2016

Context-Enhanced Mobile Device Authorization and Authentication

Babins Shrestha
University of Alabama at Birmingham

CONTEXT-ENHANCED MOBILE DEVICE AUTHORIZATION AND AUTHENTICATION

by

BABINS SHRESTHA

NITESH SAXENA, COMMITTEE CHAIR
N. ASOKAN
VIR V. PHOHA
ALAN SPRAGUE
CHENGCUI ZHANG

A DISSERTATION

Submitted to the graduate faculty of The University of Alabama at Birmingham,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

BIRMINGHAM, ALABAMA

2016

“There are so many doors to open. I am impatient to begin.”

– Charlie Gordan

—DANIEL KEYES, Flowers for Algernon

CONTEXT-ENHANCED MOBILE DEVICE AUTHORIZATION AND AUTHENTICATION

BABINS SHRESTHA

COMPUTER AND INFORMATION SCIENCES

ABSTRACT

Mobile devices (e.g., smartphones and tablets) are pervasive today, continuously opening up immense opportunities for everyday users. Their burgeoning popularity, however, brings forth various security and privacy threats. One well-established threat is of mobile malware (a form of insider attack) – malicious apps that may surreptitiously misuse the sensitive resources and services available on the device. Other threats relate to unauthorized access of the device (outsider attacks) by a malicious entity in close physical proximity to the device, or having (temporary or permanent) physical possession of the device. The traditional defensive mechanisms, such as existing anti-virus software, distance-bounding protocols or passwords, are not sufficient to defeat these threats.

This dissertation work explores the notion of "context" — a potentially unique signature of a benign usage scenario — to address insider-outsider attacks against mobile devices without undermining the overall usability of these devices. Our proposed defense system automatically detects the presence of a valid context using the information acquired by device's many on-board sensors; the absence of such a context being indicative of malicious usage. Depending upon the application scenario, we elicit the context provided, explicitly or transparently, by the device user (e.g., a hand gesture or body movement), or captured from the device's ambient environmental attributes (e.g., audio, temperature or altitude). When applicable, we use machine learning techniques and sensor fusion approaches towards designing a highly robust contextual mobile security system.

To be specific, this dissertation work comprises four parts: (1) enhancing mobile app authorization using implicit/explicit context, (2) enhancing user authentication using transparent implicit context, (3) enhancing co-presence detection using environmental context, and (4) strengthening the contextual security adversarial models and evaluating the context detection systems against such strong models.

In the first part, we present the design, implementation and evaluation of our contextual security mechanisms to defeat mobile malware attacks against prominent phone resources/services,

namely, phone calls, camera and NFC payments. We use explicit as well as implicit context to detect user-friendly explicit gestures or transparent gesture so as to ascertain if the app requesting the permission to a sensitive resource is legitimate (and not malicious). In the second part, we present the design, implementation and evaluation of schemes to authenticate users transparently in the case of mobile (NFC) payments and zero-interaction authentication systems. In the third part, we present the design, implementation and evaluation of our co-presence detection system using different environmental context to thwart outsider "relay attacks" against mobile zero-interaction authentication systems and mobile payment systems. In the fourth part, we stretch the limits of the contextual security threat model to incorporate adversaries who may be capable of actively manipulating the context or underlying sensor data (internally or externally). Further, we present our insights to defend against such strong adversaries.

DEDICATION

My parents,

Sudeshna Shrestha (wife),

Sushban and Sanim Shrestha (brothers).

ACKNOWLEDGMENT

Firstly, I would like to express my deepest gratitude to my adviser/committee chair, Dr. Nitesh Saxena, for the continuous support of my PhD study and related research, for his motivation and patience. His guidance helped me all the time of research and writing of this dissertation. His push to achieve more helped me develop as a better researcher. I could not have imagined having a better adviser and mentor for my Ph.D study.

Besides my adviser, I would like to thank my committee members (Drs. Asokan, Phoha, Sprague and Zhang) for dedicating time to evaluating this dissertation and providing me insightful comments and feedback.

