactless transaction scenarios. The performance measurements can be found in Table 5.1. The measurements were taken by device TT and represent the round trip time (i.e. the time between device TT starts

Table 5.1: Results of Contactless Transactions (in *ms*)

	Protocol 1			**Protocol 2**			**Protocol 2 (300b)**		
	min	***max***	***avg***	***min***	***max***	***avg***	***min***	***max***	***avg***
Diversification	-	-	-	30.2	42.4	39.0	28.1	50.5	38.8
Round Trip 1	116.1	169.4	152.2	26.6	81.5	32.6	21.9	40.9	27.6
Round Trip 2	95.7	135.5	109.4	146.6	201.5	194.7	95.5	115.2	109.0
Round Trip 3	52.1	133.7	70.1	24.4	32.1	27.2	24.6	38.0	28.6
Bit Similarity	3.8	5.9	4.3	3.8	5.2	4.5	3.5	5.2	4.1
Total	304.9	395.4	331.8	284.3	299.7	293.5	193.8	215.0	204.0

generating a message and the response that comes from device TI is processed). Since the clocks of the two devices cannot be perfectly synchronised, practically measuring the timing of each individual message was not feasible. In the first message of each protocol, the timing of the *SELECT* command (APDU) is also included.

The last column lists the results of the second protocol with lower number of bits emitted (300 bits), for better integration with transport related contactless transactions, which require a time restriction of 300–500ms (further discussion in Section 5.2.5). The minimum and maximum observed timings in the 100 protocol runs and the average running time are listed for each of the protocol messages. The time required for the diversification process, the comparison of the transmitted and captured bit-streams, and the total protocol running time are also listed.

5.2.5 Discussion

The results indicate that the running time of the first protocol are within the timing bounds that EMV requires. Even though the infrared emission lasts 100ms, the average total emission time was approximately 227ms. The aforementioned emission time refers to the time between requesting the infrared sequence emission from the operating system, until the completion of the process. As already discussed in Section 4.3, after inspection of the Android Open Source Project's (AOSP) source code [10], it was concluded that this delay is imposed either by the proprietary infrared drivers, or by modifications on Android's source code by Samsung. Since access to the source code of these components is not available due to their proprietary nature, it was not possible to investigate the problem further. However, it was possible to emit infrared signals, using an infrared LED attached to the GPIO pins of a Raspberry Pi, controlled by a program written in the C language, with negligible delay. Therefore, it is believed that this delay can be confined. An average extra performance cost of approximately 57ms and 82ms was measured in the public and symmetric key scenarios respectively, in the case of 500 bits.

It should be stressed that the measured performance is an indicator of the performance cost of NFC contactless transactions, using the proposed solution. An additional barrier was the multitude of EMV and physical access control implementations, as well as the limited available information regarding major transport ticketing protocols. However, industry standards like the EMV specifications were used as guidelines for developing and delivering a robust solution.

In a real-world scenario, more information is exchanged between the devices as part of the transactions, that is not covered by the protocols presented in this chapter. Real-world applications, after the establishment of a secure channel between the devices involved in a transaction, would require to transfer the random-bit sequence and the total emission time, in order for the similarity comparison to be performed.

Since some transportation ticketing transactions require to be completed within 300–500ms, as mentioned earlier, the performance of the second protocol might not be optimal for some slower devices. Therefore, the experiment was repeated with

a reduced number of random bits. A total of 300 randomly generated bits were generated in the second experimental round, constituting an emission time of 60ms. The probability of an attacker guessing the random sequence, assuming its true random nature, is $\frac{1}{2^{300}}$. The average performance of the protocol run was improved, with potential to successfully be integrated into such protocols. The average total emission time, caused due to the abovementioned delay in the emission, was 137ms. An average extra performance cost of approximately 32ms was introduced due to the issue.

5.3 Use Case: Continuous Two-Factor Authentication

In order to add an extra layer of security, access to websites, services, or systems, may require an extra verification step in addition to the login credentials. This second layer of security is often referred to as *Two-Factor Authentication* (2FA). A widely adopted 2FA technique is the use of a One Time Password (OTP), which can be sent to the user via SMS message, generated by an application on the user's device, or other means. In the case of host-based 2FA, hardware tokens, like smart cards or OTP generators, are often used. Upon the successful completion of the login credentials, users are typically requested to proceed by providing the 2FA information in order to successfully complete the login process. Access to a service that provides 2FA functionality is only granted upon successful completion of both authentication steps.

Using an OTP as a second factor usually requires input of the OTP once per session. Depending on the system settings, a session may last for several months without requesting an OTP again (e.g. access to a website). Continuous authentication is not achieved in this case. Moreover, being able to provide the OTP to the service does not imply the user's physical presence close to the device. Shoulder surfing attacks can be used for example by attackers in order to compromise the OTP and login in a remote location [27], as the expiration time of these passwords can be several seconds long.

In the case of smart cards, an extra cost is associated, and distance bounding protocols may not be applicable in this scenario. The multi-process nature of the host (e.g. a server or a personal computer to which a user wishes to login to), the operating system and its configuration, and the hardware variability do not assist towards this direction (potentially a similar issue as the one described in [156] will exist).

A 2FA model is hence proposed, based on the principle of infrared as a proximity detection method, capable of providing continuous authentication, while eliminating the risk of attacks, like shoulder surfing and relay attacks. In this scenario, the secure primary channel is established over WiFi or Bluetooth between devices TT and TI (a personal computer and a smartphone, respectively). The second channel is the infrared channel, as described in the previous sections. Further discussion on the architecture can be found in Section 5.3.2.

5.3.1 Threat Model

Since relay attack detection is based on the comparison of data captured by both devices TT and TI, both devices should be trusted. In case device TT is not trusted, captured data can be manipulated or delayed, which can lead to a false proximity result. In the case of a website, which can be accessed from multiple devices/locations, device TT cannot be considered trusted. By requiring some trusted execution environment, like Intel SGX [35], trust in device TT can be obtained. However, even though a relay attack might not be applicable in this scenario, an attacker in the vicinity to device TI can successfully login using a computer (e.g. a Raspberry Pi), which is controlled locally or remotely. Therefore, this technique is only applicable in scenarios where logging in to a service can be achieved through a single, trusted device, like a user login to an operating system and other host-based services.

Similar to contactless transactions, in order for an attacker to perform a relay attack, a relay pair TT'–TI' is required. The goal of the attacker is to be able to successfully relay infrared signals in a timely manner. Since the same design principles described in Section 4.2.1 are used, an attacker has a very limited window for caching infrared signals and replaying them at a remote location.

5.3.2 Candidate Integration Architecture

The generic architecture of the proposed system is depicted in Figure 5.1. When a user logs in to a system that supports the proposed solution, a communication initiates with an underlying service, responsible for the 2FA process. In order to provide continuous authentication, the service is called periodically (e.g. every 15 seconds). Whenever the service is called, it communicates with device TI through a WiFi or Bluetooth channel in a secure manner (i.e. using a secure protocol). Device TI must be running an application capable of communicating with the service. The application running on device TI should be linked to the service that device TT is attempting to login to (e.g. by prior login of the user to the mobile application).

Depending on the settings of the service, the underlying service can be called periodically and request from device TI to provide an authentication token, in the

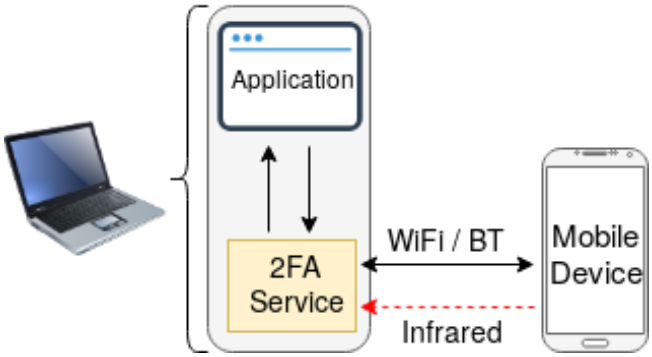


Figure 5.1: Architecture of Two-Factor Authentication

form of random infrared bits. If the authentication is unsuccessful, an action is taken. Actions can vary according to the requirements of the service. For example, the active session on device TT can be discontinued.

Since the success of the authentication transaction depends on the orientation of the two devices (i.e. whether the infrared blaster is aligned with the infrared receiver), less drastic measures might be taken instead. For example, permission escalation can be used to restrict access to certain critical actions and directories.

Either of the two protocols described in Section 5.2.2 can be used for the establishment of the secure channel, depending on the requirements and the implementation. For example, a symmetric key can be shared during the setup process by asking the user to scan a barcode. The key can be periodically updated, but this is out of the scope of this chapter. In multi-user environments, such keys can be generated through symmetric key diversification. Public key cryptography might be a better option for logging user access in such scenarios.

5.3.3 Evaluation Framework

The same devices described in Section 5.2.3 were used in this scenario as well as devices TI and TT. Instead of an NFC adapter, two alternate primary communication channels were examined; WiFi and Bluetooth.

In order to evaluate the applicability and performance of the proposed architecture, a working prototype was built. An application was developed in Java, representing the application that requires 2FA as part of the authentication process. In order for a user to remain logged in, the successful completion of the 2FA had to be achieved. Otherwise, a failure message was displayed and access to certain components of the application was restricted. The 2FA service was called every 15 seconds, in order to provide continuous authentication assurance.

Four Android and four Python applications were built for TI and TT, respectively, in order to assess all the possible combinations. Namely, both protocols described in Section 5.2.2 were implemented for each of the two primary channel options (WiFi and Bluetooth) applicable to this scenario. In the case of WiFi, the two devices were connected to the same wireless network, while in the case of Bluetooth, a pairing between the devices had to be established before the initiation of the 2FA process. The pairing was performed through the Python applications, using the `PyBluez` library [95].

Upon initialisation of the application requiring 2FA, a call towards the underlying service was triggered. The service, written in Python, would then establish a connection to device TI in the case of Bluetooth. In the case of WiFi, no session was required, since the communication was performed through UDP packets. Device TT would then request from device TI to initiate a transaction through the established Bluetooth Radio Frequency Communication (RFCOMM) channel, or a UDP packet. Upon transmission of the message, device TT would begin listening for infrared signals through an infrared receiver, as in Section 5.2. Upon receiving the message,

device TI would start emitting infrared signals through its infrared blaster. Infrared data captured during the experimental phase of Chapter 4 was used in this case as well. The communication would proceed in parallel through both channels. Using the `wdiff` tool, device TT would compute the similarity between the emitted and captured infrared sequences. If the similarity threshold was reached, upon completion of the protocol the Python service would allow access to certain elements of the application.

5.3.4 Results and Discussion

The results of 100 protocol runs over WiFi and 100 over Bluetooth are presented in tables 5.2 and 5.3, respectively. For each protocol, the minimum, maximum, and average running times are listed.

Table 5.2: Results of 2FA — WiFi (in *ms*)

	Protocol 1			**Protocol 2**		
	min	***max***	***avg***	***min***	***max***	***avg***
Diversification	-	-	-	2.6	5.2	2.7
Round Trip 1	21.7	98.0	30.3	6.2	67.9	8.6
Round Trip 2	131.5	252.6	201.2	162.1	277.3	224.2
Round Trip 3	16.8	62.1	22.6	6.3	18.2	7.9
Bit Similarity	4.0	6.9	4.7	3.9	10.6	4.6
Total	246.3	305.2	254.2	238.4	297.3	243.3

Table 5.3: Results of 2FA — Bluetooth (in *ms*)

	Protocol 1			**Protocol 2**		
	min	***max***	***avg***	***min***	***max***	***avg***
Diversification	-	-	-	2.6	3.5	2.7
Round Trip 1	74.3	182.2	116.9	52.1	153.9	76.9
Round Trip 2	123.2	200.6	175.2	129.0	262.3	209.9
Round Trip 3	54.5	144.7	70.7	35.4	154.3	57.2
Bit Similarity	3.9	5.7	4.6	3.8	13.5	4.6
Total	323.6	440.1	362.8	306.2	426.2	346.8

The results indicate the effectiveness of the proposed solution. The overhead of the process was minimal, and the effectiveness high. From a usability perspective, the setup process needs to be completed once, and thereafter the user only has to place device TI to be facing towards device TT. In order to enhance the security of the system, the user may be required to unlock the device in order for the process to initiate.

A usability concern may rise when the user needs to use device TI to make or receive a call, while using device TT. A temporal suspension of the continuous

authentication process can be applied in this occasion, in case an initial login using both factors has already been completed.

An attacker with access to device TT can further use techniques in order to compromise the system. For example, modify the security parameters by externally plugging the hard drive to a computer. Disk encryption can assist towards the prevention of such attacks.

Finally, since no time restrictions apply in this scenario, the natural ambient environment may be used as a potential alternative, instead of generating an AAE and using infrared as the actuator. However, manipulation or prediction of the natural ambient environment might be possible. For example, if the room temperature is used to detect proximity, a heater or an air-conditioner can be used in order to alter the values to the attacker's benefit. In case of ambient sound, an attacker can compromise the system by following the habits of a legitimate user. For example, when a user is watching a television show, the attacker can produce the same ambient sound (by tuning to the same television channel), and successfully authenticate. Attacks on using the sound as a proximity detection mechanism have been described and demonstrated in previous works [94, 140]. Finally, in the case of light, an attacker can cover the light sensor of device TT when device TI is in the pocket of the legitimate user, and gain access to the system. Therefore, using the natural ambient environment may not effectively protect under certain circumstances, like in the presence of an attacker with context manipulating capabilities.

5.4 Other Scenarios

Other scenarios, that are either not in compliance with the context of this chapter (Section 5.4.1), or are covered by the aforementioned scenarios (Section 5.4.2), are described in this section.

5.4.1 2FA for Logging In to Web Services

An attacker who has the ability to place a device in the vicinity of TI may be able to successfully login, using the standard procedure. Although this is not a relay attack, as no data is being relayed between devices TI and TT, access to the legitimate user's account can be achieved. In scenarios where a relay attack is not of concern, infrared emission can operate as a 2FA provider, without requiring device TT to be trusted. For example, if an attacker has compromised the credentials of a user, but does not have access to the vicinity of device TI. Since the main objective of this chapter is to provide anti-relay capabilities in real-world scenarios, and the aforementioned case does not fit this context, it is listed as a future work direction.

As a note, it should be stressed that in such scenario the authentication must not be performed by device TT, but by the service provider or device TI. Since device TT is not trusted, the overall architecture of the system in this occasion should be modified. The resulting architecture should be similar to the architecture of Sound-Proof, as

described by Karapanos et al. in [94]. In their proposal the use of ambient sound is employed in order to provide device proximity proof and use it as a means of zero-effort 2FA. The work is mostly focused on usability, and potential security problems have been discussed by both the authors and by Shrestha et al. in [140]. It is believed that most of the discussed security threats can be eliminated by replacing sound with a randomly generated infrared sequence, however some usability might be diminished.

5.4.2 Internet of Things Co-Presence Verification

In many occasions, in order for Internet of Things (IoT) devices to achieve their intended operation, proximity should be guaranteed. For example, in a security system, in order to avoid blind spots, in many cases devices should be located within visibility range. An attacker can move some components of the security system and apply a relay attack in order to deceit the system.

By applying the proposed solution, proximity evidence can be provided among such devices. A similar approach to the ones described in this chapter can be applied. When one IoT device requests proximity evidence from another, the same procedure is executed, with the requesting device acting as device TT and the proving device as device TI. In this scenario, the IoT devices of the system should be considered as trusted as well.

5.5 Summary

In this chapter, the integration of infrared as an AAE actuator for relay attack prevention in different real-world scenarios was investigated. Existing industry standards were taken into consideration, where applicable. Contactless transactions (EMV payments, physical access control, and transport ticketing), as well as continuous host-based two-factor authentication were investigated. Online two-factor authentication, and IoT co-presence verification were also discussed.

Integration architectures were described, and working prototypes were built and evaluated. Proposed protocols were undergone mechanical formal evaluation. The results indicated that integration of the proposed solution with existing industry standards can be used to counter relay attacks, and is in compliance with current operational timing requirements, where applicable.

Chapter 6

Vibration as an Artificial Ambient Environment Actuator

6.1 Introduction

In previous chapters of this thesis, AAE as a means of PRAD was described, and the use of infrared light as an AAE actuator was evaluated for its effectiveness and applicability in real-life scenarios. In this chapter, the effectiveness of vibration as an AAE actuator is investigated.

In this case, one or both transaction devices vibrate a randomly generated pattern. The impact of the vibration is recorded by both transaction devices, using a selected ambient sensor. The recorded data from the two devices is subsequently compared against each other for similarity. For this approach to be effective in terms of providing PRAD-related evidence, the comparison of two recordings that are genuinely in proximity should be distinguishable from comparing recordings that are not in proximity. Moreover, it should be difficult for an attacker to replicate the same vibration pattern at a remote location, by relaying it or otherwise. More specifically, the challenge for the attacker is to replicate the vibration both accurately and in a timely manner. In case there is a delay in replicating the pattern, part of the impact might not be captured by one of the transaction devices. To further minimise the probability of the attacker accurately replicating the vibration at a distant location, both transaction devices can vibrate a random pattern (different for each device) simultaneously. This way, a fusion of the two vibrations will be recorded.

In order to assess the effectiveness of the proposed solution, an evaluation test-bed was deployed. During the trials, 36,000 genuine and relay transaction pairs were used, covering a wide range of potential techniques that an attacker might use against the proposed system. Analysis with popular machine learning algorithms (same as in Chapter 3) indicates that the proposed solution is highly effective as a PRAD mechanism, especially in the case of applying a gyroscope as the ambient sensor to measure the impact (0.1% EER).

6.2 Vibration as an AAE Actuator

In this section, the basic principles of using vibration as an AAE actuator are presented.

6.2.1 AAE Framework

The genuine user is asked to place the Transaction Instrument (TI) on the Transaction Terminal (TT). Upon initiation of the transaction, one (unidirectional) or both (bidirectional) transaction devices vibrate using some randomly generated pattern. Simultaneously, both devices use some ambient sensor to measure the impact of the vibration over a period of time. A 500ms long vibration pattern and recording was used. In the case that both devices vibrate, a fusion of the two random vibrations is causing the impact. Candidate sensors, available on a wide range of modern smartphones that can potentially be used to measure the impact of the vibration include:

1. the accelerometer,
2. the geomagnetic rotation vector,
3. the gyroscope,
4. the magnetic field,
5. the rotation vector,
6. the gravity sensor, and
7. the linear acceleration sensor.

The vibration-based AAE channel (referred to as the *'vibration channel'*) is used as an out-of-band channel[5], along with the main communication channel (e.g. NFC, WiFi, etc.), aiming to provide proximity assurances. The initiation of the vibration channel should be subsequent to the initiation of the main communication channel for each of the devices, with as minimal delays as possible, in order to minimise the attack window.

Upon completion of the transaction, data streams captured by the sensors of the two devices are compared for similarity. The similarity comparison can be performed either by one of the transaction devices, or by a TTP. The exact characteristics and architecture of the party responsible for the comparison (i.e. one of the transaction devices or the TTP) are beyond the scope of this chapter. Of course, the recoded data streams should be communicated to the comparison party in an encrypted and authenticated form.

The comparison of the data streams should provide adequate information regarding the coexistence of the communicating devices. The comparison party takes a

[5]Out-of-band channel: Second channel

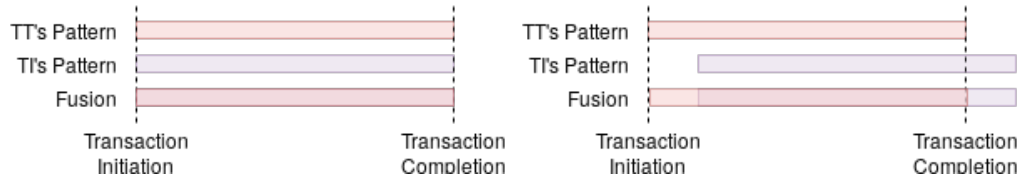
decision, based on which the transaction is either approved or rejected. The assumption of this technique is that juxtaposing the recordings of the vibrated pattern(s) will be distinguishable when the transaction devices are in proximity, compared to otherwise.

6.2.2 Threat Model

The attacker, in this chapter as well, is considered to be of opportunistic nature and requires no prior interaction or knowledge of either TT or TI. Scenarios where TT or TI are compromised will not be covered. Instead, the focus is on evaluating the proximity assurances that can potentially be provided for genuine devices, when using vibration as an AAE actuator.

It is also assumed that the attacker only has access to off-the-shelf relay equipment. Transaction limits often apply on smartphone-based transactions, for example the £30 limit that is currently imposed on digital payment transactions in the UK (as discussed in the previous chapter). Therefore, due to the limited financial gain involved, a powerful attacker with advanced and expensive relay equipment is not the major concern.

For the same reason, it is assumed that the attacker can only try to guess the pattern vibrated by the genuine devices. More advanced attacks, like acoustic attacks, during which an attacker captures the vibration pattern by performing acoustic analysis and replays it at a distant location, are not considered. Even though acoustic attacks have been demonstrated in the context of decoding a vibration pattern [67], combining an acoustic and a relay attack by relaying the decoded pattern at a remote location in real time is a challenging task for off-the-shelf attackers. Delays associated with audio recording latency, signal processing, data communication between the two relay devices, and replaying at a distant location, which would be required for such an attack, would significantly increase its complexity. Note that delay in relaying the pattern at a distant location would lead to a shift of the relayed vibration pattern on that side (e.g. Figure 6.1). However, an emulation of such an attack has been considered (during which both relay devices vibrate the same patten), and is discussed in the following sections.



(A) No Relay Attack (B) Relay Attack

FIGURE 6.1: Vibration Fusion Without and With Relay Attack

6.3 Evaluation Framework

In order to evaluate the proposed solution, a test-bed framework was deployed. Four Android devices were used to emulate a genuine (proximity) and a relay transaction. The two transactions (genuine and relay) were being captured at the same time when required, or generated based on the captured data otherwise. Three scenarios were evaluated:

1. both transaction devices vibrating,
2. only the transaction terminal vibrating, and
3. only the transaction instrument vibrating.

Each of these scenarios was evaluated with each of the six sensors:

1. accelerometer,
2. gravity sensor,
3. gyroscope,
4. linear acceleration sensor,
5. magnetic field, and
6. rotation vector.

The communicating devices in the genuine transaction scenario were a Transaction Terminal (TT), and a Transaction Instrument (TI'). In the case of the relay transaction, the communicating devices were the same transaction terminal TT, and a distant Transaction Instrument (TI), located 5ft (1.5m) away. Devices TI', and a transaction terminal co-located with TI (referred to as TT'), were used as relay devices. The transaction scenarios are as in previous chapters, as depicted in Figure 3.3.

Device TI' had a double role in this set-up as well. It acted as both a genuine and a relay device. This way, data from both scenarios could be collected simultaneously when possible (as in Chapters 3 and 4). The two devices of each pair were placed one on top of the other, such that the transaction terminal was at the bottom, facing downwards, and the transaction instrument on top, facing upwards. The two pairs (genuine and relay) were placed on separate tables, such that vibration of one pair would not affect the measurement of the other.

Five different attack techniques were performed against each sensor and each vibration scenario. Four scenarios were obtained by assuming that the attacker was capable of vibrating a pattern from one, both, or none of the transaction devices. A further scenario was obtained by assuming that the same pattern could be vibrated on both relay devices during a single transaction (emulating a perfect acoustic attack).

The notation that is used to describe the vibration mode of the transaction and relay devices in this and subsequent sections has the form of four characters, each character representing either *'Vibration'* or *'No Vibration'* using *'V'* and *'N'* respectively. The sequence of four characters represents the four devices of the test-bed, as: *TT TI' TT' TI*. For example, *VNVN* means that TT and TT' vibrate, while TI' and TI do not.

In total, 15 distinct vibration mode sets were part of the empirical analysis. Out of these 15, five sets were collected through field trials, and ten sets were generated based on the raw data from the field trials (these sets are referred as *'synthetic datasets'*). For the generation of the synthetic datasets, another two sets had to also be collected through field trials.

In the case of synthetic datasets, transactions collected by the pair TT–TI' through field trials were regarded as genuine transactions. Relay transactions were based on combining genuine transaction measurements performed by TT and TI from different, experimentally collected sets. A point to note is that synthetic data was based on raw field-data, and was not generated randomly or based on artificial-replication of specific features.

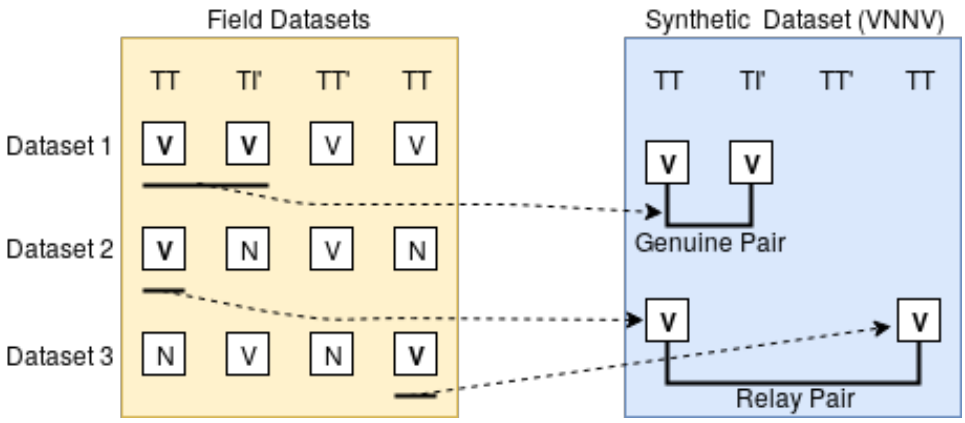


FIGURE 6.2: Synthetic Data Generation (VNNV)

Figure 6.2 depicts the generation process of the synthetic dataset *VNNV* as an example case. In this case, a genuine transaction (no relay involved) required both transaction devices to be vibrating. Three sets of raw field-data for the construction of this case were used. Considered as genuine transactions are transactions between TT and TI' from the set *VVVV*. For relay transactions, raw field-data captured by device TT from the set *VNVN* were used (where TI' did not vibrate and TT did), and data captured by device TI from the set *NVNV* (where TI vibrated and TT' did not). Table 6.1 lists all the evaluated vibration scenarios and experimentally recorded pairs that were used for the generation of synthetic pairs. Under the *'Generated By'* heading of the table, bold characters in vibration mode sequences identify the raw field-data that constitute the synthetic data for the respective mode. The two extra sets (discussed before) are marked as *'Other'* in the table.

The rationale for the synthetic data generation is due to one of the following two reasons — depending upon the vibration mode sequence:

TABLE 6.1: Vibration Modes

	TT	TI′	TT′	TI	Experimentally Recorded	Generated By	
						Genuine Pair	Relay Pair
Both Vibrate	V	V	V	V	✓	—	—
	V	V	V*	V	✓	—	—
	V	N	V	V	✗	**V**VVV	**V**NVN + VVV**V**
	V	V	N	V	✗	**V**VVV	**V**VVV + NVN**V**
	V	N	N	V	✗	**V**VVV	**V**NVN + NVN**V**
TT Vibrates	V	N	V	N	✓	—	—
	V	V	V	N	✗	**V**N**V**N	**V**VVV + VNV**N**
	V	V	V*	N	✗	**V**N**V**N	**V**VV***N** (Other)
	V	V	N	N	✗	**V**N**V**N	**V**VVV + NNN**N**
	V	N	N	N	✗	**V**N**V**N	**V**NVN + NNN**N**
TI Vibrates	N	V	N	V	✓	—	—
	N	V	V	V	✗	**N**V**N**V	**N**VNV + VVV**V**
	N	V	V*	V	✓	—	—
	N	N	V	V	✗	**N**V**N**V	**N**NNN + VVV**V**
	N	N	N	V	✗	**N**V**N**V	**N**NNN + NVN**V**
Other	V	V	V*	N	✓	—	—
	N	N	N	N	✓	—	—

*TT′ vibrates the same pattern as TI′

1. Both a genuine and relay transaction could not be captured simultaneously (i.e. when TI′'s vibration mode was not the same as TI).

2. Recorded data could be used for the generation of a pair (e.g. in the case of VVNV, whose genuine pair can be taken from VVVV, and the relay pair by combining the same pair's TT recording with NVNV's TI recording).

A total of 400 transactions per captured pair were recorded for each sensor (totalling 16,800 transaction pairs), based on which, 24,000 synthetic relay transactions were generated (400 for each synthetic set). Each synthetic set consisted of 400 genuine transactions, recorded through the experimental test-bed, and 400 synthetic ones, based on the combination of recorded transactions.

Four Android applications were developed, one for each device. The devices used in the genuine pair were two SGS4 (GT-I9500) devices. For the relay pair, an SGS4 and a Nexus 5 were used. Each device was capable of communicating with the next device and the previous device in the chain (see Figure 3.3) through WiFi, using UDP packets. For example, device TI′ could communicate with device TT as part of the genuine transaction, and TT′ as part of the relay transaction.

Figure 6.3 depicts the transaction process between the four devices. A transaction was initiated through a UDP packet from device TT to TI′, containing a transaction ID, the sensor to be used in the transaction, and which device(s) should vibrate (vibration mode). The same information, along with the random pattern that TI′ would vibrate, if the vibration mode required it, were forwarded from TI′ to TT′.

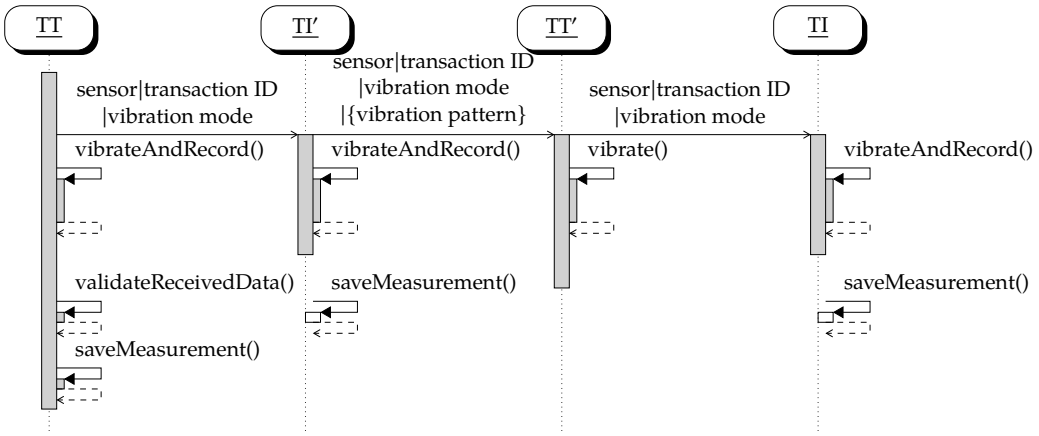


FIGURE 6.3: Measurement Recording Overview

Upon initiation of a transaction, a 500ms long random pattern was vibrated by one or both transaction devices, depending on the current transaction's vibration mode. At the same time, the devices were recording the impact of the vibration, using some sensor. Upon completion of the transaction, the measurements were stored in a local SQLite database, for each of the relevant devices for later extraction and analysis. The stored data was kept in XML format, as in Listing 3.1. Subsequent transactions were separated by a 5 second gap in order for the devices to rest. Upon completion of the field trials, the local databases were transferred to a computer for analysis.

6.4 Transaction Data Analysis

For the analysis of the collected data, the Weka machine learning software was used to determine whether the measurements of TI' and TI were in proximity with TT, i.e. whether it was possible to uniquely distinguish between (TI'_i,TT_i) and (TI_i,TT_i). As already explained, on Android, all the examined sensors produce a vector of values consisting of x, y, and z components. In this chapter as well, as in Chapter 3, the vector magnitude (Eq. 3.10) was used as a general-purpose method for producing a single, combined value, prior to generating the training sets for machine learning.

Following the evaluation procedure of Chapter 3, the 'optimal' threshold that separates illegitimate and legitimate attempts was calculated: the threshold that minimises the EER. The threshold was based on the probability estimate output by the learned classification model, i.e. the estimated probability that a transaction is legitimate. A train-test experiment was necessary in order to avoid optimistic bias in the error estimate when applying machine learning.

The same machine learning algorithms and approaches as the ones discussed and used in Chapter 3 were used in this chapter as well. More specifically, a training and a test set were created by the full set of transactions. It was ensured that legitimate and illegitimate transactions that shared the same transaction ID and used sensor (i.e. recorded simultaneously), both existed in either the training or the test set, in order to avoid potential bias. In this occasion as well, 10-fold cross-validation repeated

TABLE 6.2: Estimated EER for Machine Learning Algorithms (obtained by repeating 10-fold cross-validation 10 times)

Vibration Mode	Classifier				
	Random Forest	**Naïve Bayes**	**Decision Tree**	**Logistic Regression**	**Support Vector Machine**
Accelerometer					
Both Vibrate	0.069	0.094	0.104	0.172	**0.067**
TT Vibrates	0.096	0.185	0.143	0.247	**0.084**
TI Vibrates	0.083	0.110	0.123	0.186	**0.078**
Gravity					
Both Vibrate	0.500	0.493	0.500	**0.488**	0.492
TT Vibrates	0.499	**0.484**	0.500	0.486	0.490
TI Vibrates	0.503	**0.494**	0.500	0.500	0.497
Gyroscope					
Both Vibrate	0.003	**0.001**	0.009	0.014	0.003
TT Vibrates	**0.001**	**0.001**	0.003	0.003	**0.001**
TI Vibrates	**0.001**	0.003	0.006	0.006	0.002
Linear Acceleration					
Both Vibrate	0.015	0.017	0.042	0.047	**0.014**
TT Vibrates	0.016	**0.015**	0.038	0.044	0.016
TI Vibrates	**0.002**	**0.002**	0.020	0.009	0.003
Magnetic Field					
Both Vibrate	0.059	0.315	0.187	0.477	**0.031**
TT Vibrates	0.065	0.202	0.132	0.291	**0.035**
TI Vibrates	0.146	0.236	0.149	0.409	**0.138**
Rotation Vector					
Both Vibrate	0.008	0.008	0.031	0.113	**0.007**
TT Vibrates	**0.002**	0.010	0.014	0.106	**0.002**
TI Vibrates	0.016	**0.005**	0.024	0.039	0.013

10 times was performed (total of 100 estimates). The machine learning algorithms that were used as part of this evaluation are, namely, Random Forest, Naïve Bayes, Decision Tree, Logistic Regression, and SVM.

The EERs were computed as discussed in Section 3.2, by determining thresholds where the FAR was equal to the FRR for each tested sensor. A more detailed explanation of the machine learning analysis is provided in Sections 3.2 and 3.3.2.

The individual differences $|A_{i,j} - B_{i,j}|$ (where $A_{i,j}$ refers to the j^{th} data-point of the i^{th} sensor measurement on device A) were used as attributes that describe each pair of transactions when training and testing each machine learning model. Thus, as transactions were sampled at 5ms intervals, a total of 100 numeric features existed in each example for training and testing the machine learning model. If an example corresponded to a legitimate transaction, it was labelled as positive, otherwise it was labelled as negative.

Weka's default settings were used for each of the first four machine learning algorithms that were mentioned earlier (also in Table 6.2). For the fifth algorithm, SVMs optimised with the SMO algorithm were used. The complexity parameter C

and the width of the RBF kernel γ were tuned using a grid search by optimising AUROC as estimated using internal cross-validation on the training data. For all five algorithms, the settings that were used are the same as the ones used in the machine learning analysis of Chapter 3.

Table 6.2 lists the results obtained from the different machine learning algorithms. These results were obtained by averaging estimated EERs across all the relevant attack scenarios from Table 6.1. The best result for each sensor is shown in bold. The results for individual attack scenarios can be found in Appendices D (both devices vibrating scenario), E (terminal only vibrating scenario), and F (instrument only vibrating scenario). For each vibration mode, sensor, and learning algorithm, these tables show the mean and standard deviation of the 100 EER estimates obtained using 10-fold cross-validation repeated 10 times.

Table 6.2 shows that SVMs produce the lowest EER for most sensors/vibration modes. Random forests and Naïve Bayes also demonstrated low EERs for many sensors/vibration modes. The results from the machine learning experiments indicate that all sensors apart from the gravity sensor provide useful information for the discrimination between legitimate and distant (relay) transactions. Ranking the sensors based on discriminative power (i.e. the EER) when evaluated in conjunction with SVM classifiers yields the following ranking (best to worst): gyroscope, rotation vector, linear acceleration, magnetic field, accelerometer, and gravity. The same ranking holds for random forests. Based on Naïve Bayes, the ranking is: gyroscope, rotation vector, linear acceleration, accelerometer, magnetic field, and gravity. Finally, none of the vibration modes seems to be performing substantially better than the others, except in the cases of magnetic field, where the TI only vibrating performed substantially worse than the rest, and the case of accelerometer, where TT only vibrating performed substantially worse.

6.5 Discussion and Outcome

The analysis of the experimental data indicates that the effectiveness of the proposed solution as a PRAD mechanism is high. Using most sensors, the empirically observed relay attack detection rate is higher than that of previously proposed solutions [68, 116, 139, 153]. No particular attack against the proposed solution performs better than the others. However, some attacks perform better than others depending on the classifiers and sensors involved. The performance of the solution is not significantly affected by using any of the evaluated attacks.

Even though a different smartphone was used as device TT', no significant impact was observed in the results. Observing the results of TI only vibrating, where TT''s impact in the transaction was negligible, except in the case of using the magnetic field sensor, no significant degradation in the effectiveness of the proposed solution was observed. However, further investigation might be required before deployment in

the real world. More aspects should also be examined, like the impact of protective smartphone cases on the proposed solution.

Moreover, even though no significant performance differences were observed between different vibration modes, in order to prevent acoustic attacks by a more resourceful attacker, both devices vibrating might be preferred. Observing the vibrated pattern has been demonstrated with relatively high success rate in [67], so it might have some effectiveness against the proposed technique. Fusing vibrations from both devices can be a significant barrier against off-the-shelf attackers, as explained in Section 6.2.2.

No additional or non-standard hardware on the TI side is likely to be required, like in many other proposed solutions, as the one presented in Chapter 4 (Infrared as an AAE actuator) and [139]. The examined sensors are available on a large variety of smartphones, and most modern smartphones will be equipped with at least one of them.

In order for the proposed solution to be deployed on a large scale, the transaction terminals should be equipped with the sensor that will be used to measure the vibration. The sensor should be attached on a flat surface, on which devices will be placed on. Some usability implications might arise, for example in cases that the terminal is not steady. Also, since the vibration is 500ms long, the comparison of the captured values cannot be performed during the time of the transaction. The length of the vibration might be a barrier for shorter transactions, like in the case of transportation-related transactions that require to complete within 300ms. Further investigation is required in order to evaluate the effectiveness in such transactions. This solution might be more relevant to contactless payments, access control, and 2FA.

Finally, as already discussed, many of the previously proposed solutions might also be vulnerable in the presence of a context manipulating attacker. Since this solution is not dependant upon the surrounding environment, such attacks do not apply, unless the attacker can physically tamper with the devices.

6.6 Summary

Communicating devices are vulnerable to relay attacks. Traditional distance bounding protocols that aim to counter such attacks might not be applicable in the field of smartphones. Alternative approaches against off-the-shelf attackers have been proposed, mostly based on sensing of the ambient environment. However, these might not be suitable or secure under certain scenarios, like in the case of transactions with industry imposed time restrictions of up to 500ms (e.g. EMV and transportation related transactions). The generation of a random bit/stream-based AAE by peripherals of the transaction devices has demonstrated promising results as a PRAD mechanism when using infrared light as an AAE actuator, as seen in chapters 4 and 5.

In this chapter, the use of vibration as an AAE actuator is investigated. During the transaction, one (unidirectional) or both (bidirectional) transaction devices vibrate some random pattern, which is measured by a selected ambient sensor by both transaction devices. The two measurements are then compared for similarity, in order to establish proximity evidence.

A test-bed was designed and built for the evaluation of the proposed solution as a PRAD mechanism. Three different vibration modes (only one of the two, or both the transaction devices vibrating a random pattern) were evaluated, against five different attack scenarios. Six ambient sensors were evaluated. A total of 36,000 genuine–relay transaction pairs, based on field trials, were collected or synthetically generated. Analysis using five machine learning algorithms was performed, indicating high classification accuracy between genuine and relay transactions on all vibration modes, and using most of the sensors.

Part IV

Non-Ambient Environment-Based Relay Attack Detection

Chapter 7

Force-Based Relay Attack Detection

7.1 Introduction

In the previous part of this thesis, the generation of an AAE as a PRAD mechanism was investigated. In the remaining chapters, alternative approaches of relay attack prevention are discussed, that are not based on the ambient environment.

In this chapter, a novel approach towards PRAD is proposed, based on sensing button presses on the user's smartphone by both transaction devices (transaction terminal and transaction instrument, i.e. the user's smartphone) simultaneously. To achieve that, the transaction terminal should be equipped with a force-sensitive panel, capable of detecting the coordinates where pressure is being applied on it. During the time of the transaction, the user is requested to place the transaction instrument (smartphone) on the force-sensitive panel and press a set of buttons that are presented on the smartphone's display. The input button sequence, as well as timings of button presses and between subsequent button presses (referred to as *'releases'*) are captured by both transaction devices and used as features for similarity comparison. The concept behind this approach is that the two transaction devices that need to establish proximity evidence will record approximately the same timing measurements of the corresponding button presses and releases, in case they are in contact with each other. For an attacker to perform a successful attack, replication of the genuine user's movement when performing the button presses/releases will have to be accomplished with accuracy of a few milliseconds.

Empirical evaluation of the proposed technique was performed in two phases. First, the effectiveness of the proposed technique as a mechanism to provide proximity evidence was evaluated. Afterwards, a set of volunteers attempted to perform a relay attack multiple times by trying to replicate the movement of a genuine user. High success rate of the proposed method as a PRAD mechanism was observed. Using threshold-based evaluation, all the attack attempts were detected. Perfect classification was also achieved by using the SVM classifier. Near-perfect classification (up to 99.8%) was achieved by other well-known machine learning algorithms.

7.2 Force-Sensing PRAD

In this section, the theoretical foundation of the proposed framework, and the threat model are presented.

7.2.1 PRAD Framework

During the course of a transaction, the user is called to position the Transaction Instrument (TI) on an extension force-sensitive panel of the Transaction Terminal (TT), in order to complete the PRAD procedure. A smartphone application (running on TI) presents to the user an interface with buttons that the user is called to press on. Device TT uses the force-sensitive panel to detect the location of the presses on device TI, that is then translated to the corresponding button of device TI. The detection is based on the coordinates at which the highest amount of force/pressure is being applied on the panel.

Regarding the smartphone application, alternative interface design approaches can be followed. For example, the application might present buttons with numbers and ask the user to input a provided random 4 digit sequence. Alternatively, the application can present a button at a time that the user has to press in order for the next button to appear, until all the buttons of a random sequence have been pressed. It should be stressed that even though the aforementioned method was used during the evaluation of the proposed framework (see Section 7.3), it is not restrictive, as alternative approaches can be used instead. Figure 7.1 presents the basic architecture of the framework, when using the later application design method. Also, the use of a PIN and other forms of user identification codes may pose a threat, as an attacker could potentially capture them if the user attempts to perform a transaction with a malicious terminal.

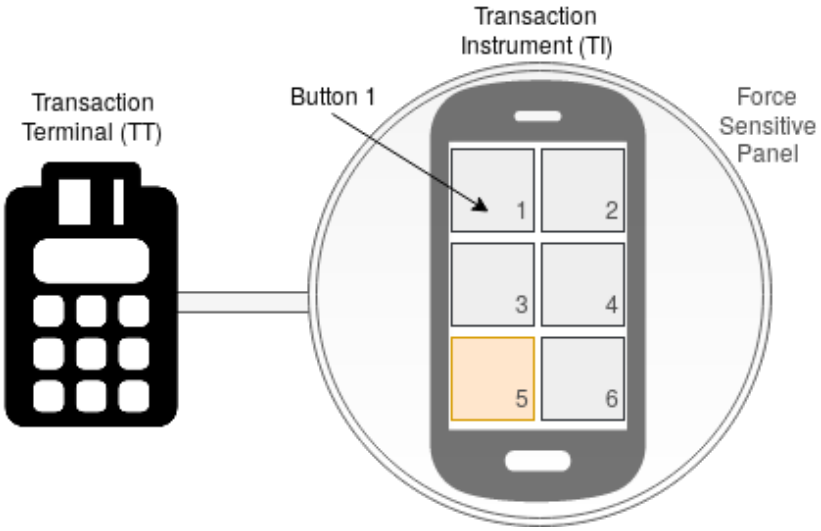


Figure 7.1: The Basic Framework Architecture

Each button is assigned an ID, based on the location of the button (e.g. the top left button is assigned the ID *'1'*). When the user presses a button, both transaction devices detect and record the ID of the button that was pressed, and the duration of the button press. For the correct detection of the buttons that are pressed on device TI, by device TT, the first should be placed on on top of the former, with its screen facing upwards. The position should be such that each button of device TI is aligned with the area that will be translated to the correct button by device TT. The duration between subsequent button presses is also recorded by the two devices. To summarise, three elements are recording by both devices simultaneously:

1. The sequence of the IDs of the pressed buttons (e.g. *'1234'*)
2. The duration of each button press (in *ms*)
3. The duration between subsequent button presses (in *ms*)

Table 7.1 presents an example of the recorded features. Further features can also potentially be used, like the amount of pressure applied, but were not considered in this work due to limitations inflicted by the architecture of the majority of modern smartphones (further discussion in Section 7.6).

Table 7.1: Example of a Record (all values represent time in *ms*, except from the *'Input Sequence'*)

Feature	**Value**
Input Sequence	5132
Press 1	117
Release 1	1434
Press 2	118
Release 2	1294
Press 3	178
Release 3	1634
Press 4	117

In order for device TI to recognise the buttons and timings, simple API calls are required. Device TT can recognise the buttons and timings based on the coordinates and duration of the detected pressure on the force-sensitive panel.

As already mentioned, the assumption is that the two devices are going to record approximately the same values (button sequence and durations of presses and releases), and that an attacker has a low probability of accurately replicating the movement of a genuine user. The accuracy of the system should be at the millisecond level. The captured data of the two devices should provide sufficient proximity evidence when compared against each other, while data captured when a relay attack is taking place should be detectable at a high rate, in accordance to the requirements of the deployment scenario.