Many thanks go to my co-authors Dr. Asokan and Dr. Truong from Finland. This dissertation would not exist without them. The unending meetings, discussions and sleepless nights during deadline days enriched our publications.

My sincere thanks also goes to my other collaborators (UAB: Anders Borg, Cooper Filby, Justin Harrison, Manar Mohamed, Maliheh Shirvanian, Prakash Shrestha; Aalto University, Finland: Sandeep Tamrakar; U of Helsinki, Finland: Xiang Gao, Dr. Nurmi; Univ of Michigan, Dearborn: Haoyu Li, Dr. Zhu, Dr. Ma), and all my colleagues at UAB. Moreover, I would like to thank all the members of the SPIES team at UAB. I made so many great friends working with this team. I would like to thank all of my colleagues for their support and beautiful memories.

This research has been funded by awards/grants from Google and NSF. I would like to thank the sponsors for funding our research.

Last but not the least, I would like to thank my family: my parents, my wife and my brothers for supporting me spiritually throughout writing this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Smart devices, such as smart phones and tablets, have become ubiquitous. Such a popularity of smart devices has attracted numerous developers to build a wide variety of applications for these devices. The manufacturers and OS developers continue to provide new capabilities to these devices in the form of hardware (e.g., sensors) and software (enhanced SDK). With different sensors embedded onto these devices, both developers and researchers from different fields have proposed new ideas and implemented applications targeting different application scenarios. For example, many smartphones have already incorporated Near Field Communication (NFC) Chips [1]. With this, apps like Google wallet [2] and Apple pay [3] have emerged which enable mobile users to make payments using their mobile devices. The developers also use inertial sensors, such as accelerometers and gyroscopes, to assist elderly people [4, 5] and people in need [6, 7]. In general, there are a myriad of applications utilizing mobile device sensors for health, exercise, games, alerts, and so on.

1.1 Security and Privacy Threats

The popularity of smart devices has also led to the various security and privacy threats. The integration of new hardware to these devices has lured malicious attackers and cyber criminals. They attempt to either misuse the sensitive resources provided by these devices by gaining access to resources, or exploiting the sensors which do not require explicit access permissions. For example, the addition of NFC chip as a reader enables apps to scan the NFC and RFID cards nearby. The malware developers have exploited this vulnerability to leech the sensitive information from the nearby NFC devices and RFID cards. Similarly, cyber criminals may also externally extract the information stored on the NFC chips using malicious readers since these chips respond promiscuously to any readers. To be specific, in this dissertation, we focus on two different kinds of attacks: (1) *insider attacks*: threats generating from within the devices in the form of malware and spyware not only compromising the device but also compromising the other nearby devices and information at the periphery of the device, and (2) *outsider attacks*:

threats generating from outside of the device in the form of unauthorized NFC transaction during theft/loss of device or unauthorized NFC reading, relay attacks with colluding attackers, and physical control or theft of devices.

1.1.1 Insider Attacks

The number and scope of malware targeting different smartphones has burgeoned in recent years [8–13]. Primary reason for the device to get infected is because users often download paid apps from untrusted sources for free, many of which may contain hidden malicious code. Once installed, such malwares can exploit the device in different ways [8]. For example, they may make premium rate phone calls, or send SMS with malicious links to user's friends, or make NFC payments [14], or take picture of user's surrounding [11]. They may even use the ambient audio to infer the keystroke [12], or use accelerometer to track the user location [9] or use gyroscope to identify pins/SSN [10]. They may use NFC in phone to scan periphery NFC device or RFID cards to get the sensitive information such as Credit card number and other information. A proof-of-concept Trojan Horse electronic pickpocket program under the cover of a tic-tac-toe game has already been developed by Identity Stronghold [15].