The proximity verification process can take place during the course of the transaction by one of the communicating devices, or afterwards, by a TTP. The captured

data should be communicated between the devices or to the TTP in an encrypted and authenticated form, but discussion regarding the trusted comparison party's (i.e. the TTP or one of the transaction devices) architecture is out of the scope of this chapter. As the main focus of this chapter is to examine the effectiveness of the proposed solution as a PRAD mechanism, the discussion is limited and the integration with existing applications is not investigated.

7.2.2 Threat Model

In line with the previous chapters of this thesis, the attacker is considered to be of opportunistic nature and requires no prior interaction or knowledge of the transaction devices TT and TI. Potential implications that might arise in case the transaction devices are compromised are out of the scope of this chapter. The focus is on providing proximity assurances on genuine devices whose operational requirements mandate that they are co-located in order to perform a transaction.

The attacker is also assumed to only have access to off-the-shelf relay equipment. As already discussed, limits typically apply on smartphone-based contactless transactions, like for example the £30 limit on payment transactions in the UK. Moreover, an attacker that might be using more advanced techniques, like a robotic arm that replicates the movement of a genuine user with accuracy, is likely going to be detected by the genuine device operators. Therefore, a very powerful attacker might not be a major concern.

7.3 Test-bed Architecture

A test-bed was designed and built, in order to evaluate the proposed solution. Android devices were used to represent TI, and an Arduino-based prototype was developed and used as TT.

An Android application was built and installed on three devices; a light and small Android smartphone, a heavier and larger Android smartphone, and an Android tablet device. As the first, an SGS5 mini (SM-G800F) was used. It features a 4.5-inch display, and weighs 120 grams. An SGS4 (GT-I9500) device was used as the second device. It features a 5-inch display and weighs 130 grams. Finally, a Nexus 9 was used as the tablet. Its display is 8.9-inches and weighs 425 grams[6].

The Android application, running on TI, featured six equal sized zones on the touch screen display (as in Figure 7.1). These zones are referred to as *'buttons'*. Initially, a random button of the six was highlighted on the screen, using a colour. This indicated to the user to press the randomly highlighted button on the screen. Releasing a button would trigger it to disappear, and the application would randomly pick and highlight a second button. A total of four buttons were randomly highlighted for each transaction. A random shared ID assigned to each transaction, along with the

[6]All device characteristics found at http://www.gsmarena.com/

IDs of the captured button sequence, and the timings of button presses and interval duration between subsequent button presses were appended to a local CSV file. That file was later extracted from the device for data comparison.

An Arduino Due was used for the TT prototyping. Four Force Sensitive Resistors (FSRs) were connected to the board's analogue inputs (Figure 7.2a). FSRs can be used to detect pressure or weight. The basic working principle is that the resistive value of the sensor alternates depending on the amount of force that is being applied on the sensor. Silicone buffer pads[7] were manually attached to the centre of the sensors, as the force resistive area of the FSRs was not reachable by the surface of the smart devices otherwise. Even though FSRs were used in this prototype, other sensors might potentially also be used alternatively, for example capacitive touch sensors.

Initially, when TI was placed on TT and the process started, a calibration phase was conducted, in order to 'cancel out' the weight of the device. For one second after the process initiation, TT would capture values from all four sensors and the maximum recorded value of each sensor would then be used as a reference point. A LED would indicate the completion of the calibration phase. Algorithm 1 describes the calibration process in more detail. As input (*sensorReading1*, etc.), the algorithm takes the values that are being measured by each of the four sensors, and returns an array (*sensorMaxVal*) that contains the maximum recorded values for each of the sensors.

Algorithm 1: Sensor Calibration of Device TT

Input : int sensorReading1, int sensorReading2, int sensorReading3, int sensorReading4
Output: int[] sensorMaxVal

```
1  sensorMaxVal[] ← {0, 0, 0, 0};
2  startTime ← getCurrentTimeMillis() ;    // gets the current time in ms
3  while getCurrentTimeMillis() - startTime ≤ 1000 do
4      this.updateSensorReadings() ; // reads current sensor measurements
5      if sensorReading1 > sensorMaxVal[0] then
6          sensorMaxVal[0] ← sensorReading1;
7      end
8      if sensorReading2 > sensorMaxVal[1] then
9          sensorMaxVal[1] ← sensorReading2;
10     end
11     if sensorReading3 > sensorMaxVal[2] then
12         sensorMaxVal[2] ← sensorReading3;
13     end
14     if sensorReading4 > sensorMaxVal[3] then
15         sensorMaxVal[3] ← sensorReading4;
16     end
17 end
18 return sensorMaxVal
```

[7]Example of buffer pads: https://www.amazon.co.uk/gp/product/B00P11D4VK/ref=s9u_simh_gw_i2

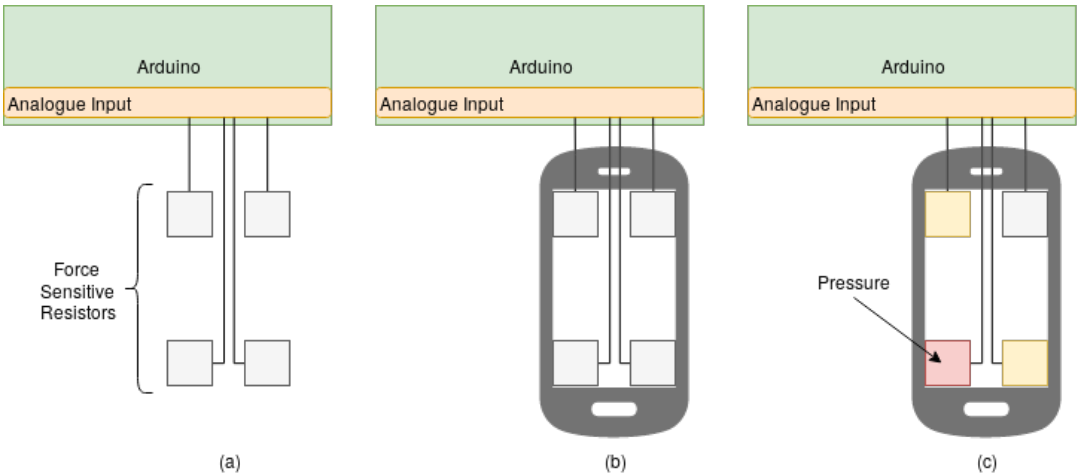


FIGURE 7.2: Detection of Presses by TT

After the calibration phase, the user was called to input the sequence on device TI. While a button was being pressed, the calibrated TT sensor(s) closest to that button was/were recording the highest force values (Figure 7.2). The indication used for detecting that a button was being pressed was that some of the sensors were recording values above their calibration point. The time during which higher values than the calibration point were being recorded by some of the FSRs was considered to be the pressing time. Similarly, the time between subsequent button presses was the time during which none of the sensors were recording higher values than the calibration threshold.

After a button was released, TT would estimate the ID of the button that was being pressed, based on the sensor recordings. To achieve this, each sensor was assigned a value, which was the average of all the values captured by that particular sensor during the button press period. Six virtual buttons had to be detected, corresponding to the buttons presented by TI. For the detection of the pressed button, the values that were assigned to each of the four sensors were used.

Algorithm 2 lists pseudocode for detecting the pressed buttons when the button ID numbers of device TI are as per Figure 7.1, and the sensor ID numbers of device TT as per Figure 7.2. The input *sensor1*, *sensor2* *sensor3*, and *sensor4* is the value assigned on each of the four sensors. Initially, the side (left or right) of the screen on which the press was on was determined by comparing the sum of the sensor values returned from each side. Once the side was determined, the proportion of the force applied on the top sensor was calculated. The proportion was divided in three equal parts. If the proportion was between 0.66 and 1, the top virtual button ID was returned. Similarly, values between 0.33 and 0.66 corresponded to the middle virtual button, and between 0 and 0.33 to the bottom one.

When four button presses were detected by the prototype, it would return the pressed button sequence and the timings to a connected computer, through Arduino's serial port. An application developed in Python would request from the experiment operator to manually input the unique ID that was assigned to the

Algorithm 2: Detection of Pressed Button

```
Input  : int sensor1, int sensor2, int sensor3, int sensor4
Output : int pressedButtonID
leftSide ← sensor1 + sensor2;
rightSide ← sensor3 + sensor4;
if leftSide > rightSide then
    force ← sensor1 / leftSide;
    if force ≥ 0 and force < 0.33 then
        return 5
    else if force ≥ 0.33 and force < 0.66 then
        return 3
    return 1
end
force ← sensor3 / rightSide;
if force ≥ 0 and force < 0.33 then
    return 6
else if force ≥ 0.33 and force < 0.66 then
    return 4
return 2
```

particular transaction by TI. The returned values and the transaction ID would be appended to a CSV file with the same format as the one on TI's side.

Two experimental frameworks were developed, for proximity and relay attack evaluation. At the end of each phase, the stored data would be extracted from each of the mobile devices and moved to a computer for the evaluation of the results.

7.4 Proximity Detection Framework

In order to evaluate the performance of the proposed solution in a proximity detection scenario, 100 transactions were performed with each of the three devices listed in Section 7.3. A random sequence was generated and presented by TI on each transaction. The FSRs were aligned in a set-up for the smallest device (SGS5 mini), and this set-up was maintained throughout the experimental phase, regardless of the device being used. After the completion of the first phase, using each of the three TI devices, the collected data in CSV format was extracted from the mobile devices and transferred to a computer for comparison. The data from the Arduino was already stored on the computer (passed through the serial port, as described in the previous section).

The aim of the framework was to assess whether sufficient information for establishing proximity evidence is collected during the process. The results were used to set acceptable upper and lower bounds for presses and releases.

7.4.1 Evaluation Methodology

A Python application was developed for the data analysis process, which would compare the input sequence recorded by both devices, and then calculate the difference of individual corresponding features (button press and release timings). The minimum and maximum press and release differences were detected for each of the devices, and for all 300 measurements combined. The minimum and maximum press and release differences were used as limits for distinguishing genuine from relay attack transactions in the relay attack detection framework (in the threshold-based evaluation — see Section 7.5.2).

7.4.2 Results and Discussion

The results of the proximity detection framework are listed in Table 7.2. The results refer to the time difference in milliseconds between the timings recorded by TT and TI. For example, the difference of the first press for a single transaction is calculated as:

$$Diff_{Press1} = TT_{Press1} - TI_{Press1}$$

The maximum, minimum, and the average calculated values are listed in the table, as well as the span between the maximum and minimum observed values. The analysis has been performed for each of the three devices, as well as for the combination of all the recordings of the three (referred to as *'Total'*). Negative timings denote that the measurement of a press or release by TI was longer than that of TT.

Table 7.2: Proximity Detection Results — in *ms* (negative results indicate that TI's measurement durations were larger than TT's)

	Minimum		Maximum		Span		Average	
	Press	**Release**	**Press**	**Release**	**Press**	**Release**	**Press**	**Release**
SGS5 mini	-46	11	-11	48	35	37	-29.09	28.71
SGS4	-28	-8	9	30	37	38	-5.32	5.26
Nexus 9	-18	-8	12	23	30	31	-2.16	2.83
Total	-46	-8	12	48	58	56	-12.19	12.27

Differences among the devices were observed, likely related to their weight. However, the measured time span was approximately the same for all three devices. Therefore, a real-world deployment is recommended to take into account the device model, in order to minimise the attack window approximately by half.

High accuracy was also ascertained in the detection of the input pattern by device TT. Prior to the initiation of the experimental phase, the pattern detection of the smaller devices was found to be very high. Failures occurred in a few occasions where the buttons on device TI were pressed very close to neighbouring buttons. Slightly reducing the size of the buttons effectively restricted this issue.

The detection of button presses on the tablet device required longer presses than on the smartphones, which had no special input requirements. Even though perfect

detection was recorded when longer button presses were applied, the use of tablets is not recommended without readjustment of the underlying FSRs.

7.5 Relay Attack Detection Framework

The effectiveness of the proposed solution as an anti-relay mechanism was subsequently evaluated. A set of 10 volunteers was gathered, who tried to attack the system in two phases, explained below. A separate Android application was developed for the purposes of the experiments, based on the application used in the proximity evaluation framework. The application was presenting predefined sequences instead of random ones, so that all volunteers tried to attack the same sequences. Device TT's architecture did not alter between the two frameworks (proximity and relay attack detection).

During the first attack phase, a set of videos of a person inputting a sequence (genuine user) were presented to the volunteers. The sequence timings entered by the genuine user on TI were stored in a CSV file as per Section 7.4 for later comparison against relay attack data. The videos were played on a large screen (24-inch), and the movement of the genuine user was clear and evident, according to the volunteers. The camera used to record the videos was placed on the left of the device, at a 45 degree angle, in order to make the presentation of the three dimensions more clear. Prior to the initiation of the experiments, the volunteers were asked to choose the rotation and flip of the video that they preferred. A set of 10 different patterns were presented to each volunteer, who was asked to attempt to replicate/mimic the movement of the genuine user with accuracy while the video was playing, or afterwards. The same pattern was presented on both the video and the device provided to the volunteers. Also, the same device (SGS5 mini) was used on the video and by the volunteers. Data from both devices was stored in separate CSV files, separately for each volunteer.

All volunteers were university students and stuff who had a good understanding of security and relay attacks. Prior to the experiments, their goal and the principle on which the anti-relay mechanism was based were thoroughly explained to them. Moreover, four of the volunteers had background in playing some musical instrument (guitar, piano, or both), ranging from medium to advanced level.

During the second phase of the experiment, the volunteers were asked to try to attack the exact same video 10 times. Meaning, a genuine user entered a sequence which was recorded and then the volunteers were given 10 tries to try to replicate the same sequence as close to the genuine user as possible. The volunteers were able to watch the video of the genuine user inputting the sequence, while attempting their attacks. This tested the possibility of whether an attacker who watches the same pattern being entered a number of times, their potential of replication would increase. The same set-up as in the previous phase was used. In both phases the attacker was powerful, as the input sequence was known, and there was a clear view of the genuine user's movements.

7.5.1 Evaluation Methodology

Measurements captured from the device TI that the genuine user was using on the video were compared against measurements from device TT used by the volunteers. As per Figure 4.1, the pair (TI–TT'), where TT' was the terminal device used in the video, acted as the proximity pair during the analysis (i.e. no relay attack involved). The pair (TI–TT) acted as the distance pair (i.e. during a relay attack). The transaction instrument operated by the volunteers is referred to as TI'.

In order to evaluate the performance of the volunteers, two methods were used; threshold- and machine learning-based analyses. Initially, acceptable minimum and maximum thresholds for presses and releases were set, based on the results of the proximity evaluation framework (Section 7.4.2). The recorded presses and releases from the two devices were sequentially compared against each other. If the difference between a press or a release captured by TT and TI was within the bounds, it was considered to be acceptable. Otherwise, a relay attack was detected. The point at which the inconsistency appeared was the detection point. For example, if an attacker performed the first press and the first release correctly, but the second press was out of bounds, the second press would be considered as the detection point. Figure 7.3 depicts a typical scenario in which the volunteer fails to replicate the movement of the genuine user with enough accuracy. The volunteer's attempt was recorded by devices TT and TI', while device TI represents the corresponding measurement of the genuine user. The comparison that took place was between devices TT and TI. Device TI' is also shown for reference purposes. The attempt could be detected at the first press, as the attacker held the button for a shorter period of time than the genuine user. This part of the evaluation was conducted in two phases. The relay attack data was first subjected against the thresholds set by combining all three devices, and then against device specific thresholds (SGS5 mini).

Weka was used to apply a suite of well-known classifiers (same as the ones used

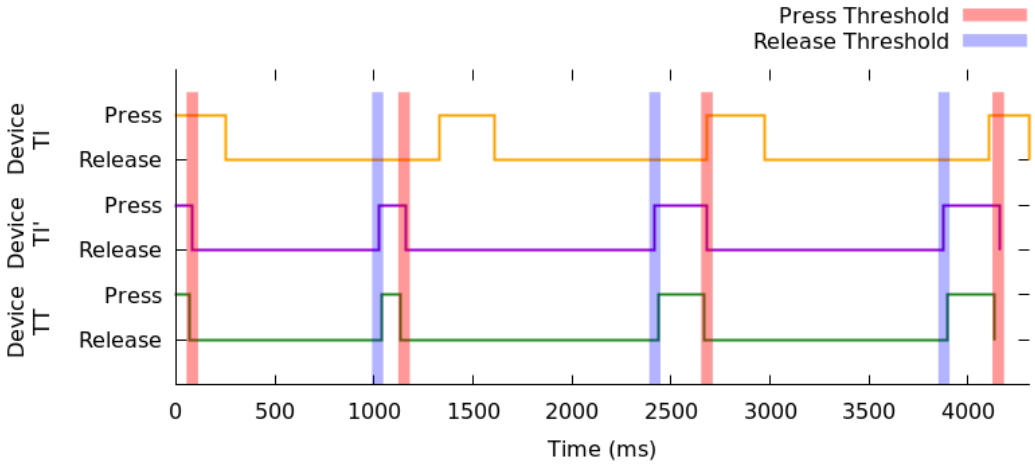


Figure 7.3: Graphical Representation of a Typical Attack Attempt (the device topology is as in Figure 4.1 and device TI' is displayed here for reference purposes)

in previous chapters). The classifiers were trained on a set of feature vectors with corresponding binary labels (genuine or relayed transaction), which were collected beforehand. The trained model was used to classify subsequent transaction data streams as genuine or relayed. The Random Forest, Naïve Bayes, Logistic Regression, Decision Tree (grown using the C4.5 algorithm), and SVM with the RBF kernel (the SVM with RBF hyper-parameters, C and γ, were established using standard exhaustive grid search) classifiers were tested. The threshold was based on the probability estimate output by the learned classification model, i.e. the estimated probability that a transaction is genuine. As genuine transactions were also considered transactions collected during the proximity evaluation phase (Section 7.4). In this chapter as well, the same configuration as the one listed in Chapter 3 was used for the machine learning-based analysis.

As already discussed in Chapter 3, in order to assess the accuracy of Weka, it was benchmarked against the Python library `scikit-learn`. The same algorithms and parameters as in Weka were used. The results that `scikit-learn` produced were exactly the same as Weka's.

7.5.2 Results and Discussion

A total of 200 relay transactions were evaluated. The results of both threshold- and machine learning-based analyses are presented in this section.

Evaluation 1: Threshold-Based

The results of the threshold-based analysis are listed in Table 7.3. *'General Threshold'* refers to using the threshold set by combining the proximity results of all three tested devices, while *'Device Specific Threshold'* to using the threshold set by the SGS5 mini, as it was the device used for the relay attack detection evaluation. *'Detected'* refers to the percentage of relay attempts detected at a particular press or release, because the attackers failed to accurately replicate the movement of the genuine user at that point. *'Correct'* refers to the percentage of times that a single press or release was successfully replicated by the attacker. Finally, the first phase refers to the user trying to attack a different video on each try, and the second phase, the attacker trying to attack the same video 10 times.

None of the volunteers were able to successfully attack the system when using threshold-based analysis. Moreover, the volunteers with background in playing a musical instrument did not present improvements over the rest of the users. However the sample was limited, so further investigation might be required.

The best attempt is presented in Figure 7.4. Device TI' in the figure represents the measurement recorder by the transaction instrument that the volunteer was using. It illustrates the corresponding proximity transaction, for which all presses and releases were within bounds. The relay attack was detected on the third press, using the

Table 7.3: Threshold-Based Relay Attack Detection Results

	Press 1	Release 1	Press 2	Release 2	Press 3	Release 3	Press 4	False Accept
General Threshold – Phase 1								
Detected	73	24	0	2	1	0	0	0
Correct	27	10	37	10	24	5	31	—
Device Specific Threshold – Phase 1								
Detected	84	16	0	0	0	0	0	0
Correct	16	7	25	9	9	4	23	—
General Threshold – Phase 2								
Detected	57	37	5	1	0	0	0	0
Correct	43	11	29	24	26	9	20	—
Device Specific Threshold – Phase 2								
Detected	65	33	2	0	0	0	0	0
Correct	35	6	21	22	20	6	9	—

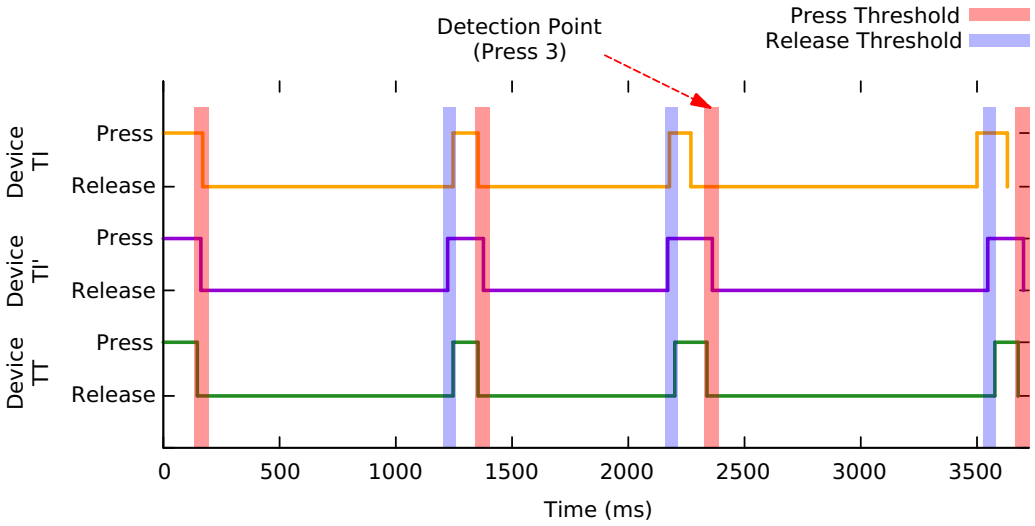


Figure 7.4: Graphical Representation of the Best Attack Attempt (the device topology is as in Figure 4.1 and device TI' is displayed here for reference purposes)

threshold set by the combination of the three devices. However, using device specific threshold, the attack was successfully detected on the first release (Figure 7.5).

The majority of the transactions were detected on the first press or release in both cases, i.e. both when using general and when using device specific thresholds, even when the volunteers tried to attack the same video multiple times (second phase). During the second phase, a small incline was observed in the user performance as the same video was being attacked multiple times, most evident in the performance of releases. Figure 7.6 depicts that incline, along with the average, maximum, and minimum performance of each round's presses and releases.

Evaluation 2: Machine Learning-Based

The results of the machine learning-based analysis, obtained by repeating stratified 10-fold cross-validation, are presented in Table 7.4. A training set of 300 genuine

(including the transactions from the proximity detection evaluation — Section 7.4) and 200 relay attack transactions was used. The same settings for each algorithm, as the ones used in Chapters 3 and 6 were used on Weka. The metrics listed are the classification accuracy *(Accuracy)* (see Eq. 3.3), the Area Under the Receiver Operating Characteristic (ROC) Curve *(AUC)*, the *F1-score* (see Eq. 3.1), and the *EER*.

Perfect classification was achieved by using the SVM classifier. Near perfect classification (> 98%) was also achieved by the Random Forest, Naïve Bayes, and Decision Tree classifiers, with the best performance observed by the first three (> 99.5% accuracy). The Random Forest algorithm failed to accurately classify two relay transactions, the Naïve Bayes one, and the Decision Tree six.