The current mobile operating systems, such as Android or iOS, check the permission and requires a user to grant the permission in out-of-context fashion such as during the installation via manifest [16, 17] or during the first use via system prompt [17, 18] as shown in Figure 1.1. The app gets permanent access to the requested resources once the user agrees to install the app or allow the app to access the resource. This approach requires users to be aware and diligent when they install the app. It is well-known that most users simply press "Yes" so as to proceed with the installation without heeding the warnings prompted to them [19, 20]. Eling et al. [21] show that 40.4% of the users in their study accepted unnecessary runtime permission requests for minimal reward. Moreover, most users ignore permissions, privacy policies, and terms of agreement altogether [22]. Applications are also reviewed to detect if there is a malicious code hidden underneath. Although the review process sets a bar for a malware developer, they can still circumvent this approach by exploiting the hardware/sensors which are considered harmless. The inertial sensors such as accelerometer and gyroscope do not require any explicit permission for access. The malware developers have used accelerometer [9, 12, 13] and gyroscope [10] to extract sensitive information compromising the privacy of the user. Also, the review process has

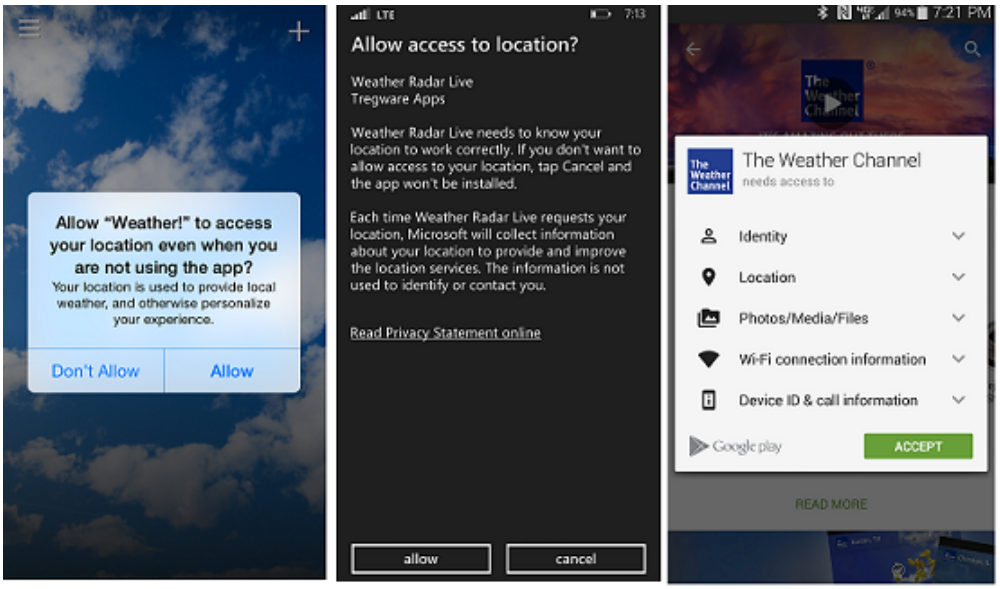


FIGURE 1.1: Apps in iPhone, Windows phone and Android phone requesting for permissions to access resources/services.

failed in the past and users with rooted/jail-broken phones can easily install the apps which may not have been reviewed [23, 24].

1.1.2 Outsider Attacks

These are the attacks that are generated from outside of the device. The devices need not be infected for such kinds of attacks to occur. A specific target application for outsider attacks is NFC. With the integration of NFC chip onto the new smartphones, users are now able to make payment via phones, or use their phones as an RFID cards. However, this chip stores sensitive information such as credit card number and other relevant information. Since the smartphone promiscuously provides its information to any reader that tries to access the information, it is vulnerable to a clandestine eavesdropping. For example, an adversary with an NFC reader can easily read credit card information stored on the victim's phone by just walking past the victim. This not only allows fraudulent and illegitimate purchases from the victim's account [8] but can also lead to owner tracking and privacy problem [25]. The stolen information may also be used to impersonate an NFC device via cloning [8, 25].

Another common outsider attack occurs when the mobile device is in physical control of a malicious entity, for example, upon theft of the device or under "lunch-time" access to the device. In this situation, the attacker may be able to fully or partially control the device just like its legitimate user. For example, the theft of PKES (Passive Keyless Entry and Start) keys will

simply allow the thief to unlock the car or the theft of the NFC payment enabled phone will allow the thief to perform the NFC payment transactions. The user needs to be transparently authenticated when using the mobile device as an authentication token since user convenience is one of the main goals of these systems.