TABLE 7.4: Machine Learning Classification Results Obtained by Repeating 10-Fold Cross-Validation 10 Times

	Random Forest	**Naïve Bayes**	**Logistic Regression**	**Decision Tree**	**Support Vector Machine**
Accuracy (%)	99.62	99.80	86.58	98.78	100.00
F1-score	0.9969	0.9984	0.90	0.9901	1.0
EER	0.0022	0.0047	0.1993	0.104	0.0

7.6 Discussion and Outcome

The analysis of the experimental data indicated the high effectiveness of the proposed solution as a PRAD mechanism. However, some usability concerns might arise, especially for visually impaired users, and users with motor difficulties. Compared to

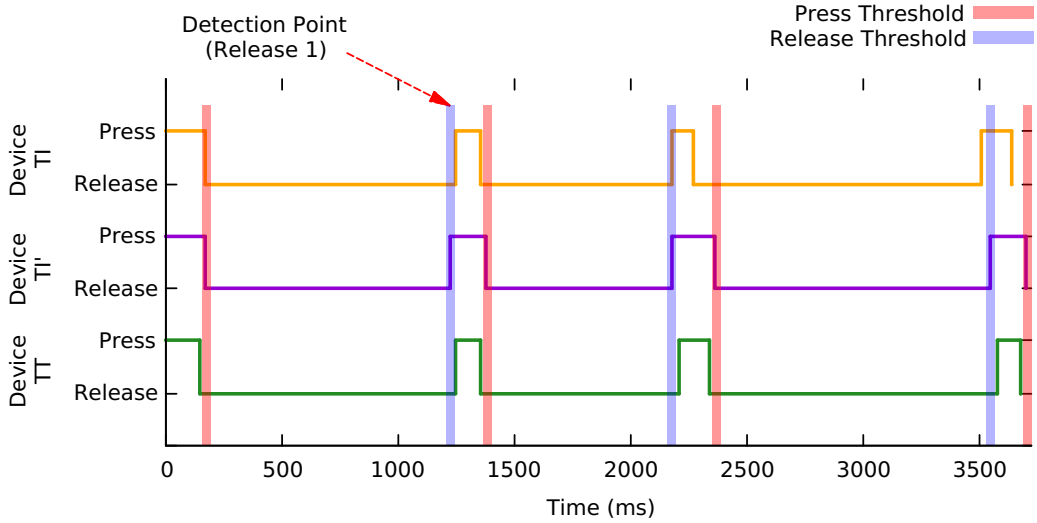


FIGURE 7.5: Graphical Representation of the Best Attack Attempt When Device-Specific Thresholds Apply — In This Occasion: SGS5 mini (the device topology is as in Figure 4.1 and device TI' is displayed here for reference purposes)

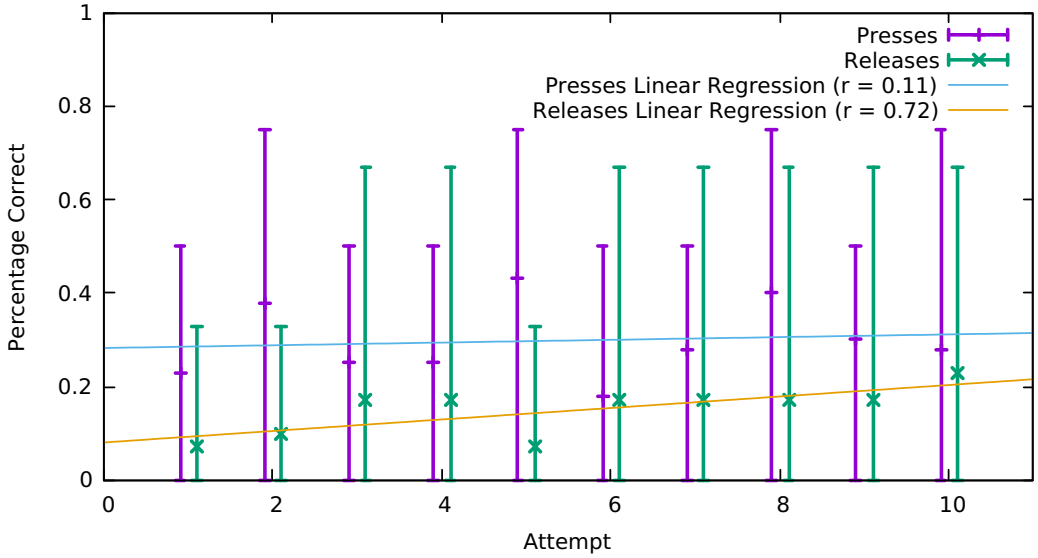


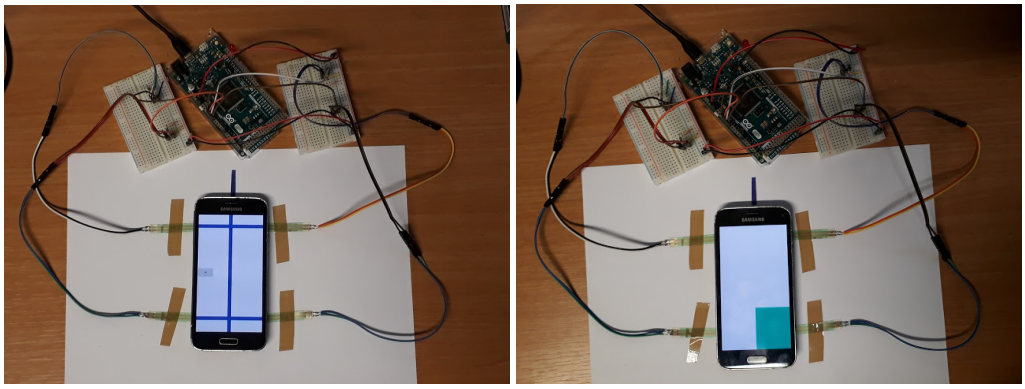
FIGURE 7.6: Performance Variation of Each of the 10 Attempts in the Second Phase

the majority of previous works, additional steps are required by the user, including the correct placement of device TI on TT.

Since the positioning of device TI on TT is important for the latter to accurately detect the ID of the pressed button, device-specific positioning lines were presented on the TI's display prior to the process initiation. This assisted in more equally dividing the mass across the four FSRs. The model of the device could be determined through API calls, which was used to display the correct positioning. Figure 7.7 depicts the guidelines and the pattern input interface on the SGS5 mini device. It should be stressed that the positioning precision had to be performed with only some degree of accuracy, meaning that perfect positioning of the devices was not necessary. The detection was found to be very accurate even without perfect placement of the devices. The impact of not correctly aligning the two devices may vary, depending upon the amount of offset between the location of the user's presses on device TI and the expected coordinates range of the corresponding button on TT. In case there is offset between the devices, and force is applied by the user outside of the detection range for a particular button by device TT (as seen in Figure 7.8), the former might not detect the correct button.

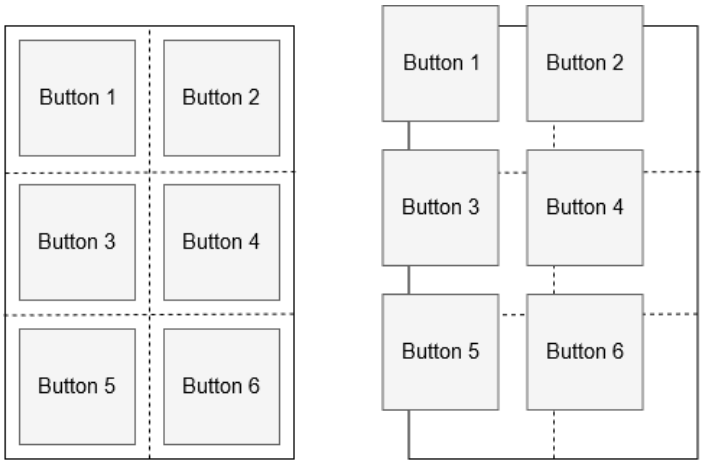
Failure to press on the correct button (mis-pressing) may also lead to inconsistencies between the captured data of the two devices, so the process will have to be restarted. Moreover, a vibration of the phone during the process may cause inconsistencies, so it should temporarily be disabled, until the completion of the process. Finally, the performance of the solution might also degrade if phone cases are used on TI, or the device's camera is located above the point where an FSR should touch. However, significant advantages over previous works exist, so depending on the deployment scenario, this technique might be preferable.

The relay attack detection rate was empirically found to be higher than many



(A) Device Calibration (B) Pattern Entry

FIGURE 7.7: The Evaluation Test-bed



a) Correct Placement b) Offset

FIGURE 7.8: Button Alignment Between TI and TT During Correct and Incorrect Device Placement (grey squares represent buttons of device TI, while white squares represent the corresponding TT coordinates range)

previously proposed solutions [68, 116, 139, 153]. The detection rate can potentially improve even further by considering more features, like the amount of pressure detected by the two devices. Even though this is possible on TT, as the FSRs are capable of detecting the amount of pressure that is being applied on them, at the moment, the majority of smartphones are not capable of accurately measuring the amount of pressure that is being applied on their screen. Some Android devices estimate the amount of pressure through the number of pixels being covered on the screen by the user's finger. This technique is not very accurate, and it is also not available on all devices. The feature is not supported by the SGS4. On the other two devices, the accuracy was found to be highly dependent on the finger orientation rather than the amount of pressure being applied on the screen. Moreover, inconsistencies were observed among the pressure values recorded by the two devices. It was concluded that the technology was not mature enough to provide quality data.

Moreover, no additional or non-standard hardware on the TI side is required (like

for example in the case of Chapter 4). Finally, this solution is not dependent upon the surrounding environment, therefore it is not vulnerable to attacks that require the presence of an attacker with context manipulating capabilities, as described in [138], unless the attacker is capable of physically tampering with the devices.

The usability issues associated with this solution might make it more relevant to contactless payments and access control in scenarios where high throughput of customers is not a requirement, as well as 2FA. However, the high effectiveness in combination with not requiring extra hardware on the transaction instrument's side pose clear advantages. Moreover, since the comparison of the collected data can be based on simple comparisons, it might be feasible to be deployed in offline scenarios, where the comparison is performed by one of the transaction devices.

7.7 Summary

In this chapter, a novel approach for PRAD was presented, by using bidirectional sensing and comparing of subsequent button presses and releases by the transaction devices. Unlike the PRAD approaches that were proposed and/or evaluated in the previous chapters of this thesis, this approach does not rely on the generation or sensing of the ambient environment. Instead, it is based on sensing of user-performed actions. This approach is meant to be used as part of the transaction process, like the PIN entry.

For the evaluation of the proposed technique, a test-bed was designed and built, that consisted of two phases. Initially, the effectiveness in proximity detection was examined. Afterwards, the test-bed was subjected against a set of volunteers who tried to attack the system. All the attack attempts were successfully detected through threshold-based analysis. Moreover, perfect classification was achieved by using the SVM classifier. Classification accuracy of up to 99.8% was achieved by other well-known machine learning classifiers.

Chapter 8

Host-Based Relay Attack Detection

8.1 Introduction

While the previous chapters of this thesis focused mainly on PRAD, assuming that the genuine transaction devices have not been compromised, and the attacker can only stand in the middle, this may not always be the case. Relay attacks that require a malicious application to be installed on the user's device have been demonstrated [131]. The malicious application is capable of acting as one of the relay devices, relaying transaction data and tokens, captured from the host device, without the genuine user's consent. While the type of relay attack targeted in the previous chapters can be identified as a Mafia Fraud Attack, the type aimed in this chapter combines elements of both the Mafia Fraud Attack and the Terrorist Fraud Attack (see Section 2.3). In this scenario, the topology of the devices is similar to the Terrorist Fraud Attack, but the user is not knowingly cooperating with the attacker.

However, one of the major challenges for an attacker is to get a malicious application distributed to a substantial population of genuine users. In the Android platform it has been reported that approximately 86% of all malware distribution relies on repackaged applications [171]. Therefore, protecting against repackaged applications may prevent this type of relay attacks as well. As already discussed in Section 2.5, repackaged applications in the course of this thesis are considered applications that try to impersonate a genuine application by slight modifications/variations to the genuine application's artwork and/or changes to its source code in a way that the repackaged application looks and/or feels like the genuine application. The main objective of a repackaged application is to mimic a genuine application so it can target novice users that gravitate towards the popularity/functionality of the genuine application. In this definition, excluded are applications that infringe the potential intellectual property of the original application and present themselves as unique/different applications.

Repackaged applications are a serious threat and are part of the OWASP's *Top Ten Mobile Risks* for 2014 [122], and 2016 [121], posing at places 10 and 8, respectively. Many different methods for detection of repackaged applications have been proposed, but most of them rely on the application stores to perform the detection [40, 41, 45, 75, 135, 166, 167, 170].

In this chapter, a method for detecting repackaged applications is proposed, initiated on the client side, prior to an application's installation. The target application is being checked against a database of legitimate applications, hosted by a TTP. If it does not exist in the database, string and image similarity algorithms are used to detect original applications with similar name and icon pairs. These two elements characterise an application prior to its installation. An attacker who wants to increase the spreading rate of a repackaged application by taking advantage of the already established popularity of an original application might not be likely to alter these elements.

The experimental results demonstrate that the proposed mechanism is both effective and efficient in detecting repackaged applications. Furthermore, in comparison to existing methods, the proposal scored high on a set of predefined criteria. However, this approach is meant to complement static analysis tools, as it provides the important feature of being application source agnostic, and not replace them.

8.2 Proposed Solution

A repackaging detection technique that takes advantage of the attacker's reluctance to significantly alter elements that characterise an application without substantially minimising the attack vector, is proposed. The elements that characterises an application prior to its installation are its name and icon. Since there is a high probability that little or no modification is likely to have taken place on these elements of a repackaged application, image and string comparison algorithms can be used in order to determine the potential original application by comparing them against a database of authentic applications.

This technique is initiated from the client side, prior to the installation of an application. Relevant data from the application is transferred to the TTP's server that maintains a database of original and trusted applications, as well as a blacklist of malware and repackaged applications. Based on the analysis, the TTP will either verify that the application in question is genuine, or return a list of genuine applications that this application may be trying to masquerade.

It is assumed that applications kept in the TTP's database have been sanitised and do not contain malicious code. It is also assumed that developer–application pairs in the database have been confirmed and no repackaged applications exist. Applications from the Google Play Store are used for the population of the trusted database. Studies have found that 1.2% of all applications of the Play Store are repackaged [18], so it is reasonable to consider it as relatively trusted for the purposes of this chapter. However, in a real world scenario, building the TTP's database is a challenging task, but the construction methodology is out of the scope of this thesis.

8.2.1 Threat Model

The capabilities of an attacker for the scope of this chapter are as follows:

- An attacker can access any legitimate Android application. This includes being able to decompile, modify, recompile, or copy and use elements (including, but not limited to media files) of applications.
- An attacker cannot significantly alter the name and the icon that characterise an application, without going unnoticed by potential victims.
- An attacker can distribute applications on any channel (e.g. third-party application stores, online forums, etc.), except the Google Play Store. In the scope of this chapter, the Google Play Store is considered as the trusted entity.
- An attacker cannot sign applications using the original developer's signing key.
- An attacker cannot influence the TTP's database.
- An attacker cannot forcefully install an application or have access to a victim's device.

8.2.2 Assumptions

As already mentioned, the assumptions are the following:

- It is assumed that applications kept in the TTP's database have gone through thorough analysis and do not contain malicious code.
- It is assumed that developer–application pairs in the database have been confirmed and no repackaged applications exist.

Although the exact detection details that the Google Play Store uses are not publicly available, the use of repackaging detection mechanisms in combination with users being able to report applications, aid towards building a safer ecosystem. The use of repackaged application detection mechanisms in certain application stores, such as the Google Play Store, does not degrade the importance of the proposed solution, since Android users are not restricted to downloading applications from the Google Play Store, and the proposed solution is application store agnostic.

8.2.3 Requirements

The requirements that should be met in order to ensure the feasibility of the proposed solution are the following:

- The process should be fast. A user who is installing an application is not likely to wait for more than a few extra seconds for the process to finish.
- Using a remote server to assist the detection process should use as little bandwidth as possible. Bandwidth may cost, and uploading an application that might be several megabytes in size may not be acceptable for a user, who might refrain from using such service.

- The detection process should not have a noticeable impact on the user device's performance. The method should be integrated to the package installer and run prior to an application's installation.

8.2.4 Proposed Solution's Overview

A high level overview of the proposed solution would be as follows:

1. On the client side, when the user has selected to install an application, prior to the initiation of the installation, data required for the detection process is extracted from the `.apk` file and transferred to the TTP's server through a secure channel (e.g. using SSL).

2. The TTP's server, after receiving the data from the device, processes it and takes a decision whether the target application is safe (i.e. exists in the trusted applications database), unknown to its database, potentially repackaged, or repackaged/malicious. The decision is then returned to the client.

3. According to the returned result, the client device will either continue by installing the application without any warnings (in case it is classified as safe), or warn the user that it is unknown, potentially malicious, or malicious. In case it is classified as potentially repackaged, a list of possible original applications is also presented to the user, with links to safe ways of obtaining them.

The proposed approach should either be integrated with Android's application installer as an opt-in feature, or be a separate opt-in application or process that executes prior to the application installer. Some privacy concerns might arise, since information regarding the application that is going to be installed are transmitted to a remote server. Offering the service as an opt-in feature would enable users to refrain from using it. Anonymising the transmitted data can help improve the privacy, but the investigation of privacy concerns is out of the scope of this chapter.

The proposed solution uses techniques already available on smartphones, making it technically feasible. Since high performance is one of the requirements of the proposed solution, the impact on the time required for the installation of an application should be minor. Finally, the user is notified only in the case of a suspicious application, making the process invisible to the user, unless a threat is detected. Therefore, from the perspective of the user, unless an issue is detected, the impact on the user experience should be negligible. However, issues might arise in case there is no internet connection during the installation. Such issues should be considered prior to deployment, but such discussion is beyond the scope of this chapter.

8.2.5 The Detection Process

The TTP's server requires five elements for the detection process. These elements are extracted from the target `.apk` file on the client side, prior to the initiation of the installation process, and are transmitted to the server over a secure channel.

The five elements are:

1. The hash of the `.apk` container (referred as *'signed `.apk`'*). It is used to speed up the process in case the application being tested is known to be legitimate (exists in the trusted applications database).

2. The hash of the `.apk` container, excluding the `META-INF` folder (referred as *'unsigned `.apk`'*). It is used to detect blacklisted applications, regardless of the developer's signature, or applications that are legitimate but have been signed with a different developer signature.

3. The hash of the developer's signature. It is used to determine whether a potentially repackaged application could be a version of a legitimate application that is not maintained in the TTP's database. In such case, the application being tested and the potentially legitimate application should be sharing the same developer signature.

4. The application's name.

5. The application's icon.

In case the application being tested does not exist in the trusted application database, the application name and icon are used for detection of visually similar applications.

A pre-computed list of these elements for all trusted applications is stored in the TTP's database, against which the received data is checked. The names and icons of older versions of an application are also kept, in case they have changed.

Algorithm 3: Repackaged Application Check

Input: String signedHash, String unsignedHash, String appName, Image appIcon

```
/* Check the blacklist database for an unsigned hash match     */
if blacklisted(unsignedHash) then
    return getOriginalApp(unsignedHash)
end
/* Check the applications database for a signed hash match     */
else if signedHashExists(signedHash) then
    return applicationSafe()
end
/* Check the applications database for an unsigned hash match  */
else if unsignedHashExists(unsignedHash) then
    return getOriginalApp(unsignedHash)
end
/* Find similar applications based on their name and icon      */
else
    return getSimilarApps(appName, appIcon, devSignature)
end
```

Algorithm 3 explains the process followed by the server in order to check the originality of an application. In the algorithm, the function *blacklisted(String unsignedHash)* queries the blacklist database and returns true if the input unsigned hash exists. This first step increases the overall performance of the system by quickly detecting known malicious applications. The functions *signedHashExists(String signedHash)* and *unsignedHashExists(String unsignedHash)* query the trusted database for matches and return true if the signed or unsigned application's hash exist, respectively. The function *getOriginalApp(unsignedHash)* returns the original application that is linked to that hash, if such exists. The function *applicationSafe()* returns a code that denotes that the application is legitimate. However, if no trusted application is found, *getSimilarApps(String name, Image icon, String devSignature)* returns an array of trusted applications that are similar to the input, based on their name and icon. The detection process is described in Section 8.2.6. In case an application does not exist in the database, but is found to be similar to another, it may be an updated (or older) version. For this reason, the hashes of the developer signatures of the tested application are compared and if they match, the server notifies that it may be a different version of a certain legitimate application. A developer should use the same signature(s) for different versions of an application, otherwise the operating system will refuse to complete the update process.

The result is finally returned to the user's device. If the application does not exist in the database, but similar applications are detected, a list of these is presented to the user. If the application is found to be legitimate, the installation process will continue without any further warnings. If the application is found to be blacklisted, signed with a different signature than the original, or not found in the trusted database, the user is warned. A flow chart of the process is also presented in Figure 8.1.

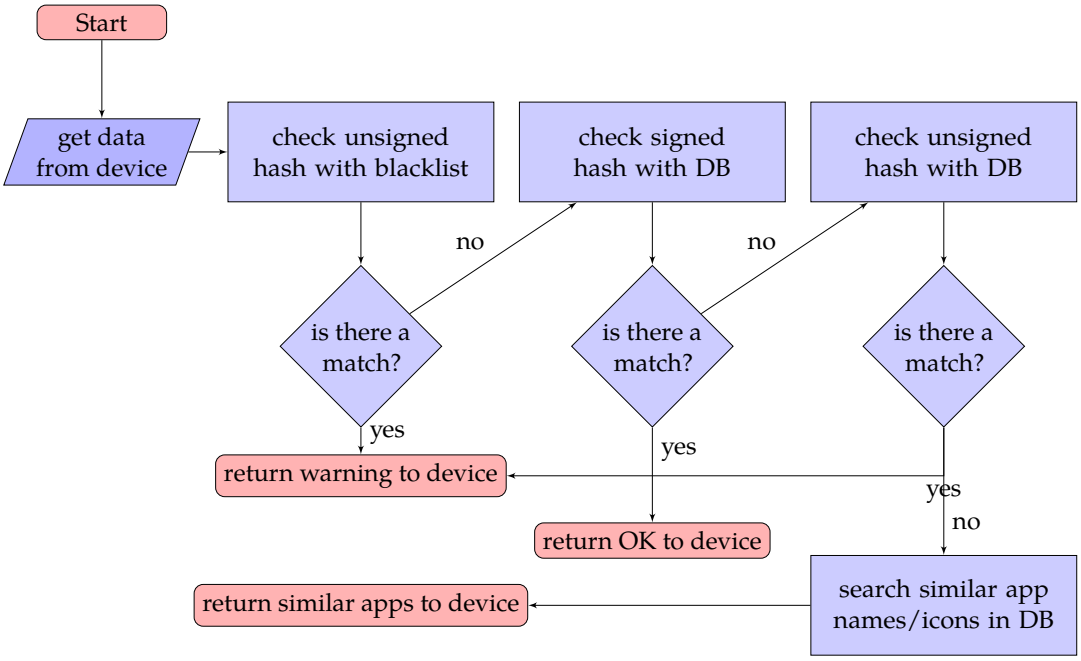


Figure 8.1: Repackaging Detection Flow Chart

Any application with signature or code modifications is treated as non-trusted by the proposed solution. Threats from such applications have been discussed in Section 2.5.1. Table 8.1 lists the threat level of the different possibilities of the detection process.

TABLE 8.1: Possibilities

	Safe	Potentially Unsafe	Unsafe
Signature exists in malicious applications DB			
Signed `sha(apk)` exists			✓
Signature exists in genuine applications DB			
Signed `sha(apk)` exists	✓		
Signed `sha(apk)` does not exist		✓	
Unsigned `sha(apk)` does not exist		✓	
Signed `sha(apk)` does not exist		✓	
Unsigned `sha(apk)` exists		✓	
Signature does not exist in genuine applications DB			
Unsigned `sha(apk)` exists		✓	
Unsigned `sha(apk)` does not exist		✓	

8.2.6 Similarity Detection

Well-known string and image similarity methods were evaluated. For completion reasons, a comparison and rationale behind the choice of the selected methods for the proposed solution is provided.

Name Similarity

For the name similarity detection, the Jaro-Winkler [163] method was used. Previous work [29] has shown that it is more accurate than other string distance metrics for short strings, like names.

It was compared against the popular Levenshtein distance metric. The results are presented in Table 8.2. The input string was checked against the whole set of data stored in the trusted database. More information regarding the construction of the database and the data sets that were used is provided in Section 8.3.2. In the table, the *'Rank'* column demonstrates the order, while the *'Similarity'* column contains the similarity between the input string and the returned string, as calculated by the algorithms. The results that Jaro-Winkler returned were found to be perceptually more accurate and the performance of the algorithm significantly better.

Icon Similarity

Various methods for the icon similarity detection were tested. In contrast to string similarity, image similarity detection algorithms are more complicated and the accuracy of their results is highly dependent on the given image set.

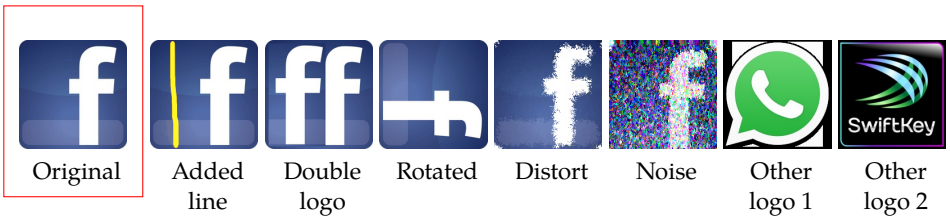


FIGURE 8.2: Test Set

The state-of-the-art in image recognition are training algorithms [42]. Since application icons in most occasions are drawings that do not frequently change, training an algorithm can be extremely challenging, due to the lack of input data that is required. Therefore, a more applicable method for tackling this problem had to be used.

Comparing the histograms of two images produced promising results during the experimental phase. Colour similarity between the images and pHashes (perceptual hashing) [98] were also tested, but gave significantly worse results. Finally, a method introduced by Jacobs et al. [88] was tested. Their method uses Haar Wavelet Decomposition techniques in order to determine how many significant wavelet coefficients the input image has in common with a target image and therefore determine the amount of similarity.

The results of the image comparison between the original image and the rest of the images shown in Figure 8.2, using the Haar Wavelet and histogram comparison methods, are presented in Table 8.3. More experiments were conducted, using different sets of images, that produced similar results in terms of accuracy.

TABLE 8.2: Jaro-Winkler and Levenshtein Distance Top 10 Results Based on Input *'googl app stoy'*

Results	**Jaro-Winkler**		**Levenshtein**	
	Rank	**Similarity**	**Rank**	**Similarity**
Google Play Store	1	0.904008	1	7
Google Earth	2	0.873016	3	8
Google Classroom	3	0.86756	2	8
Google Fit	4	0.866667	5	8
Google Sheets	5	0.864469	6	8
Google Docs	6	0.862284	4	8
Google Apps Device Policy	7	0.855801	—	—
Google Translate	8	0.839782	—	—
Google Slides	9	0.839744	—	—
Google Goggles	10	0.83631	8	9
GoPro App	—	—	7	8
Google Finance	—	—	9	9
Hack App Data	—	—	10	9
Average query execution time*	*3.1 seconds*		*6.8 seconds*	

*Based on 50 executions of the query

The histogram comparison was tested for binarised versions of the images as well. The OpenCV [119] library was used for the histogram comparison, and the Correlation Algorithm was used to calculate the distance between two histograms. The Correlation Algorithm was chosen because it produced significantly better results than the other histogram comparison algorithms that OpenCV provides (Chi-Square, Intersection, and Bhattacharyya distance).

The Haar Wavelet method was noticed to be more resistant to images with added noise or distortion, while the histogram method could detect rotated images better. Histogram comparison is also more likely to return high similarity in cases where the target image has similar colours as the original one, but a different pattern is depicted. This could possibly lead to many false positives. Moreover, since application icons are typically relatively small, it is more likely that added noise or distortion will go unnoticed, than rotation. Therefore, the Haar Wavelet method was chosen to be used. More efficient algorithms for image and string similarity that will further improve the results may exist, but this is not the main focus of this chapter.

Application Similarity Metrics

A user is more likely to trust (and consequently install) an application in which they can recognise symbols that are familiar to them [129]. Therefore, the higher the resemblance, the more likely it is for an application to get installed.

In the proposed technique, two similarity detection algorithms were used; name similarity and icon similarity. These algorithms return their similarity score, $\mathcal{X}$ and $\mathcal{Y}$ for the name and icon similarity respectively. To combine these similarity scores and compute the overall similarity, Equation 8.1 was used.

$$Sim_{a,b} = 10\frac{\mathcal{X} \times 10^{\mathcal{X}} + \mathcal{Y} \times 10^{\mathcal{Y}}}{2} \tag{8.1}$$

In the equation, $\mathcal{X}$ represents the similarity percentage between the names of the applications a and b, and $\mathcal{Y}$ the similarity percentage between their icons, as returned from the algorithms described in Sections 8.2.6 and 8.2.6, respectively. The proposed equation weights the name and icon similarities in a way that high similarity in one

Table 8.3: Image Comparison Methods

	Haar Wavelet	**Histogram**	**Binarised Icon's Histogram**
Original	100.00	100.00	100.00
Added line	57.49	61.55	100.00
Double logo	46.97	71.98	94.33
Rotated	15.72	88.84	99.99
Distort	89.70	53.48	99.97
Noise	47.15	38.81	91.68
Other logo 1	4.01	7.67	52.87
Other logo 2	8.52	0.70	99.81

of the two elements would have a significant impact on the overall similarity. The same weight is given to the two similarity scores, which are then averaged.

During the evaluation phase, in order for two applications to be considered similar, the similarity score returned by Equation 8.1 should have exceeded 40%. The threshold of 40% was chosen because during experimentation it was giving the most promising results in terms of false-positive and false-negative rates. However, this score mainly focuses on improving the usability of the proposed solution by not returning to the user a large list of potentially irrelevant applications. It can be readjusted according to the operational requirements and the similarity metrics that are used. Similarly, alternative approaches for weighing the name and icon similarity results may be used.

8.3 Implementation

In order to test the effectiveness of the solution, two tools were built. An Android application capable of extracting the required features from an `.apk` file, and a server responsible for the repackaging detection process.

8.3.1 The Android Application

The Android application takes as input an `.apk` file and extracts the required features, as described in Section 8.2.5. It first extracts the name, the icon, the package name, and calculates the hashes of the signatures that exist in the application. The name (since it may contain special characters) and the icon are encoded using Base64 encoding in order to be transmitted over the network. Finally, it decompresses the application to a new directory, removes the `META-INF` directory, compresses it again and calculates the SHA256 of the signed and unsigned applications. The extracted data is being sent to the detection server, using SSL, in order to maintain the integrity and confidentiality.

In the experiments, an SGS5 mini (SM-G800F) was used, running Android 4.4.2. It is regarded as a low-specifications Android smartphone.

Challenges

Ideally this mechanism should be built in the package manager process or automatically run when an application is about to be installed. Due to Android limitations, the previous is not possible without further system modifications. A customised version of Android can integrate both, since the package manager is part of the Android AOSP. Since the integration of the solution to the operating system would not have any impact on proving its robustness, a proof of concept application that takes an `.apk` file as input and performs the whole process was built instead.

8.3.2 The Server Application

The server application was mainly written in Java, running on top of a Tomcat server, version 7.0.57. The host machine was a laptop with 8GB of RAM and an Intel Core i7–2620M CPU, running at 2.7GHz. The OS was Ubuntu 14.10, Desktop edition. The database software used in the experiments was MySQL, version 5.5.41. A set of 2676 free applications were crawled from the Google Play Store (referred to as $\mathcal{S}_T$), using a crawler that was written in Python, and were added to the database. These applications were considered as authentic and non-repackaged.

The database consisted of three tables. The first one was holding the blacklisted application unsigned hashes, along with a link to the corresponding original application (if there was one). The second contained the developers of the trusted applications, along with the hash of their public key. The last one contained the essential information about the applications. This information included the application name and icon, the signed and unsigned application's hashes, the developer's ID, as well as other data that were crawled from the Google Play Store, like the package name and total number of downloads.

All the hashes were calculated using the SHA256 algorithm. The built-in `MessageDigest` Java class was used for the calculation of the hashes.

Application Similarity

- *Name similarity:* The name similarity was performed in the database, using an implementation of Jaro-Winkler written in SQL[8], in order to increase the performance. No further actions were taken in order to optimise the code for performance.

- *Icon similarity:* Image similarity detection was the biggest challenge, due to the multitude of the existing algorithms. It had to be efficient, while returning as accurate results as possible. It was developed in Perl, using the `Image::Seek` module, which is capable of calculating image similarity based on the Haar Wavelet Decomposition method. Coefficients of the decomposition of the application icon in question were compared with pre-computed coefficients of trusted applications in the database.

8.4 Results and Evaluation

After the population of the trusted server with legitimate applications, as described in Section 8.3.2, three experimental phases were conducted in order to assess the robustness of the proposed solution. In the first two, applications from the Drebin malicious dataset [14, 15] (referred to as $\mathcal{S}_M$) were used as input on the client device.

[8]Jaro-Winkler SQL source code: https://androidaddicted.wordpress.com/2010/06/01/jaro-winkler-sql-code/

During the third phase, legitimate applications that exist in the trusted database were used, in order to measure the performance of the solution in this scenario.

In the first phase of the experiment, the accuracy of the proposed solution was tested against applications with very high probability of being repackaged. According to previous research [103], altering the package name of an application would lower its visibility in markets. Therefore, a set of such applications can be generated by extracting applications from the set $\mathcal{S}_M$, with package names that exist in the Play Store. A total of 227 free (non-paid) applications (referred to as $\mathcal{S}_F$) were extracted and used during the experimentation.

In the second phase, 100 random applications (referred to as $\mathcal{S}_R$) were chosen from $\mathcal{S}_M$, excluding $\mathcal{S}_F$ (so, $\mathcal{S}_R \subseteq \mathcal{S}_M \setminus \mathcal{S}_F$), in order to further investigate the accuracy of the solution with applications that are most likely not repackaged (false positive). The main goal of this test was to measure the amount of non-repackaged applications that would be detected as repackaged.

In the last phase, 50 representative legitimate applications (referred to as $\mathcal{S}_L$, where $\mathcal{S}_L \subseteq \mathcal{S}_T$) were randomly selected and tested in order to measure the performance of the system when legitimate applications are tested.

The first two phases of the experiments were analysed in terms of accuracy and performance. Manual inspection of the applications in the sets was finally performed in order to assess the quality of the results.

8.4.1 Results

Table 8.4 presents the results from the three experimental phases described in the previous section. *'Average Time/Check'* refers to the average time required from the moment the smartphone device initiated the process, until it displayed the returned results on its screen. Cases in which the original version of the repackaged application being checked was included in the returned results were marked as true positives. Returned results that did not include the original application were marked as false positives. True negatives were considered cases in which no results were returned from the server and no corresponding original application existed. Finally, cases in which no similar applications were returned, and the trusted database included the original application were marked as false negatives.

During the first phase of the experiment, 189 applications (83.3%) returned exactly one result. A total of 15 applications (6.6%) returned two results. One application

Table 8.4: Experimental Results

	Set Size	True Positive	True Negative	False Positive	False Negative	Average Time/Check
Phase 1	$\mathcal{S}_F$ (227 apps)	203	3	2	19	3.42 seconds
Phase 2	$\mathcal{S}_R$ (100 apps)	11	88	1	0	3.9 seconds
Phase 3	$\mathcal{S}_L$ (50 apps)	50	0	0	0	0.6 seconds

(0.4%) returned three results and the 22 remaining (9.7%) did not return any result. The time required for the extraction of the features from the `.apk` file on the device was approximately 0.18 seconds. Approximately 2.6 seconds were spent on the application name comparison on the server side. The time required for the image similarity detection was negligible (<1ms). The average application size was 2.1MB. A slight time overhead may be added on the client side in case of larger applications, since the decompression phase, as well as the computation of the hashes might take longer to complete. However, this process only takes place once, prior to the applications installation, causing a negligible impact on the overall operating system's performance.

During the second phase, 9 applications returned exactly one result. Two returned two results and one returned four results. 3.5 seconds were required for the server to take a decision, 3.2 of which were required for the string similarity comparison. A total of 0.14 seconds were required for the Android device in order to extract the required data from the `.apk` file. The average application size was 1.5MB.

Finally, during the third phase, only applications that already existed in the database were tested. This phase was performed in order to calculate the system's performance in this occasion. A total of 0.04 seconds were required by the server for the detection process. Data extraction on the client required 0.3 seconds to complete and the average `.apk` file size was 4MB.

The database that holds the Haar transformation norms and gets populated when new icons are added to the set of application icons, requires an average (after 5 runs of the population script) of 1 minute and 7 seconds, for a total of 2676 images. Therefore, 0.025 seconds for each new added icon. However, this process only needs to take place during the database population phase, and will not impact the later detection process.

8.4.2 Evaluation

The results were evaluated in two phases.

First Phase

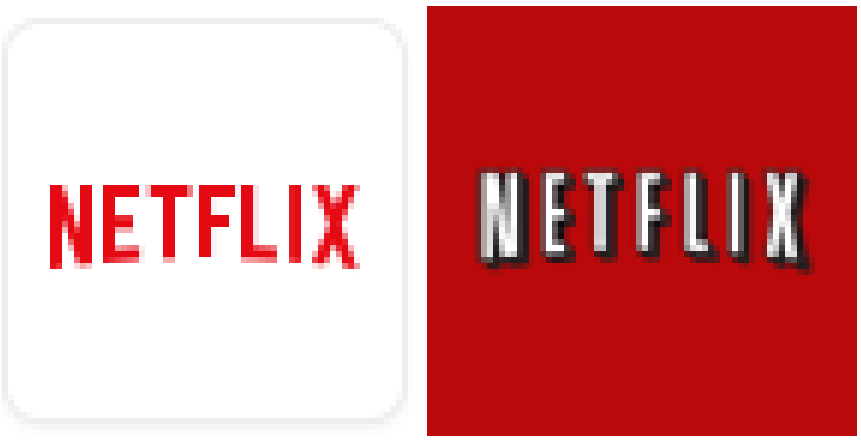
After manual analysis, both false positives detected in this phase were found not to be repackaged applications of the applications with the same package names in the Google Play Store. Both the package names were simple (`com.hotel` and `com.acid`) and the existence of applications with the same package name was most probably due to a coincidence.

Nineteen false negatives were also detected. Fifteen of them shared similar icons with the corresponding original applications. Their application names were in Chinese, in contrast with the original applications whose names are in English. The measurement of the name similarity returned 0 and the amount of the icon similarity was not enough to compensate. Since the samples in $\mathcal{S}_M$ were collected between

2010 and 2012, and the original corresponding applications may have been updated numerous times since then, it is unclear whether the application names of the original applications were in Chinese at the time of capture, or the names of the repackaged applications have been altered. The original application developers in all cases seem to be based in China, based on Google Play Store data.

The remaining four false negative results were returned by applications that had a completely different icon than the original application, and a different name. Again, it was not clear whether the attacker who performed the repackaging altered the name and icon, or these were the name and icon of the original application at the time of repackaging.

The remaining three applications that did not return any results were not repackaged versions of applications that exist in the Google Play Store. They only shared the same package names with Google Play Store applications. Therefore, correctly, no results were returned.



(A) Original (B) Repackaged

FIGURE 8.3: Netflix Application Icons

Forty applications that were correctly recognised had similar names, but different icons to the original ones in the database. For example, a repackaged version of Netflix contained an old application logo, which compared to the newer one has reverse colours, different fonts, shadows, and rounded corners (Figure 8.3). Applications with resized icons between the original and the repackaged version were not counted, since all icons were resized to 128×128 pixels prior to the similarity detection.

Moreover, it was observed that in all occasions when the server would suggest more than one potentially original applications, the one with the highest similarity to the application being tested was the correct suggestion.

Second Phase

During this phase, no false negatives were detected. There was one false positive and eleven true positives.

The true positive results were not detected as repackaged when only looking at the package name, that suggests that it might be a common practice for attackers to alter it. Four applications were also detected to have a slightly different icon than the original applications.

Finally, the false positive was caused by an application that had the default icon that the Eclipse IDE adds on newly created Android projects. A total of four applications with this particular icon existed in the database and were returned. It is a common practice for developers to change the default icon, in order to brand the application.

Observations

The proposed solution demonstrated its effectiveness in detecting repackaged applications (91.5% detectability, according to the results of the first experimental phase). Its high performance in terms of speed also demonstrates that it is likely to be applicable to real-world scenarios. However, further investigation is required prior to deployment, mainly in order to address potential privacy concerns, issues regarding the population of the TTP's database, and handling in cases of poor or no internet connectivity.

During the experiments, repackaged applications that were not using the original application's source code, but only its name and icon, were also detected. These applications would probably have gone undetected by many countermeasures that are based on different methods, discussed in Section 2.5.2.

An application that is known to use this repackaging technique, disguising itself as the Google Play Store [165], but using the icon of an older version of the application and *'googl app stoy'* as its name, was successfully detected. Only one result was returned by the server; the original Google Play Store. Although this type of attack is not new [33], this kind of applications might not get detected by most alternative repackaging detection techniques. More applications like the previous were detected during the experiments.

The best example is a fraudulent application with package name `net.android.app`. Four variants of the application were found in the set $\mathcal{S}_F$, using this package name. After manual inspection of the disassembled code, it was concluded that all variants share the same source code, which is capable of sending text messages to premium-rate numbers, without the user's authorisation.

The first version of the sample (based on that it was found in the first chronologically ordered directory of $\mathcal{S}_M$), uses *'браузер'* (Russian for *'browser'*) as its name, and the logo of Microsoft Internet Explorer as its icon. Since no similar applications were included on the server, correctly, no similar applications were returned.

The later variants of the sample used different names and icons. The first two used *'Skype'* as their name and the original Skype icon. The last used *'Opera Mini'* as its name and the Opera Mini browser's icon. It is believed that this was a well

thought move from the attacker's side towards increasing the distribution vector of the application.

Most of the existing application repackaging detection methods may not be capable of detecting this type of application repackaging. The results suggest that it is not a common practice for attackers to alter the names and icons of repackaged applications.

8.4.3 Comparison with Previous Works

Table 8.5 presents a comparison between the proposed solution and popular previous solutions, described in Section 2.5.2. A short description and explanation is provided below:

Most of the previous works rely on the detection being performed on the application store side *(criterion 1)*. It has been noticed that many third-party markets do not remove malicious applications, and in many occasions the distribution of malicious applications and adware is being done intentionally [103]. Since anyone with the required technical knowledge can potentially create an application market, it is unsafe to rely on them for the detection and removal of potentially harmful applications. Many application markets are not likely to scan for repackaged applications due to the computational cost and/or the lack of technical expertise. Finally, there are dedicated websites and forums that distribute Android applications. Scanning application market servers for malicious and repackaged applications does not protect against applications that have been downloaded from such sources. The proposed solution was designed to be able to initiate the detection process from the client side, making it application download source agnostic *(criterion 2)*, and notifying the user in order to prevent the installation of a potentially repackaged application *(criterion 3)*.

Another shortcoming of the previous techniques is that they are mostly based on code analysis, or use features that are strongly linked with the application's code. Therefore, they would not be capable of detecting applications that claim to be legitimate by just imitating their name and application icon, but using a completely different code-base *(criterion 4)*, or when advanced obfuscation techniques have been

Table 8.5: Comparison with Previous Works

Criteria	DNADroid	DroidMOSS	AppInk	Proposed
1. App store centric	✓	✓	✗	✓*
2. Download source agnostic	✗	✗	✓	✓
3. Client notification	✗	✗	✓	✓
4. Detects stolen branding	✗	✗	✗	✓
5. Detects stolen source code	✓	✓	✗	✗
6. Resilient to code obfuscation	✗	✗	✓	✓
7. Requires trusted sample	✓	✓	✓	✓

*Although the current implementation focuses on the client side, there are no restrictions for scanning an application store using the proposed solution

applied *(criterion 6)*. Solely in 2014, at least two samples were detected, claiming to be the Google Play Store application, none of which was using the original application's source code [13, 165]. Such solutions, however, may be able to detect whether an application has stolen parts of the source code of another application *(criterion 5)*.

Finally, most proposed solutions have to maintain samples of trusted applications (or some other trusted derivative), against which a tested application is compared *(criterion 7)*. A repackaged application cannot be detected by any method, unless that method maintains the necessary information of the authentic corresponding application.

8.5 Summary

In this chapter, a repackaging detection method was proposed, that takes advantage of the attacker's inability to significantly modify the application's name and icon, while maintaining the same attack vector. Therefore, an attacker is not likely to perform such modifications. The experimental results showed that the initial argument was valid. Through a prototype that was built, it was possible to detect the original applications with high confidence, given a repackaged application as an input.

Compared to previous works, the proposed solution was found to be about as accurate as many application market level detection techniques, but capable of initialising the detection process from the client side, allowing for source-independent detection. Only a few kilobytes of data is required for the detection, which adds a slight overhead only on the application installation process. This feature is very important, as many application stores may not use repackaging detection techniques. Google has started providing to users client-based detection of malicious applications through Google Play Protect [64], even though in the past it used to mainly rely on the server-based *Bouncer* [96, 105]. Moreover, no special actions are required on the developer's side. Finally, it is capable of detecting a variation of repackaged applications that only share the same name and icon with the original. Since repackaged applications are responsible for approximately 86% of the malware distribution on the Android platform, protecting against them can protect against various types of attacks, including the variation of relay attacks described previously in this chapter.

Chapter 9

Conclusion and Future Work

Relay attacks pose a serious threat towards the entities involved in a transaction. The victims of such attacks may include not only the users who are performing a transaction, but also banks (e.g. in the case of financial transactions), transport operators (e.g. in the case of attacks on ticketing systems), and corporations (e.g. in the case of relay attacks on access control systems), among others. This may result in financial loss or loss of reputation, identity theft, and more.