Moreover, NFC devices also suffer from "ghost-and-leech" relay attacks [26–28]. Since a smartphone with NFC chip can act as a replacement for RFID tag which is used in a wireless authentication scheme, it suffers from similar threats since it responds promiscuously to any reader in its close proximity. In such mobile or wireless authentication system which moreover provides "Zero-Interaction-Authentication" ($\mathcal{ZIA}$) [29], two attackers collude to exploit such scheme. Here, an attacker (ghost) relays the challenge from the legitimate reader to a colluding entity (leech). The leech acts as a reader and provides the challenge to the victim's device. The leech relays the information from victim's device to the ghost which then provides the information to the legitimate reader. In such fashion, a ghost and a leech pair can impersonate as a victim device and falsely authenticate with the reader.

A similar attack is applicable to ZIA schemes that use a mobile device (e.g., a phone or a car key) to unlock another device (e.g., a computer or a car) just based on the co-presence of the two devices, i.e., without any explicit user interaction.

1.2 Context Detection

In this dissertation work, we set out to defend against the insider attacks caused by malware and the outsider attacks caused by unauthorized readings, relay attacks and physical control. Our observation is that the sensors that the attackers often use to exploit different vulnerabilities may also be used to protect the users. Our idea is to leverage the sensors to infer the "context" in which the device is used. For example, the context when there is a legitimate request to access a resource/service on the device differs from the context when there is a malicious request. The attacks mentioned above, in order to remain stealthy, occur in scenarios when the users do not intend to access the resources/services or in scenarios when two prover-verifier devices are far apart (e.g., in ZIA, the phone is at a restaurant, while the computer is at the office). If such context can be captured/identified in some ways, these attacks could be prevented.

As such, we propose to extract the context when there is legitimate request for the permission

to access a given device service/resource, or when there is a user intent to authorize a particular action. In other words, whenever the legitimate user wants to authorize the app to access some service or resource, or wants to authorize himself for the credit card transaction or for the access to the room, the system automatically detects the context and access will be allowed. On the other hand, if an attacker attempts to do so, the appropriate context may be missing and therefore access will be denied.

We can extract different forms of contexts to prevent users against insider and outsider attacks. We observe that whenever a user wants an app to access a resource or service, he either performs a gesture which can be either implicit or explicit. For example, when a user wants to make a phone call, he moves his phone towards the ear, or to take a picture, he moves his phone to orient his phone for a perfect snap. Similarly, when a user tries to authenticate himself or authorize a transaction, he moves his phone with his hand to the reader. The inertial sensors on the phone will report the corresponding sensor changes. When malware tries to make phone call or take a picture, the gesture will be missing. Also, when two colluding adversaries attempt to relay the information from a user device to the reader who are far apart, the ambient context around them will be different. They will be sensing different Wi-Fi, Bluetooth devices around them, listen different ambient audio and report different physical context such as temperature, humidity, gas content or pressure. Hence, we can leverage the context to provide sound access control and prevent malicious authentication and authorization.

To simplify our exposition, we define three broad categories of context that may be used to enhance mobile device security and privacy:

1. ***Explicit Context***

   When the user has to perform an explicit action to permit the app so it can access a given resource/service, such a context is called an "Explicit Context". Such actions or gestures are not transparent to the users. Explicit context might be a burden to the users as they have to put an extra effort while authorizing the app each time. However, it is better than using PINs/passwords which are often forgotten, and has significant security advantage over using nothing at all. We provide intuitive gestures such as waving a hand in front of the phone or tapping/rubbing the phone utilizing proximity and light sensors. These gestures are more secure than using a "Yes/No" dialog box as the most users are habituated

to pressing "Yes" when prompted. Moreover, a "Yes/No" dialog box will require user to explicitly press a button and interrupt the users even when the user did not intended any app to authorize. Our gesture-based approach has advantage over this as a user can just be prompted using a "Toast" mechanism or notified using a notification bar.