Relay attacks are difficult to be detected by authorities beyond the victim user. In the field of smart cards, distance bounding protocols have demonstrated high effectiveness in relay attack detection. However, in the field of smartphones, distance bounding protocols might not be effective, as discussed in previous chapters. One alternative PRAD solution that has been proposed suggests the collection of measurements from the immediate ambient environment of the transaction devices (the natural ambient environment). The measurements are subsequently compared for similarity, in order to establish proximity evidence. The data should be collected during the course of a transaction, using some ambient sensor(s). For example, in the case of a payment transaction, when the user taps the transaction instrument on the transaction terminal, both devices should collect data during the course of the transaction. One of the devices or a TTP (e.g. the bank) is then responsible for the PRAD process, through comparison of the collected data.

In this thesis, the effectiveness of the natural ambient environment was initially assessed. The evaluation focused on time-restricted mobile contactless transactions, taking into account industry requirements, such as the ones imposed by EMV. Previous work did not take into account such restrictions, which nonetheless apply on a large number of security-critical mobile contactless transactions, such as mobile payments and transportation-related transactions.

The initial evaluation of the natural ambient environment as a PRAD mechanism demonstrated results that may be inadequate in respect to PRAD for security-critical applications. Subsequently, different approaches were proposed in order to improve the accuracy of PRAD for such transactions.

First, the generation of an AAE was proposed. For this approach, the devices that communicate during a transaction are responsible for generating and/or measuring the AAE that is produced through some of their peripherals (e.g. vibration). The

generation of an AAE should be based on a random stream or sequence and be easy for the genuine transaction devices to establish PRAD-related information, while being difficult for an attacker to relay at a remote location. Similar to natural ambient environment sensing, the data collected by the transaction devices are compared for similarity and a transaction is approved if the similarity exceeds some certain threshold.

From a list of proposed peripherals that can potentially be used as AAE actuators, infrared light and vibration were evaluated, both of which demonstrated positive results (~98% TAR, and 0% FAR on infrared, and near perfect classification of genuine and relayed transactions on vibration). The applicability of infrared in real-world scenarios was also evaluated, showing that there is potential for the proposed solution to be integrated to commercial applications/products.

Moreover, two approaches that do not rely on sensing of the ambient environment were proposed. The first of them suggested the use of a random sequence of virtual buttons that are presented on predefined locations of the display of the transaction instrument (smartphone). The transaction instrument is placed on a special 'plate', operated by the transaction terminal, capable of detecting the approximate location of where force is being applied on the device which is placed on it. The genuine user of the smartphone is called to touch the buttons that are displayed at random predefined locations of the screen. The transaction terminal is capable of detecting which button was pressed on the mobile device, based on the capabilities of the sensing 'plate'. Both transaction devices record the pressed button sequence, the duration of button presses, as well as the duration between subsequent button presses. The timings and the sequence recorded by the two devices are then compared for similarity in order to determine whether the devices are in proximity. The solution demonstrated high effectiveness as a PRAD mechanism. A set of ten attackers tried to attack the proposed system, however all of the attempts (200 in total) were successfully detected, both using threshold- and machine learning-based analysis techniques.

Finally, a method to detect repackaged applications was proposed. Repackaged applications have been found in the past responsible for the distribution of approximately 86% of all Android malware. Moreover, relay attacks that use a malicious application installed on a genuine user's device, instead of relying on masqueraded devices, have been demonstrated. Since repackaged applications are a major source of malware distribution, detecting such applications prior to their installation is crucial for protecting against a large variety of attacks, including relay attacks. The proposed solution suggested that looking at the features that characterise the application prior to its installation, i.e. its name and icon, in case it is not found to be genuine, can assist towards detecting whether the application might be potentially repackaged. An evaluation of the proposed solution revealed high accuracy in detecting repackaged applications. Moreover, the detection process is initiated from the client side, which makes the solution application source agnostic. This is a major advantage over other proposed solutions that rely on application markets to run the repackaging detection,

since users may be using various application sources (e.g. websites), and some markets may not apply repackaging detection techniques.

9.1 Comparison of the Proposed Solutions

While a variety of solutions have been proposed in this thesis, there is no one solution that can fit as a PRAD mechanisms in all possible scenarios. Each one has advantages and disadvantages over the rest. Table 9.1 compares the proposed solutions against a set of criteria.

Table 9.1: Comparison of Proposed Solutions

	AAE		Force Sensing	Repackaging Detection
	Infrared	Vibration		
1. Requires Extra HW on TI Side	•	✗	✗	✗
2. Comparison During Transaction Time	✓	✗	⋆	—
3. Resists Context-Manipulating Attacker	✗	✓	✓	✓
4. Requires User Interaction	•	•	✓	○
5. Unattended Operation	✓	•	✗	—
6. Device Size May Impact Accuracy	✗	⋆	○	✗
7. Phone Cases May Affect Accuracy	○	⋆	⋆	✗
8. PRAD Distance	Up to a Few Metres	0m	0m	—

✓: Yes, ✗: No, •: Potentially/Minor, ○: In some cases, ⋆: Not Evaluated, —: Not Applicable

In the table, the first criterion refers to whether extra hardware (HW) is required on the transaction instrument, other than what is already widely available on modern smartphones. In the case of using infrared as an AAE actuator, extra HW might be required, since not all modern smartphones are equipped with an infrared emitter. Therefore, this PRAD technique might be more suitable for scenarios in which the devices are provided to the users, and cover some HW requirements, as in the case of companies providing devices to the employees that can be used in access control. However, as explained in Chapter 4, an infrared emitter is available in a variety of devices. All other solutions do not require HW that is not likely to be available. In the case of vibration, sensors as the accelerometer are highly recommended to be included on Android devices, and HW features that can be used for notifying the user (such as vibration) *must* be included [5]. Similarly, according to the same specifications document, devices *must* be equipped with at least a 2.5 inches display, and support touchscreen input (therefore the force sensing PRAD approach would not require extra HW on the device side). Finally, devices *should* include support for one or more forms of 802.11, therefore the repackaging detection approach does not have extra requirements either. This is an advantage over certain previous works as well, that rely on sensors that are uncommon on Android devices (e.g. temperature, humidity, and precision gas).

The next criterion refers to the ability of the proposed solution to perform the similarity comparison between the captured data from the two devices during the

time-frame of the transaction. In Chapter 5, this was evaluated for infrared, with successful results. In the case of vibration, a TTP is required, since the vibration duration is 500ms (equal to the maximum permitted transaction time). Thus, the comparison can only be performed post-transaction time. So, if the transaction is accepted or rejected, the transaction devices will be notified posteriori. This mechanism is therefore intended for transactions in which at least one of the devices is online. In the case of force sensing, such evaluation was not performed, however the amount of data that has to be transferred between the devices is a few bytes long (smaller than in the case of infrared), and the threshold-based analysis only requires a minimal amount of calculations. The amount of data and calculations required for the force sensing PRAD mechanism is an aliquot part of the amount of data and calculations required for the infrared AAE-based mechanism. It is safe to assume that it is possible for the force sensing PRAD to effectively perform the comparison during the transaction time, when threshold-based analysis is used, but further investigation might be required. In the case of device-specific thresholds, possibly a TTP will be required, in order to hold a database of all the thresholds and models. It is worth noting, however, that in this particular technique the user input takes place during the transaction, but outside of the 500ms time-frame, as happens in the case of PIN authentication. Finally, in the case of repackaging detection, this criterion is not applicable, since the detection takes place during the installation of the application and not during the transaction time.

The third criterion refers to the effectiveness of the solution in the presence of a context-manipulating attacker. In the case of natural ambient sensing, as it has already been discussed, a context-manipulating attacker may be capable of creating false acceptances and rejections when certain sensors are used to measure the ambient environment [138]. In the case of the proposed solutions, such an attacker cannot interfere with the non AAE-based solutions. In the case of using vibration as an AAE actuator, the evaluation demonstrated that the vibrations performed by an attacker do not largely impact the effectiveness of the solution. However, as the evaluation of the effectiveness against a context-manipulating attacker was not the main objective of this work, and further attacks might be possible, further investigation may be required. Finally, in the case of infrared, an attacker was found not capable of achieving a false positive when using off-the-shelf equipment and custom-built devices, based on various off-the-shelf components. However, an attacker can create a false negative by emitting random infrared signals from a distance that may result in the transaction terminal capturing a wrong sequence. Physical barriers to block signals originating from indirect/unintended sources from reaching the infrared sensor were proposed as a countermeasure.

The fourth criterion refers to the PRAD process requiring user interaction. Some form of user interaction is required on all solutions, however, in most of them it is minimal. In the case of infrared as an AAE actuator, the user should keep the transaction instrument's infrared emitter in a 30 degree angle from the transaction

terminal's infrared receiver (on any direction). In the case of vibration as an AAE actuator, the user has to place the transaction instrument on the transaction terminal prior to the initiation of the transaction. In the case of application repackaging, if an application is found to be potentially repackaged, the user is given a warning containing instructions. Finally, in the case of force sensing, the user has to place the transaction instrument on the force sensing plate that is attached to the transaction terminal, and press on a button sequence. This solution requires the highest amount of user interaction. It might be preferred over the others in certain scenarios however, since no extra hardware is required, its effectiveness was found to be very high, and no TTP might be necessary.

The fifth criterion refers to the capabilities of the proposed solutions to operate without requiring to be assisted by a user. This might be useful in various scenarios, for example in IoTs, when two devices need to prove to each other that they are co-located. In the case of infrared, this is possible as long as the infrared emitters and receivers are placed within a 30 degree angle. In the case of vibration it is also possible, as long as the devices are placed on top of each other. However, the impact from the vibration might cause the devices to move over time. Force sensing on the other hand always requires user interaction, so unsupervised operation is not possible. Finally, this criterion is not applicable in the case of repackaging detection.

The sixth criterion is about the impact that the size of the device might have in the accuracy. In the cases of infrared and repackaging detection, the device size is not relevant to the performance of the proposed solution. In the case of force sensing, the device size and weight affected the solution to some extent, but as a countermeasure, device-specific thresholds were also used. In the case of vibration, the impact of the device size was not evaluated, as it was not the main objective.

The seventh criterion refers to the impact that phone cases might have on the accuracy. In the case of repackaging detection, phone cases are not relevant. In the case of infrared as an AAE actuator, phone cases do not affect the accuracy, except if they block the infrared emitter of the device. In the cases of vibration as an AAE actuator and force sensing, no evaluation was performed, as it was not the main scope of these works, but it is likely that especially thicker, softer, and oddly shaped cases may affect the proposed solutions.

The last criterion refers to the operating distance of the solutions. In certain scenarios, like IoTs, device co-location might not necessarily require the transaction devices to be touching. For example, in the case of a home security system, two devices might need to be located in the same room. Using infrared as an AAE actuator, this is possible, since the solution can operate up to a few metres. However, in other scenarios, like EMV payments, a few metres might be a distance in which a relay attack or denial of service attack might be possible. Infrared can still operate in such scenarios, if barriers are applied. It is also worth noting that relay attacks could not be performed successfully during the evaluation that was performed, therefore no evidence exists to suggest that they are possible. In the cases of vibration as an AAE

actuator and force sensing however, the devices should be touching in order for the proximity evidence to be collected successfully. Finally, this criterion is not applicable in the case of repackaging detection.

Finally, it is worth mentioning that the best performance was observed by using the force sensing PRAD, as well as by using vibration as an AAE actuator. In these cases, the least amount of false rejects and/or accepts was observed. However, positive results were observed by all proposed solutions, and as discussed above, certain solutions may be more appropriate than others for certain use cases.

9.2 Applicability of the Proposed Solutions

As is evident from the previous section and discussions in previous chapters, certain proposed solutions might be more suitable for certain scenarios. Prior to deployment of one of the proposed solutions in a real-world application, various aspects should be taken into consideration.

Firstly, usability aspects should be considered. Some proposed solutions may be difficult to operate by users with motor difficulties or impaired vision (e.g. the force sensing-based PRAD). As discussed in Chapter 4, the devices should have the ability to negotiate the sensors and methods to be used for the PRAD. The transaction terminals should be able to provide a multitude of different PRAD techniques, such that the system can be easily used by all potential users. Further investigation is required to be performed by the relevant industries, in order to assess potential usability concerns under specific use cases.

Moreover, the transaction terminals are required to be equipped with extra hardware (except in the case of the host-based detection). In this case as well, the transaction devices should be capable of negotiating the sensors that re going to be used, since not all smartphones are equipped with all the available sensors. Devices that are already in the market will thus be capable of performing PRAD techniques without requiring extra hardware.

Prior to deploying the proposed solutions, further investigation is required to assess their performance in real-world scenarios. A larger variety of devices should also be used during the assessment. Also, alternative techniques should also be investigated, that could potentially provide better results.

Finally, the host-based PRAD can be integrated to existing devices without requiring extra hardware to be installed on the transaction devices. However, in order to be applicable to the industry, certain elements of the solution should be discussed. The authorities responsible for populating and maintaining the trusted applications database should be decided. Moreover, the solution should be integrated to the smartphone's operating system. In Chapter 8, potential approaches that can be followed to perform the integration have been discussed.

Since smartphone-based contactless transactions are on the rise, the proposed solutions can assist towards creating a more secure ecosystem. In various industries,

like banking and transportation, relay attacks may pose a serious threat, leading to economical and reputation loss. Corporations might also experience unauthorised access to their premises, and IoTs that are required to be placed within some distance from each other might be intentionally moved to a greater distance, without being detected, with the aid of relay attacks. Moreover, the users may be affected by such attacks. User impersonation is possible, and in case no relay attack countermeasures have been used, it is hard to prove that a transaction was not genuine, since it occurred between genuine devices. The solutions that have been proposed in this thesis can aid towards providing proximity assurances in scenarios where traditional smart card distance bounding protocols may not be applicable.

9.3 Comparison to Previous Works

Compared to previous works on smartphone-based PRAD, the proposed solutions described in this thesis were found to have a variety of advantages. First of all, the proposed solutions were found to provide better results than previously proposed mobile-based PRAD techniques. Moreover, compliance with industry standards that require transactions to complete in short time-frames (up to 500ms) was achieved. Certain of the proposed solutions also do not require a TTP in order to perform the PRAD. Finally, the proposed solutions were found more resilient against an attacker with context manipulating capabilities.

The major disadvantage of the proposed solutions, compared to previously proposed solutions, is the user friendliness. Most of the previous solutions do not require extra steps on the side of the user in order to perform a transaction (with the exception of [116]). However, the accuracy of previously proposed solutions is lower, especially in the case of transactions with a time limit of up to 500ms. According to the results presented in Chapter 3, at best, approximately one in every five genuine transactions that use the natural ambient environment as a PRAD mechanism will fail and will have to be repeated.

9.4 Future Work

As part of the ongoing research, the work presented in this thesis can be expanded towards various directions. The evaluation of past proposals (Chapter 3) can be further investigated by simultaneously measuring multiple sensors within the transaction time duration. This is to evaluate whether the risk associated with using individual sensors would be reduced. Moreover, a larger range of devices can be used to further assess the effectiveness of the natural ambient environment as a PRAD mechanism.

Similarly, the effectiveness of all the proposed PRAD countermeasures (presented in Chapters 4 to 8) can also be examined further by assessing a variety of different

devices. Moreover, user studies can be carried out, in order to assess the usability, performance, and user acceptance of the proposed solutions outside of a lab environment.

Individual proposals can be extended towards various directions. Chapter 4 *(Preventing Relay Attacks Using Infrared Light)* can be extended by investigating various attacks and potential mitigations against them, like denial of service attack through emission of interfering infrared sequences from a short distance. The use of infrared as an AAE actuator can be investigated further in continuous two-factor authentication for online logins, expanding the solution presented in Chapter 5 *(Infrared Light for Proximity Critical Scenarios)*.

Chapter 6 *(Vibration as an Artificial Ambient Environment Actuator)* can also benefit by examining the impact of collecting data from multiple sensors simultaneously (sensor fusion) on the performance of the proposed solution. Furthermore, the impact that phone cases might have on the proposed solution can be investigated.

Additional features that would minimise the potential of an attacker to perform a relay attack can be examined, expanding the investigation presented in Chapter 7 *(Force-Based Relay Attack Detection)*. For example, the pressure intensity. However, it should be stressed that many devices lack features, like that of measuring the intensity of the pressure that is being applied on their screen.

Finally, the proposed solution presented in Chapter 8 *(Host-Based Relay Attack Detection)*, can be expanded by further investigating string and image similarity algorithms and techniques. A more in depth study can assist towards the better selection of the threshold over which two applications are considered similar, leading to even lower false-positive and false-negative rates. Finally, code optimisations can further improve the proposed solution's performance.

Appendix A

Ambient Sensors

This appendix provides a short description of each sensor.

A.1 Accelerometer

The accelerometer sensor —deployed in most modern smartphones– measures the acceleration applied to the device on the x, y and z axes; its units are metres per second per second (ms^{-2}).

A.2 Ambient Temperature

The ambient temperature sensor returns the room temperature in Celsius degrees (°C).

A.3 Bluetooth

Bluetooth is a technology that facilitates wireless communication and operates in the ISM band centred at 2.4 gigahertz. As a proximity sensor, we measure the Bluetooth devices in the vicinity (their names and MAC addresses).

A.4 Geomagnetic Rotation Vector (GRV)

The GRV sensor measures the rotation of the device using the device's magnetometer and accelerometer; it returns a vector containing the angles that the device is rotated in the x, y and z axes.

A.5 Global Positioning System (GPS)

The GPS sensor based a satellite-based global positioning and velocity measurement. A latitude and longitude pair is returned, representing a geographical location on Earth.

A.6 Gravity

The gravity sensor on mobile handsets measures the effect of Earth's gravity on the device, measured in metres per second per second (ms^{-2}).

A.7 Gyroscope

The gyroscope measures the rate of rotation of the device about the x, y and z axes; its units are radians per second ($rads^{-1}$).

A.8 Relative Humidity

The relative humidity sensor returns the percentage of the relative ambient humidity in the air.

A.9 Light

The light sensor measures the lighting conditions surrounding the mobile handset. Android measures this quantity in lux.

A.10 Linear Acceleration

The linear acceleration sensor measures the affect of a device's movement on itself; its units are metres per second per second (ms^{-2}).

A.11 Magnetic Field

The magnetic field sensor detects the Earth's magnetic field along three perpendicular axes x, y and z. Android measures these values in microteslas (μT).

A.12 Network Location

A latitude and longitude pair is returned, representing a geographical location on Earth.

A.13 Pressure

The pressure sensor measures the atmospheric pressure surrounding the mobile handset. It is measured in hectopascals (hPa).

A.14 Proximity

The proximity sensors detects distance, measured in centimetres. In many devices the sensor returns only a boolean value, declaring whether something is in close proximity to the device or not.

A.15 Rotation Vector

Rotation vector is a software sensor, similar to the GRV, but also incorporates the gyroscope. The returned values represent the angles which the device has rotated through the x, y and z axes.

A.16 Sound

For the sound sensor's measurement, we use the device's microphone to record the noise in the vicinity of the mobile handsets and retrieve the maximum amplitude that was sampled, every time it becomes available by the Android operating system.

A.17 WiFi

This sensor uses traditional WiFi to detect the networks in the vicinity of the mobile device. The MAC addresses and ESSIDs of the nearby networks are returned.

Appendix B

Protocol 1 – Scyther Script

```
1  usertype IRsequence;
2  hashfunction h;
3  usertype SessionKey;
4  secret Cert: Function;
5
6  protocol protPK(TI,TT) {
7    role TI {
8      fresh nti: Nonce;
9      var ntt: Nonce;
10     fresh IRSeq: IRsequence;
11     fresh IRTiming: Ticket;
12     var Y: Ticket;
13     fresh K: SessionKey;
14     recv_1(TT,TI, TT,ntt,Cert(TT));
15     send_2(TI,TT, {TI,TT,nti,ntt,K,IRSeq}pk(TT),{h(K,ntt)}sk(TI),
           Cert(TI));
16     send_3(TI,TT, {TI,TT,nti,ntt,IRTiming}K);
17     recv_4(TT,TI, {TI,TT,nti,ntt,Y,{h(nti)}sk(TT)}K);
18
19     claim(TI, Alive);
20     claim(TI, Secret, K);
21     claim(TI, Secret, nti);
22     claim(TI, Niagree);
23     claim(TI, Nisynch);
24     claim(TI, Secret, IRSeq);
25     claim(TI, Secret, IRTiming);
26     claim(TI, Secret, Y);
27   }
28
29   role TT {
30     var nti: Nonce;
31     fresh ntt: Nonce;
32     var X: Ticket;
33     var Y: Ticket;
34     fresh apprv: Ticket;
35     var K: SessionKey;
```

```
36      send_1(TT,TI, TT,ntt,Cert(TT));
37      recv_2(TI,TT, {TI,TT,nti,ntt,K,X}pk(TT),{h(K,ntt)}sk(TI),Cert(
            TI));
38      recv_3(TI,TT, {TI,TT,nti,ntt,Y}K);
39      send_4(TT,TI, {TI,TT,nti,ntt,apprv,{h(nti)}sk(TT)}K);
40
41      claim(TT, Alive);
42      claim(TT, Secret, K);
43      claim(TT, Secret, nti);
44      claim(TT, Niagree);
45      claim(TT, Nisynch);
46      claim(TT, Secret, X);
47      claim(TT, Secret, Y);
48      claim(TT, Secret, apprv);
49    }
50  }
```

Appendix C

Protocol 2 – Scyther Script

```
usertype IRsequence;
hashfunction h;

macro K = k(TI,TT);

symmetric-role protocol protSK(TI,TT) {
  role TI {
    fresh nti: Nonce;
    var ntt: Nonce;
    fresh IRSeq: IRsequence;
    fresh IRTiming: IRsequence;
    var Y: Ticket;
    recv_1(TT,TI, {TT,ntt}K);
    send_2(TI,TT, {TI,TT,nti,ntt,IRSeq}K);
    send_3(TI,TT, {TI,TT,nti,ntt,h(IRSeq),IRTiming}K);
    recv_4(TT,TI, {TI,TT,nti,ntt,Y,h(IRSeq)}K);

    claim(TI, Alive);
    claim(TI, Niagree);
    claim(TI, Nisynch);
    claim(TI, SKR, ntt);
    claim(TI, SKR, nti);
    claim(TI, Secret, IRSeq);
    claim(TI, Secret, IRTiming);
    claim(TI, Secret, Y);
  }

  role TT {
    var nti: Nonce;
    fresh ntt: Nonce;
    var X: Ticket;
    var Y: Ticket;
    fresh apprv: Ticket;
    send_1(TT,TI, {TT,ntt}K);
    recv_2(TI,TT, {TI,TT,nti,ntt,X}K);
    recv_3(TI,TT, {TI,TT,nti,ntt,h(X),Y}K);
```

```
37      send_4(TT,TI, {TI,TT,nti,ntt,apprv,h(X)}K);
38
39      claim(TT, Alive);
40      claim(TT, Niagree);
41      claim(TT, Nisynch);
42      claim(TT, SKR, ntt);
43      claim(TT, SKR, nti);
44      claim(TT, Secret, X);
45      claim(TT, Secret, Y);
46      claim(TT, Secret, apprv);
47    }
48  }
```

Appendix D

Both Devices Vibrate

Table D.1: Estimated EER Results for Machine Learning Algorithms (obtained by repeating 10-fold cross-validation 10 times)

Vibration Mode	Classifier				
	Random Forest	**Naïve Bayes**	**Decision Tree**	**Logistic Regression**	**Support Vector Machine**
Accelerometer					
VVVV	0.073 ± 0.030	0.115 ± 0.041	0.120 ± 0.047	0.200 ± 0.042	0.058 ± 0.026
VVV*V	0.075 ± 0.032	0.111 ± 0.035	0.122 ± 0.041	0.215 ± 0.046	0.067 ± 0.028
VNVV	0.051 ± 0.027	0.066 ± 0.026	0.079 ± 0.036	0.148 ± 0.039	0.049 ± 0.022
VVNV	0.068 ± 0.030	0.100 ± 0.039	0.099 ± 0.041	0.147 ± 0.042	0.084 ± 0.032
VNNV	0.076 ± 0.033	0.078 ± 0.038	0.099 ± 0.047	0.148 ± 0.036	0.079 ± 0.036
Gravity					
VVVV	0.497 ± 0.048	0.498 ± 0.042	0.500 ± 0.000	0.459 ± 0.041	0.500 ± 0.000
VVV*V	0.508 ± 0.053	0.505 ± 0.041	0.500 ± 0.000	0.540 ± 0.038	0.500 ± 0.000
VNVV	0.509 ± 0.053	0.487 ± 0.056	0.500 ± 0.000	0.453 ± 0.050	0.458 ± 0.050
VVNV	0.493 ± 0.054	0.492 ± 0.041	0.500 ± 0.000	0.510 ± 0.041	0.500 ± 0.000
VNNV	0.494 ± 0.053	0.483 ± 0.055	0.500 ± 0.000	0.478 ± 0.056	0.502 ± 0.062
Gyroscope					
VVVV	0.004 ± 0.009	0.004 ± 0.011	0.009 ± 0.014	0.021 ± 0.016	0.004 ± 0.009
VVV*V	0.002 ± 0.008	0.000 ± 0.000	0.008 ± 0.014	0.009 ± 0.013	0.003 ± 0.009
VNVV	0.004 ± 0.010	0.001 ± 0.005	0.010 ± 0.015	0.008 ± 0.014	0.003 ± 0.009
VVNV	0.005 ± 0.011	0.002 ± 0.008	0.010 ± 0.016	0.019 ± 0.017	0.003 ± 0.009
VNNV	0.002 ± 0.010	0.000 ± 0.003	0.009 ± 0.015	0.013 ± 0.016	0.002 ± 0.007
Linear Acceleration					
VVVV	0.000 ± 0.000	0.000 ± 0.000	0.021 ± 0.025	0.000 ± 0.000	0.000 ± 0.000
VVV*V	0.000 ± 0.000	0.000 ± 0.000	0.023 ± 0.027	0.000 ± 0.000	0.000 ± 0.000
VNVV	0.016 ± 0.019	0.015 ± 0.019	0.045 ± 0.032	0.057 ± 0.032	0.016 ± 0.018
VVNV	0.027 ± 0.021	0.028 ± 0.022	0.058 ± 0.034	0.090 ± 0.042	0.026 ± 0.022
VNNV	0.032 ± 0.025	0.042 ± 0.028	0.065 ± 0.042	0.088 ± 0.036	0.026 ± 0.023
Magnetic Field					
VVVV	0.082 ± 0.030	0.316 ± 0.060	0.218 ± 0.073	0.488 ± 0.062	0.042 ± 0.022
VVV*V	0.100 ± 0.029	0.315 ± 0.053	0.220 ± 0.068	0.530 ± 0.052	0.045 ± 0.026
VNVV	0.008 ± 0.012	0.312 ± 0.059	0.140 ± 0.058	0.441 ± 0.064	0.009 ± 0.013
VVNV	0.097 ± 0.033	0.311 ± 0.061	0.219 ± 0.100	0.484 ± 0.060	0.048 ± 0.023
VNNV	0.009 ± 0.014	0.322 ± 0.046	0.136 ± 0.052	0.444 ± 0.060	0.009 ± 0.014
Rotation Vector					
VVVV	0.007 ± 0.012	0.005 ± 0.017	0.031 ± 0.027	0.109 ± 0.047	0.008 ± 0.013
VVV*V	0.007 ± 0.012	0.005 ± 0.016	0.035 ± 0.027	0.114 ± 0.054	0.005 ± 0.010
VNVV	0.009 ± 0.012	0.010 ± 0.024	0.031 ± 0.026	0.109 ± 0.045	0.007 ± 0.012
VVNV	0.009 ± 0.013	0.009 ± 0.024	0.030 ± 0.026	0.116 ± 0.051	0.007 ± 0.014
VNNV	0.008 ± 0.012	0.009 ± 0.024	0.030 ± 0.025	0.116 ± 0.051	0.007 ± 0.014

*TT′ vibrates the same pattern as TI′

Appendix E

Transaction Terminal Vibrates

Table E.1: Estimated EER results for machine learning algorithms, obtained by repeating 10-fold cross-validation 10 times

Vibration Mode	Classifier				
	Random Forest	**Naïve Bayes**	**Decision Tree**	**Logistic Regression**	**Support Vector Machine**
Accelerometer					
VNVN	0.055 ± 0.025	0.111 ± 0.039	0.095 ± 0.041	0.168 ± 0.038	0.033 ± 0.019
VVVN	0.037 ± 0.026	0.082 ± 0.039	0.068 ± 0.033	0.148 ± 0.049	0.015 ± 0.016
VVV*N	0.121 ± 0.036	0.465 ± 0.271	0.216 ± 0.054	0.474 ± 0.049	0.092 ± 0.048
VVNN	0.138 ± 0.040	0.141 ± 0.049	0.174 ± 0.050	0.248 ± 0.055	0.146 ± 0.040
VNNN	0.131 ± 0.036	0.124 ± 0.041	0.164 ± 0.056	0.195 ± 0.043	0.133 ± 0.037
Gravity					
VNVN	0.503 ± 0.053	0.492 ± 0.041	0.500 ± 0.000	0.481 ± 0.042	0.500 ± 0.000
VVVN	0.496 ± 0.058	0.473 ± 0.054	0.500 ± 0.000	0.484 ± 0.056	0.483 ± 0.083
VVV*N	0.489 ± 0.052	0.447 ± 0.056	0.500 ± 0.000	0.471 ± 0.056	0.457 ± 0.055
VVNN	0.503 ± 0.059	0.481 ± 0.051	0.500 ± 0.000	0.484 ± 0.052	0.509 ± 0.081
VNNN	0.502 ± 0.053	0.527 ± 0.049	0.500 ± 0.000	0.508 ± 0.041	0.500 ± 0.000
Gyroscope					
VNVN	0.002 ± 0.006	0.000 ± 0.000	0.003 ± 0.008	0.001 ± 0.006	0.000 ± 0.003
VVVN	0.000 ± 0.000	0.001 ± 0.007	0.003 ± 0.008	0.003 ± 0.011	0.001 ± 0.005
VVV*N	0.000 ± 0.000	0.005 ± 0.011	0.003 ± 0.010	0.001 ± 0.004	0.000 ± 0.000
VVNN	0.000 ± 0.000	0.001 ± 0.004	0.003 ± 0.009	0.005 ± 0.011	0.001 ± 0.004
VNNN	0.002 ± 0.006	0.000 ± 0.000	0.002 ± 0.007	0.005 ± 0.011	0.003 ± 0.008
Linear Acceleration					
VNVN	0.000 ± 0.000	0.000 ± 0.000	0.012 ± 0.015	0.000 ± 0.000	0.001 ± 0.004
VVVN	0.007 ± 0.012	0.004 ± 0.009	0.031 ± 0.021	0.037 ± 0.022	0.003 ± 0.008
VVV*N	0.000 ± 0.000	0.000 ± 0.000	0.013 ± 0.020	0.000 ± 0.000	0.000 ± 0.000
VVNN	0.034 ± 0.025	0.034 ± 0.026	0.058 ± 0.033	0.087 ± 0.032	0.034 ± 0.023
VNNN	0.038 ± 0.022	0.035 ± 0.025	0.077 ± 0.037	0.095 ± 0.036	0.042 ± 0.025
Magnetic Field					
VNVN	0.020 ± 0.019	0.328 ± 0.051	0.146 ± 0.053	0.443 ± 0.064	0.014 ± 0.018
VVVN	0.117 ± 0.042	0.321 ± 0.058	0.203 ± 0.114	0.480 ± 0.053	0.058 ± 0.030
VVV*N	0.082 ± 0.035	0.125 ± 0.040	0.123 ± 0.048	0.183 ± 0.045	0.045 ± 0.025
VVNN	0.080 ± 0.028	0.122 ± 0.036	0.119 ± 0.041	0.176 ± 0.039	0.044 ± 0.022
VNNN	0.024 ± 0.020	0.116 ± 0.039	0.071 ± 0.041	0.171 ± 0.041	0.014 ± 0.018
Rotation Vector					
VNVN	0.006 ± 0.013	0.005 ± 0.016	0.022 ± 0.023	0.119 ± 0.046	0.006 ± 0.012
VVVN	0.005 ± 0.011	0.004 ± 0.013	0.020 ± 0.022	0.118 ± 0.048	0.005 ± 0.010
VVV*N	0.000 ± 0.002	0.014 ± 0.019	0.011 ± 0.015	0.096 ± 0.038	0.000 ± 0.000
VVNN	0.000 ± 0.000	0.014 ± 0.019	0.008 ± 0.015	0.100 ± 0.045	0.000 ± 0.000
VNNN	0.000 ± 0.000	0.014 ± 0.019	0.008 ± 0.012	0.099 ± 0.046	0.000 ± 0.000

*TT′ vibrates the same pattern as TI′

Appendix F

Transaction Instrument Vibrates

Table F.1: Estimated EER Results for Machine Learning Algorithms (obtained by repeating 10-fold cross-validation 10 times)

Vibration Mode	Classifier				
	Random Forest	Naïve Bayes	Decision Tree	Logistic Regression	Support Vector Machine
Accelerometer					
NVNV	0.101 ± 0.038	0.111 ± 0.043	0.125 ± 0.052	0.180 ± 0.040	0.104 ± 0.039
NVVV	0.026 ± 0.021	0.037 ± 0.028	0.069 ± 0.033	0.090 ± 0.031	0.012 ± 0.014
NVV*V	0.071 ± 0.030	0.115 ± 0.042	0.110 ± 0.041	0.188 ± 0.042	0.046 ± 0.026
NNVV	0.100 ± 0.036	0.128 ± 0.039	0.148 ± 0.054	0.245 ± 0.054	0.082 ± 0.030
NNNV	0.118 ± 0.040	0.158 ± 0.059	0.164 ± 0.049	0.228 ± 0.041	0.144 ± 0.037
Gravity					
NVNV	0.485 ± 0.056	0.487 ± 0.047	0.500 ± 0.000	0.502 ± 0.040	0.500 ± 0.000
NVVV	0.498 ± 0.056	0.506 ± 0.041	0.500 ± 0.000	0.497 ± 0.048	0.500 ± 0.000
NVV*V	0.501 ± 0.049	0.488 ± 0.042	0.500 ± 0.000	0.477 ± 0.039	0.500 ± 0.000
NNVV	0.514 ± 0.058	0.493 ± 0.051	0.500 ± 0.000	0.506 ± 0.049	0.482 ± 0.091
NNNV	0.516 ± 0.054	0.497 ± 0.051	0.500 ± 0.000	0.517 ± 0.054	0.501 ± 0.123
Gyroscope					
NVNV	0.000 ± 0.003	0.000 ± 0.000	0.005 ± 0.011	0.008 ± 0.014	0.000 ± 0.002
NVVV	0.002 ± 0.006	0.011 ± 0.015	0.004 ± 0.009	0.011 ± 0.014	0.006 ± 0.012
NVV*V	0.002 ± 0.008	0.003 ± 0.009	0.007 ± 0.013	0.003 ± 0.008	0.002 ± 0.007
NNVV	0.002 ± 0.006	0.000 ± 0.000	0.006 ± 0.012	0.004 ± 0.009	0.000 ± 0.004
NNNV	0.000 ± 0.003	0.000 ± 0.000	0.006 ± 0.012	0.005 ± 0.010	0.000 ± 0.003
Linear Acceleration					
NVNV	0.001 ± 0.004	0.000 ± 0.003	0.024 ± 0.023	0.007 ± 0.012	0.002 ± 0.006
NVVV	0.000 ± 0.000	0.000 ± 0.000	0.022 ± 0.021	0.000 ± 0.000	0.000 ± 0.000
NVV*V	0.000 ± 0.000	0.000 ± 0.000	0.017 ± 0.020	0.000 ± 0.000	0.000 ± 0.000
NNVV	0.002 ± 0.007	0.002 ± 0.007	0.017 ± 0.021	0.014 ± 0.015	0.004 ± 0.009
NNNV	0.009 ± 0.012	0.008 ± 0.014	0.020 ± 0.022	0.025 ± 0.019	0.010 ± 0.014
Magnetic Field					
NVNV	0.267 ± 0.051	0.273 ± 0.052	0.124 ± 0.109	0.435 ± 0.053	0.276 ± 0.048
NVVV	0.116 ± 0.039	0.119 ± 0.049	0.110 ± 0.062	0.197 ± 0.047	0.126 ± 0.045
NVV*V	0.254 ± 0.056	0.262 ± 0.060	0.198 ± 0.129	0.444 ± 0.056	0.268 ± 0.050
NNVV	0.045 ± 0.025	0.253 ± 0.056	0.171 ± 0.063	0.483 ± 0.066	0.010 ± 0.013
NNNV	0.046 ± 0.026	0.271 ± 0.060	0.143 ± 0.087	0.488 ± 0.064	0.011 ± 0.013
Rotation Vector					
NVNV	0.019 ± 0.022	0.003 ± 0.012	0.030 ± 0.026	0.041 ± 0.025	0.016 ± 0.020
NVVV	0.009 ± 0.015	0.010 ± 0.016	0.007 ± 0.012	0.003 ± 0.008	0.000 ± 0.000
NVV*V	0.017 ± 0.018	0.004 ± 0.012	0.024 ± 0.024	0.060 ± 0.024	0.017 ± 0.019
NNVV	0.019 ± 0.019	0.004 ± 0.015	0.030 ± 0.024	0.050 ± 0.029	0.015 ± 0.019
NNNV	0.017 ± 0.019	0.004 ± 0.015	0.030 ± 0.024	0.041 ± 0.025	0.015 ± 0.018

*TT′ vibrates the same pattern as TI′

Bibliography

[1] *A Cashless Future on the Horizon*. White Paper. VeriFone, 2010. URL: http://www.verifone.co.za/media/1420610/Verifone_Cashless_Future_Contactless.pdf.

[2] M. Alattar and M. Achemlal. "Host-Based Card Emulation: Development, Security, and Ecosystem Impact Analysis". In: *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS)*. Aug. 2014, pp. 506–509.

[3] A. Aldini et al. "Detection of repackaged mobile applications through a collaborative approach". In: *Concurrency and Computation: Practice and Experience* 27.11 (), pp. 2818–2838. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.3447. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3447.

[4] *AN10922: Symmetric key diversifications*. Tech. rep. NXP, Mar. 2010.

[5] *Android 8.0 Compatibility Definition*. Tech. rep. Google Inc., May 2018.

[6] Android API. *ConsumerIrManager*. https://developer.android.com/reference/android/hardware/ConsumerIrManager.html. 2017.

[7] Android API. *Host-based Card Emulation*. https://developer.android.com/guide/topics/connectivity/nfc/hce.html. 2017.

[8] *Android API Reference Documentation: Sensors Overview*. http://developer.android.com/guide/topics/sensors/sensors_overview.html.

[9] Android Developers. *Android keystore system*. https://developer.android.com/training/articles/keystore. 2018.

[10] Android Developers. *Android Open Source Project*. 2018. URL: https://source.android.com/.

[11] Android Developers. *ART and Dalvik*. https://source.android.com/devices/tech/dalvik/. 2018.

[12] Android Developers. *Platform Architecture*. https://developer.android.com/guide/platform/. 2018.

[13] AndroTotal. *Android FakeMarket Analysis*. http://blog.andrototal.org/post/73944579198/android-fakemarket-analysis. Jan. 2014.

[14] D. Arp et al. "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket". In: *NDSS*. 2014.

[15] D. Arp et al. *The Drebin Dataset*. 2016. URL: https://www.sec.cs.tu-bs.de/~danarp/drebin/.

[16] J.-D. Aussel and L. Rousseau. *pyscard – Python for smart cards*. 2014. URL: https://pyscard.sourceforge.io/.

[17] M. Benedettini. *Gist: key_generation.py*. https://gist.github.com/mbenedettini/1409585. 2011.

[18] L. Botezatu. *1.2 Percent of Google Play Store is Thief-Ware, Study Shows*. https://hotforsecurity.bitdefender.com/blog/1-2-percent-of-google-play-store-is-thief-ware-study-shows-7340.html. 2013.

[19] I. Boureanu, A. Mitrokotsa, and S. Vaudenay. "Towards Secure Distance Bounding". In: *Fast Software Encryption*. Springer. 2014, pp. 55–67.

[20] S. Brands and D. Chaum. "Distance-Bounding Protocols". English. In: *Advances in Cryptology — EUROCRYPT '93*. Ed. by T. Helleseth. Vol. 765. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1994, pp. 344–359. ISBN: 978-3-540-57600-6. URL: http://dx.doi.org/10.1007/3-540-48285-7_30.

[21] L. Breiman. "Random Forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[22] *CALYPSO FUNCTIONAL SPECIFICATION, Card Application*. Tech. rep. Calypso Networks Association, Apr. 2014.

[23] CardWerk. *ISO 7816 Part 4: Interindustry Commands for Interchange*. URL: http://www.cardwerk.com/smartcards/smartcard%5C_standard%5C_ISO7816-4.aspx.

[24] S. Chakradeo, B. Reaves, and W. Enck. "MAST: Triage for Market-scale Mobile Malware Analysis". In: *ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 2013, pp. 13–24.

[25] K. Chen, P. Liu, and Y. Zhang. "Achieving Accuracy and Scalability Simultaneously in Detecting Application Clones on Android Markets". In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. Hyderabad, India: ACM, 2014, pp. 175–186. ISBN: 978-1-4503-2756-5. URL: http://doi.acm.org/10.1145/2568225.2568286.

[26] K. Chen et al. "Finding Unknown Malice in 10 Seconds: Mass Vetting for New Threats at the Google-Play Scale". In: *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, 2015, pp. 659–674. ISBN: 978-1-931971-232. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/chen-kai.

[27] F. Cheng. "A Secure Mobile OTP Token". In: *Mobile Wireless Middleware, Operating Systems, and Applications: Third International Conference, Mobilware 2010, Chicago, IL, USA, June 30 - July 2, 2010. Revised Selected Papers*. Ed. by Y. Cai et al. Springer Berlin Heidelberg, 2010, pp. 3–16. ISBN: 978-3-642-17758-3.

[28] S.-s. Choi and S.-h. Cha. "A survey of Binary similarity and distance measures". In: *Journal of Systemics, Cybernetics and Informatics* (2010), pp. 43–48.

[29] W. W. Cohen, P. D. Ravikumar, and S. E. Fienberg. "A Comparison of String Distance Metrics for Name-Matching Tasks". In: *IJCAI-03 Workshop on Information Integration on the Web*. 2003, pp. 73–78.

[30] V. Consonni and R. Todeschini. "New similarity coefficients for binary data". In: *Match-Communications in Mathematical and Computer Chemistry* 68.2 (2012), p. 581.

[31] *Contactless Mobile Payments (finally) Gain Momentum*. Online Report. Deloitte, 2015. URL: http://www.deloitte.co.uk/tmtpredictions2015/contactless-mobile-payments-finally-gain-momentum/.

[32] J. H. Conway. *On Numbers and Games*. Ak Peters Series. Taylor & Francis, 2000. ISBN: 9781568811277.

[33] P. Coogan. *More fraudware headaches for the Android Marketplace*. https://www.symantec.com/connect/blogs/more-fraudware-headaches-android-marketplace. 2012.

[34] V. Coskun, B. Ozdenizci, and K. Ok. "A Survey on Near Field Communication (NFC) Technology". English. In: *Wireless Personal Communications* 71.3 (2013), pp. 2259–2294. ISSN: 0929-6212. URL: http://dx.doi.org/10.1007/s11277-012-0935-5.

[35] V. Costan and S. Devadas. "Intel SGX Explained". In: *IACR Cryptology ePrint Archive* 2016 (2016).

[36] C. Cremers et al. "Distance Hijacking Attacks on Distance Bounding Protocols". In: *Security and Privacy, 2012 IEEE Symposium on*. May 2012, pp. 113–127.

[37] C. J. Cremers. *Scyther tool: Compromising Adversaries version*. https://people.cispa.io/cas.cremers/scyther/compromise/index.html. 2014.

[38] C. J. Cremers. *Scyther User Manual*. https://github.com/cascremers/scyther/blob/master/gui/scyther-manual.pdf. 2014.

[39] C. J. Cremers. "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols". In: *International Conference on Computer Aided Verification*. Springer. 2008, pp. 414–418.

[40] J. Crussell, C. Gibler, and H. Chen. "AnDarwin: Scalable Detection of Semantically Similar Android Applications". English. In: *Computer Security - ESORICS 2013*. Vol. 8134. LNCS. Springer, 2013, pp. 182–199.

[41] J. Crussell, C. Gibler, and H. Chen. "Attack of the Clones: Detecting Cloned Applications on Android Markets". In: *Computer Security - ESORICS 2012*. Vol. 7459. LNCS. Springer, 2012, pp. 37–54.

[42] R. Datta, J. Li, and J. Z. Wang. "Content-based Image Retrieval: Approaches and Trends of the New Age". In: *the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval*. Hilton, Singapore: ACM, 2005, pp. 253–262.

[43] Y. Desmedt. "Major security problems with the 'unforgeable'(feige)-fiat-shamir proofs of identity and how to overcome them". In: *Proceedings of SECURICOM*. Vol. 88. 1988, pp. 15–17.

[44] Y. Desmedt, C. Goutier, and S. Bengio. "Special Uses and Abuses of the Fiat-Shamir Passport Protocol". In: *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*. Vol. 293. Lecture Notes in Computer Science. Springer, 1987, pp. 21–39.

[45] A. Desnos and G. Gueguen. "New "open source" step in Android application analysis". In: PacSec Conference 2012, 2012.

[46] *Digital Payments Solutions Industry Considerations*. Online Report. The UK Cards Association, June 2017. URL: http://www.theukcardsassociation.org.uk/wm_documents/Digital%20Wallets%20-%20Industry%20Considerations%20Outline.pdf.

[47] S. Drimer and S. J. Murdoch. "Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks." In: *USENIX Security*. Ed. by N. Provos. USENIX Association, 2007.

[48] K. O. Elish et al. "Profiling user-trigger dependence for Android malware detection". In: *Computers & Security* 49 (2015), pp. 255–273. ISSN: 0167-4048. URL: http://www.sciencedirect.com/science/article/pii/S0167404814001631.

[49] M. Emms et al. "Harvesting High Value Foreign Currency Transactions from EMV Contactless Credit Cards without the PIN". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2014, pp. 716–726.

[50] *EMV Contactless Specifications for Payment Systems: Book A - Architecture and General Requirements*. Spec V2.6. EMVCo, LLC, Apr. 2016.