2. ***Implicit Context***

   If the users do not need to perform any extra movement or put additional effort to permit the app to access the corresponding resource/service, such context is called an "Implicit Context". Such actions or gestures are transparent to the users. The implicit context is better than the explicit gesture since the process is transparent to the users. As explained above, while making a phone call, we can use inertial sensors in the smartphone to identify if the user has intended to make the call and moved the phone along with his hand to the ear. As we will present later, we can use the sensors embedded in the smartphone as well as those on a wearable device, such as smartwatch/bracelet, to identify the gesture in a robust manner. The implicit context can be used to authorize the apps and to authenticate the users in the form of behavioral biometrics (e.g., unique user hand movements). We use different inertial sensors to extract such implicit context to provide robust authorization/authentication.

3. ***Environmental Context***

   The context which can be derived from sensing the ambient environment around the device is called an "Environmental Context". Such context can be used to detect the proximity of two devices or co-presence of the two devices. The users do not need to provide any gestures for authorization. The property of environmental context can be leveraged to authenticate users transparently during $\mathcal{Z}IA$. We use different ambient sensors embedded within the phone as well as those from the off-the-shelf ambient sensing devices such as Sensordrone[1] to sense the ambient information. We show that combinations of these sensors provide robust proximity detection and secure authentication mechanism for $\mathcal{Z}IA$ systems.

[1] http://www.sensorcon.com/sensordrone/

1.3 Contributions

Our work constitutes the design, implementation and evaluation of a variety of context detection mechanisms to enhance the security of mobile devices against various insider and outsider threats. The main contributions of our work are as follows:

1. We propose a novel approach to defend against malware on mobile devices based on intuitive gestures serving as the explicit context. We suggest different lightweight gestures that utilize different low-power sensors ubiquitously available on smartphones and tablets. Based on this context, we also provide selective unlocking for the sensitive resources such as NFC/RFID tag. We argue for a model which only approves the permission when the context is detected instead of promiscuously providing the information to any reader. We also use these contexts to selectively read the NFC/ RFID tag instead of executing the command promiscuously.

2. We propose the use of transparent gestures serving as the implicit context to defend against malware for three specific services, phone calling, camera snapping, and NFC tapping, which can be detected using inertial sensors embedded in the current smartphones.

3. We build an NFC tapping gesture detection biometrics to authenticate users transparently when they make NFC payment transactions at point-of-sale (POS) terminals. We extract multiple features from the phone's different sensors when a user taps her phone to NFC transaction terminal and implement the machine learning approach to identify if the sensor data corresponds to the owner of the device (or not). Moreover, we show that NFC tapping biometrics can be extracted with a high overall accuracy, while it does not seem possible for even a trained active attacker to succeed in mimicking the tapping gesture of a victim user.

4. We develop walking biometrics enhanced *WUZIA* (*"Walk-Unlock $\mathcal{Z}IA$"*), a multi-modal walking biometrics approach tailored to enhance the security of $\mathcal{Z}IA$ systems against stolen prover devices still with zero-interaction. Further, we demonstrate that *WUZIA* offers a high degree of detection accuracy, based on multi-sensor and multi-device fusion. We show that walking biometrics can be extracted with a high overall accuracy when using one of the devices (phone or watch), and be almost error-free when both devices are used

together. Furthermore, we analyze *WUZIA* against active attackers such as an imposter attack and a state-of-the-art active attack.

5. We propose the use of ambient sensors to detect the co-presence of devices based on environmental context so as to defend against relay attacks. We detect different forms of contexts using individual environmental sensors as well as a fusion of different sensors. We demonstrate that the fusion of multiple sensor modalities is effective, i.e., we can detect the context in a robust way, which helps to improve the security as well as usability.