[51] *EMV Contactless Specifications for Payment Systems: Book D - EMV Contactless Communication Protocol Specification*. Spec V2.6. EMVCo, LLC, Mar. 2016.

[52] EMVCo. *EMV Integrated Circuit Card, Specifications for Payment Systems, Book 2, Security and Key Management, Version 4.3*. Nov. 2011.

[53] A. Francillon, B. Danev, and S. Capkun. "Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars". In: *Network & Distributed System Security*. NDSS. The Internet Society, Feb. 2011.

[54] L. Francis et al. "Practical NFC Peer-to-peer Relay Attack Using Mobile Phones". In: *Proceedings of the 6th International Conference on Radio Frequency Identification: Security and Privacy Issues*. RFIDSec'10. Istanbul, Turkey: Springer-Verlag, 2010, pp. 35–49.

[55] L. Francis et al. "Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones". In: *IACR Cryptology Archive* 2011 (2011), p. 618.

[56] E. Frank, M. A. Hall, and I. H. Witten. *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*. 4th ed. Burlington, MA: Morgan Kaufmann, 2016.

[57] G. P. Hancke and M. G. Kuhn. "An RFID Distance Bounding Protocol". In: *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*. Sept. 2005, pp. 67–73.

[58] M. von Gagern et al. *GNU Wdiff*. https://www.gnu.org/software/wdiff/. 2014.

[59] S. Gambs, C. E. R. K. Lassance, and C. Onete. "The Not-so-Distant Future: Distance-Bounding Protocols on Smartphones". In: *Smart Card Research and Advanced Applications*. Ed. by N. Homma and M. Medwed. Cham: Springer International Publishing, 2016, pp. 209–224. ISBN: 978-3-319-31271-2.

[60] E. Garcia. *A Tutorial on Distance and Similarity*. http://www.minerazzi.com/tutorials/distance-similarity-tutorial.pdf. 2018.

[61] C. Gibler et al. "AdRob: Examining the Landscape and Impact of Android Application Plagiarism". In: *the 11th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys '13. Taipei, Taiwan: ACM, 2013, pp. 431–444.

[62] Google. *Application Fundamentals*. https://developer.android.com/guide/components/fundamentals. 2018.

[63] Google. *Getting Started with the NDK*. https://developer.android.com/ndk/guides/. 2018.

[64] Google. *Google Play Protect*. https://www.android.com/play-protect/. 2018.

[65] B. Gruver. *smali/baksmali*. https://github.com/JesusFreke/smali. 2018.

[66] G. Haken et al. "Evaluation of Apple iDevice Sensors As a Potential Relay Attack Countermeasure for Apple Pay". In: *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*. CPSS '17. Abu Dhabi, United Arab Emirates: ACM, 2017, pp. 21–32. ISBN: 978-1-4503-4956-7.

[67] T. Halevi and N. Saxena. "Acoustic Eavesdropping Attacks on Constrained Wireless Device Pairing". In: *IEEE Transactions on Information Forensics and Security* 8.3 (Mar. 2013), pp. 563–577. ISSN: 1556-6013.

[68] T. Halevi et al. "Secure Proximity Detection for NFC Devices Based on Ambient Sensor Data". English. In: *Computer Security – ESORICS 2012*. Ed. by S. Foresti, M. Yung, and F. Martinelli. LNCS. Springer, 2012. ISBN: 978-3-642-33166-4.

[69] G. P. Halkes et al. *dwdiff*. https://os.ghalkes.nl/dwdiff.html. 2017.

[70] G. P. Hancke. "Design of a Secure Distance-bounding Channel for RFID". In: *J. Netw. Comput. Appl.* 34.3 (May 2011), pp. 877–887. ISSN: 1084-8045. URL: http://dx.doi.org/10.1016/j.jnca.2010.04.014.

[71] G. P. Hancke. "Distance-bounding for RFID: Effectiveness of 'terrorist fraud' in the presence of bit errors". In: *2012 IEEE International Conference on RFID-Technologies and Applications (RFID-TA)*. Nov. 2012, pp. 91–96.

[72] G. Hancke, K. Mayes, and K. Markantonakis. "Confidence in Smart Token Proximity: Relay Attacks Revisited". In: *Computers & Security* 28.7 (2009), pp. 615–627. ISSN: 0167-4048. URL: http://www.sciencedirect.com/science/article/pii/S0167404809000595.

[73] G. P. Hancke. "Practical attacks on proximity identification systems". In: *Security and Privacy, 2006 IEEE Symposium on*. IEEE Computer Society, May 2006, pp. 328–333. ISBN: 0-7695-2574-1. URL: http://dblp.uni-trier.de/db/conf/sp/sp2006.html#Hancke06.

[74] G. P. Hancke and M. G. Kuhn. "Attacks on Time-of-flight Distance Bounding Channels". In: *Proceedings of the First ACM Conference on Wireless Network Security*. WiSec '08. ACM. Alexandria, VA, USA: ACM, 2008, pp. 194–202. ISBN: 978-1-59593-814-5. URL: http://doi.acm.org/10.1145/1352533.1352566.

[75] S. Hanna et al. "Juxtapp: A Scalable System for Detecting Code Reuse Among Android Applications". In: *the 9th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. DIMVA'12. Heraklion, Crete, Greece: Springer, 2013, pp. 62–81.

[76] L.-A. C. Hayek. "Analysis of amphibian biodiversity data". In: (1994).

[77] W. B. Hayes. "Some Sampling Properties of the Fager Index for Recurrent Species Groups". In: *Ecology* 59.1 (1978), pp. 194–196. ISSN: 00129658, 19399170. URL: http://www.jstor.org/stable/1936649.

[78] *HCE security implications –Analyzing the security aspects of HCE: White paper*. Tech. rep. UL, Jan. 2014.

[79] G. Henderson. *Wiring Pi - GPIO Interface library for the Raspberry Pi*. http://wiringpi.com/. 2018.

[80] *Host Card Emulation (HCE) 101: White paper*. Tech. rep. Smart Card Alliance, Aug. 2014.

[81] *How to Optimize the Consumer Contactless Experience? The Perfect Tap*. Tech. rep. MasterCard, 2014.

[82] W. Hu et al. "MIGDroid: Detecting APP-Repackaging Android malware via method invocation graph". In: *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*. Aug. 2014, pp. 1–7.

[83] Y.-C. Hu, A. Perrig, and D. B. Johnson. "Wormhole attacks in wireless networks". In: *IEEE Journal on Selected Areas in Communications* 24.2 (Feb. 2006), pp. 370–380. ISSN: 0733-8716.

[84] H. Huang et al. "A Framework for Evaluating Mobile App Repackaging Detection Algorithms". English. In: *Trust and Trustworthy Computing*. Vol. 7904. LNCS. Springer, 2013, pp. 169–186.

[85] J. Hunter et al. *Matplotlib*. 2002. URL: https://matplotlib.org/.

[86] IBM Knowledge Center. *Distances Similarity Measures for Binary Data*. https://www.ibm.com/support/knowledgecenter/en/SSLVMB_24.0.0/spss/base/cmd_proximities_sim_measure_binary.html. 2011.

[87] *Cards and security devices for personal identification – Contactless proximity objects – Part 1: Physical characteristics*. Standard. International Organization for Standardization, Apr. 2018.

[88] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. "Fast Multiresolution Image Querying". In: *the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '95. USA: ACM, 1995, pp. 277–286.

[89] R. Jin et al. "MagPairing: Pairing Smartphones in Close Proximity Using Magnetometers". In: *IEEE Transactions on Information Forensics and Security* 11.6 (June 2016), pp. 1306–1320. ISSN: 1556-6013.

[90] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001. URL: http://www.scipy.org/.

[91] J.-H. Jung et al. "Repackaging Attack on Android Banking Applications and Its Countermeasures". English. In: *Wireless Personal Communications* 73.4 (2013), pp. 1421–1437.

[92] Juniper Research Ltd. *Apple Pay Accounts for 1 in 2 OEM Pay Users Globally, Reaching 200 Million by 2020*. https://www.juniperresearch.com/press/press-releases/apple-pay-accounts-for-1-in-2-oem-pay-users. June 2018.

[93] Juniper Research Ltd. *NFC Mobile Ticketing Users to Reach 375 Million Users by 2022, Despite a Slow Start*. https://www.juniperresearch.com/press/press-releases/nfc-mobile-ticketing-users-to-reach-375-million-us. Dec. 2017.

[94] N. Karapanos et al. "Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound". In: *24th USENIX Security Symposium*. Washington, D.C.: USENIX Association, Aug. 2015. ISBN: 978-1-931971-232.

[95] P. Karulis. *PyBluez*. 2015. URL: https://pypi.org/project/PyBluez/.

[96] M. Kassner. *Google Play: Android's Bouncer can be pwned*. https://www.techrepublic.com/blog/it-security/-google-play-androids-bouncer-can-be-pwned/. 2012.

[97] Z. Kfir and A. Wool. "Picking Virtual Pockets using Relay Attacks on Contactless Smartcard Systems". In: *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*. IEEE. 2005, pp. 47–58.

[98] Klinger, Evan and Starkweather David. *pHash: The open source perceptual hash library*. http://www.phash.org/. 2018.

[99] T. Korak and M. Hutter. "On the power of active relay attacks using custom-made proxies". In: *2014 IEEE International Conference on RFID (IEEE RFID)*. Apr. 2014, pp. 126–133.

[100] S. M. Kywe et al. "Detecting Camouflaged Applications on Mobile Application Markets". In: *Information Security and Cryptology - ICISC 2014*. Ed. by J. Lee and J. Kim. Cham: Springer International Publishing, 2015, pp. 241–254. ISBN: 978-3-319-15943-0.

[101] C. E. R. K. Lassance. "Implementing Distance-bounding protocols on Android smartphones". MA thesis. INRIA/Centrale Supélec, 2015.

[102] A. Leamas et al. *LIRC – Linux Infrared Remote Control*. http://www.lirc.org/. 2016.

[103] M. Lindorfer et al. "AndRadar: Fast Discovery of Android Applications in Alternative Markets". English. In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. LNCS. Springer, 2014, pp. 51–71.

[104] D. C. Litzenberger. *pycrypto*. 2013. URL: https://pypi.org/project/pycrypto/.

[105] H. Lockheimer. *Android and Security*. http://googlemobile.blogspot.co.uk/2012/02/android-and-security.html. 2012.

[106] D. Ma et al. "Location-Aware and Safer Cards: Enhancing RFID Security and Privacy via Location Sensing". In: *IEEE TDSC* 10.2 (2013), pp. 57–69.

[107] W. I. MacGregor et al. *A Recommendation for the Use of PIV Credentials in Physical Access Control Systems (PACS)*. Special Publication (NIST SP) – 800–116. NIST, Nov. 2008.

[108] G. Madlmayr et al. "NFC devices: Security and privacy". In: *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*. IEEE. 2008, pp. 642–647.

[109] L. Malisa, K. Kostiainen, and S. Capkun. "Detecting Mobile Application Spoofing Attacks by Leveraging User Visual Similarity Perception". In: *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. CODASPY '17. Scottsdale, Arizona, USA: ACM, 2017, pp. 289–300. ISBN: 978-1-4503-4523-1. URL: http://doi.acm.org/10.1145/3029806.3029819.

[110] L. Malisa et al. "Mobile Application Impersonation Detection Using Dynamic User Interface Extraction". In: *Computer Security – ESORICS 2016*. Ed. by I. Askoxylakis et al. Cham: Springer International Publishing, 2016, pp. 217–237. ISBN: 978-3-319-45744-4.

[111] D. Maltoni et al. *Handbook of Fingerprint Recognition*. Springer Science & Business Media, 2009.

[112] *MasterCard Contactless Performance Requirement*. Online. MasterCard, Mar. 2014.

[113] K. E. Mayes and K. Markantonakis. "Mobile Communication Security Controllers an Evaluation Paper". In: *Inf. Secur. Tech. Rep.* 13.3 (Aug. 2008), pp. 173–192. ISSN: 1363-4127. URL: http://dx.doi.org/10.1016/j.istr.2008.09.004.

[114] S. McConnell. *Code Complete*. 2nd ed. Microsoft Press, June 2004. ISBN: 0735619670.

[115] W. McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by S. van der Walt and J. Millman. 2010, pp. 51–56.

[116] M. Mehrnezhad, F. Hao, and S. F. Shahandashti. "Tap-Tap and Pay (TTP): Preventing Man-In-The-Middle Attacks in NFC Payment Using Mobile Sensors". In: *2nd International Conference on Research in Security Standardisation (SSR'15)*. Oct. 2014.

[117] *MF1P(H)x1y1, MIFARE Plus EV1, Rev. 2*. Preliminary short data sheet. NXP, Apr. 2016.

[118] MINERAZZI. *Binary Similarity Calculator*. http://www.minerazzi.com/tools/similarity/binary-similarity-calculator.php. 2016.

[119] OpenCV. *OpenCV Library*. https://opencv.org/. 2018.

[120] Y. Oren, D. Schirman, and A. Wool. "Range Extension Attacks on Contactless Smart Cards". In: *Computer Security – ESORICS 2013*. Ed. by J. Crampton, S. Jajodia, and K. Mayes. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 646–663. ISBN: 978-3-642-40203-6.

[121] OWASP. *Mobile Top 10 2016-Top 10*. https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10. 2016.

[122] OWASP. *OWASP Mobile Security Project - Top Ten Mobile Risks*. https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks. 2014.

[123] S. Pannifer, D. Clark, and D. Birch. *ISO 7816 Part 4: Interindustry Commands for Interchange*. Tech. rep. June 2014.

[124] J. C. Platt. "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines". In: *ADVANCES IN KERNEL METHODS-SUPPORT VECTOR LEARNING*. 1998.

[125] J. R. Quinlan. *C 4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA: Morgan Kaufmann, 1993.

[126] A. Ranganathan et al. "Design and Implementation of a Terrorist Fraud Resilient Distance Bounding System". In: *Computer Security – ESORICS 2012*. Ed. by S. Foresti, M. Yung, and F. Martinelli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 415–432. ISBN: 978-3-642-33167-1.

[127] W. Rankli and W. Effing. *Smart Card Handbook*. 4th ed. Wiley, June 2010. ISBN: 978-0-470-74367-6.

[128] K. B. Rasmussen and S. Capkun. "Realization of RF Distance Bounding." In: *USENIX Security Symposium*. 2010, pp. 389–402.

[129] J. Riegelsberger. "Trust in Mediated Interactions". PhD thesis. University College London, July 2005.

[130] M. Roland, J. Langer, and J. Scharinger. "Practical Attack Scenarios on Secure Element-Enabled Mobile Devices". In: *2012 4th International Workshop on Near Field Communication*. Mar. 2012, pp. 19–24.

[131] M. Roland, J. Langer, and J. Scharinger. "Applying relay attacks to Google Wallet". In: *Near Field Communication (NFC), 2013 5th International Workshop on*. Feb. 2013, pp. 1–6.

[132] M. Roland, J. Langer, and J. Scharinger. "Relay Attacks on Secure Element-Enabled Mobile Devices". In: *Information Security and Privacy Research: 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012, Heraklion, Crete, Greece, June 4-6, 2012. Proceedings*. Ed. by D. Gritzalis, S. Furnell, and M. Theoharidou. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–12. ISBN: 978-3-642-30436-1. URL: http://dx.doi.org/10.1007/978-3-642-30436-1_1.

[133] scikit-learn. *scikit-learn: Machine Learning in Python*. https://scikit-learn.org/stable/index.html. 2018.

[134] H. Shahriar and V. Clincy. "Detection of repackaged Android Malware". In: *The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)*. Dec. 2014, pp. 349–354.

[135] Y. Shao et al. "Towards a Scalable Resource-driven Approach for Detecting Repackaged Android Applications". In: *the 30th Annual Computer Security Applications Conference*. New Orleans, Louisiana, USA: ACM, 2014.

[136] C. Shepherd, R. N. Akram, and K. Markantonakis. "Towards Trusted Execution of Multi-modal Continuous Authentication Schemes". In: *Proceedings of the 32nd Symposium on Applied Computing*. ACM. 2017, pp. 1444–1451.

[137] B. Shrestha et al. "Sensor-based Proximity Detection in the Face of Active Adversaries". In: *IEEE Transactions on Mobile Computing* (2018), pp. 1–1. ISSN: 1536-1233.

[138] B. Shrestha et al. "Contextual Proximity Detection in the Face of Context-Manipulating Adversaries". In: *CoRR* abs/1511.00905 (2015). URL: http://arxiv.org/abs/1511.00905.

[139] B. Shrestha et al. "Drone to the Rescue: Relay-Resilient Authentication using Ambient Multi-sensing". In: *Financial Cryptography and Data Security*. Springer, 2014, pp. 349–364.

[140] B. Shrestha et al. "The Sounds of the Phones: Dangers of Zero-Effort Second Factor Login Based on Ambient Audio". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: ACM, 2016, pp. 908–919. ISBN: 978-1-4503-4139-4.

[141] Stata. *Stata Manuals*. https://www.stata.com/manuals13/mvmeasure_option.pdf. 2007.

[142] StatCounter. *Operating System Market Share Worldwide — June 2018*. http://gs.statcounter.com/os-market-share. June 2018.

[143] Statista. *Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 1st quarter 2018*. https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/. June 2018.

[144] Statista. *Proximity mobile payment transaction value in the United States from 2015 to 2021 (in billion U.S. dollars)*. https://www.statista.com/statistics/244475/proximity-mobile-payment-transaction-value-in-the-united-states/. July 2017.

[145] M. Sun, M. Li, and J. C. Lui. "DroidEagle: Seamless Detection of Visually Similar Android Apps". In: *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. WiSec '15. 2015.

[146] *The Future of Ticketing: Paying for Public Transport Journeys Using Visa Cards in the 21st Century*. Whitepaper. VISA, Jan. 2013.

[147] K. Tian et al. "Detection of Repackaged Android Malware with Code-Heterogeneity Features". In: *IEEE Transactions on Dependable and Secure Computing* PP.99 (2017), pp. 1–1. ISSN: 1545-5971.

[148] R. Todeschini et al. "Similarity Coefficients for Binary Chemoinformatics Data: Overview and Extended Comparison Using Simulated and Real Data Sets". In: *Journal of Chemical Information and Modeling* 52.11 (2012). PMID: 23078167, pp. 2884–2901.

[149] *Transactions Acceptance Device Guide (TADG)*. Specification Version 3.1. VISA, Nov. 2016.

[150] *Transit and Contactless Open Payments: An Emerging Approach for Fare Collection*. White Paper. Smart Card Alliance Transportation Council, Nov. 2011.

[151] R. Trujillo-Rasua, B. Martin, and G. Avoine. "The Poulidor distance-bounding protocol". In: *Radio Frequency Identification: Security and Privacy Issues*. Springer, 2010, pp. 239–257.

[152] H. T. T. Truong et al. "Comparing and Fusing Different Sensor Modalities for Relay Attack Resistance in Zero-Interaction Authentication". In: *Pervasive Computing and Communications, 2014 IEEE International Conference on*. IEEE. 2014, pp. 163–171.

[153] H. T. T. Truong et al. "Using contextual co-presence to strengthen Zero-Interaction Authentication: Design, integration and usability". In: *Pervasive and Mobile Computing* 16, Part B (2015). Selected Papers from the Twelfth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2014), pp. 187–204. ISSN: 1574-1192. URL: http://www.sciencedirect.com/science/article/pii/S1574119214001771.

[154] R. Tulloss. "Assessment of Similarity Indices for Undesirable Properties and a New Tripartite Similarity Index". In: *Mycology in sustainable development: expanding concepts, vanishing borders* (1997), p. 122.

[155] *UK Card Payments Summary 2016*. Tech. rep. The UK Cards Association, 2016.

[156] A. Umar, K. Mayes, and K. Markantonakis. "Performance Variation in Host-Based Card Emulation Compared to a Hardware Security Element". In: *Mobile and Secure Services, 2015 First Conference on*. IEEE. 2015, pp. 1–6.

[157] P. Urien and S. Piramuthu. "Elliptic curve-based RFID/NFC authentication with temperature sensor input for relay attacks". In: *Decision Support Systems* 59 (2014), pp. 28–36. ISSN: 0167-9236.

[158] A. Varshavsky et al. "Amigo: Proximity-Based Authentication of Mobile Devices". English. In: *UbiComp 2007*. Ed. by J. Krumm et al. LNCS. Springer, 2007, pp. 253–270. ISBN: 978-3-540-74852-6.

[159] R. Verdult and F. Kooman. "Practical Attacks on NFC Enabled Cell Phones". In: *Near Field Communication (NFC), 2011 3rd International Workshop on*. Feb. 2011, pp. 77–82.

[160] T. Vidas and N. Christin. "Sweetening Android Lemon Markets: Measuring and Combating Malware in Application Marketplaces". In: *Data and Application Security and Privacy* (2013), pp. 197–207.

[161] J. Vila and R. J. Rodríguez. "Practical Experiences on NFC Relay Attacks with Android". In: *Radio Frequency Identification*. Ed. by S. Mangard and P. Schaumont. Cham: Springer International Publishing, 2015, pp. 87–103. ISBN: 978-3-319-24837-0.

[162] S. van der Walt, S. C. Colbert, and G. Varoquaux. "The NumPy Array: A Structure for Efficient Numerical Computation". In: *Computing in Science Engineering* 13.2 (Mar. 2011), pp. 22–30. ISSN: 1521-9615.

[163] W. E. Winkler. "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage". In: *the Section on Survey Research*. Wachington, DC, 1990, pp. 354–359.

[164] M. Xu et al. "Toward Engineering a Secure Android Ecosystem: A Survey of Existing Techniques". In: *ACM Comput. Surv.* 49.2 (Aug. 2016), 38:1–38:47. ISSN: 0360-0300. URL: http://doi.acm.org/10.1145/2963145.

[165] J. Zhai and J. Su. *What are you doing? - DSEncrypt Malware*. https://www.fireeye.com/blog/threat-research/2014/06/what-are-you-doing-dsencrypt-malware.html. 2014.

[166] F. Zhang et al. "ViewDroid: Towards Obfuscation-resilient Mobile Application Repackaging Detection". In: *the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks*. Oxford, United Kingdom: ACM, 2014.

[167] Y. Zhauniarovich et al. "FSquaDRA: Fast Detection of Repackaged Applications". English. In: *Data and Applications Security and Privacy XXVIII*. LNCS. Springer, 2014, pp. 130–145.

[168] M. Zheng, M. Sun, and J. C. S. Lui. "DroidAnalytics: A Signature Based Analytic System to Collect, Extract, Analyze and Associate Android Malware". In: *Proceedings - 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013*. 2013, pp. 163–171. arXiv: arXiv:1302.7212v3.

[169] W. Zhou, X. Zhang, and X. Jiang. "AppInk: Watermarking Android Apps for Repackaging Deterrence". In: *Proceedings of the 8th ACM SIGSAC* (2013), pp. 1–12.

[170] W. Zhou et al. "Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces". In: *the second ACM conference on Data and Application Security and Privacy - CODASPY '12*. 2012, pp. 317–326.

[171] Y. Zhou and X. Jiang. "Dissecting Android Malware: Characterization and Evolution". In: *Security and Privacy, 2012 IEEE Symposium on*. May 2012, pp. 95–109.