6. We enhance the security model of contextual co-presence detection systems to consider an attacker who can manipulate the context (outsider attack). To this end, we demonstrate that it is possible to manipulate the readings of different ambient sensors (and combinations thereof) using low-cost, off-the-shelf equipment, representing a realistic attacker. Based on such manipulation capabilities, we comprehensively examine and quantify the advantage a multi-modality attacker, who can manipulate multiple sensor modalities simultaneously. For systems that use multiple modalities, we investigate two different sensor fusion approaches – *features-fusion* and *decisions-fusion* based on majority voting (equal voting and weighted voting), and show that both approaches are vulnerable to contextual attacks but the latter can be more resistant in some cases, at the cost of slight degradation in usability. We provide further guidelines to improve the security of co-presence detection systems in the face of such strong adversaries.

7. We present a novel attack against *Sound-Proof* [30], a notable zero-effort two-factor web authentication (2FA) scheme which uses ambient sounds as contextual information to detect the co-presence of the authentication (token) and the authentication terminal (browser). We introduce, design and develop the *Sound-Danger* attack system that exploits a wide variety of a smartphone's functionality to break *Sound-Proof*. To achieve this, we re-implement the *Sound-Proof*'s audio correlation algorithm and evaluate it against *Sound-Danger* under a large variety of attack settings. Then, as a representative example of how to deploy our attacks in practice, we collect general population statistics via an online survey to determine the phone usage habits and patterns relevant to our attacks. We then use these common statistics from our population sample to show how our different correlation-based attacks against Sound-Proof can be carefully executed to have maximum impact. We discuss the

implications of this attack against other ambient audio context based security systems, and outline mitigation strategies to resist the impact of such a strong attacker.

1.4 Related Publications

The work presented in this dissertation is based on the following publications and articles under submission, co-authored during my doctoral studies at UAB.

1. B Shrestha, M Shirvanian, P Shrestha, N Saxena. The Sounds of the Phones: Dangers of Zero-Effort Second Factor Login based on Ambient Audio. [31]

2. B Shrestha, M Mohamed, N Saxena. Walk-Unlock: Zero-Interaction Authentication Protected with Multi-Modal Gait Biometrics. *Under submission* [32]

3. B Shrestha, M Mohamed, S Tamrakar, N Saxena. Theft-Resilient Mobile Payments: Transparently Authenticating NFC Users with Tapping Gesture Biometrics. *Under submission*

4. B Shrestha, N Saxena, HTT Truong, N Asokan. Contextual Proximity Detection in the Face of Context-Manipulating Adversaries. *Under submission* [33]

5. M Mohamed, B Shrestha, N Saxena. SMASheD: Sniffing and Manipulating Android Sensor Data. In proceedings of the Sixth ACM Conference on Data and Application Security and Privacy (CODASPY), 2016[2] [34]

6. B Shrestha, D Ma, Y Zhu, H Li, N Saxena. Tap-Wave-Rub: Lightweight Human Interaction Approach to Curb Emerging Smartphone Malware. IEEE Transactions on Information Forensics and Security (TIFS), Vol: 10 (11), 2015 [35]

7. B Shrestha, M Mohamed, A Borg, N Saxena, S Tamrakar. Curbing Mobile Malware Based on User-Transparent Hand Movements. IEEE International Conference on Pervasive Computing and Communications (PerCom), 2015 [36]

8. HTT Truong, X Gao, B Shrestha, N Saxena, N Asokan, P Nurmi. Using Contextual Co-Presence to Strengthen Zero-Interaction Authentication: Design, Integration and Usability. Pervasive and Mobile Computing (PMC), Vol: 16, 2015 [37]

[2]This work is not part of this dissertation per se.

9. HTT Truong, X Gao, B Shrestha, N Saxena, N Asokan, P Nurmi. Comparing and Fusing Different Sensor Modalities for Relay Attack Resistance in Zero-Interaction Authentication. IEEE International Conference on Pervasive Computing and Communications (PerCom), 2014 [38]

10. B Shrestha, N Saxena, HTT Truong, N Asokan. Drone to the Rescue: Relay-Resilient Authentication Using Ambient Multi-Sensing. Financial Cryptography and Data Security (FC), 2014 [39]

11. B Shrestha, N Saxena, J Harrison. Wave-to-Access: Protecting Sensitive Mobile Device Services via a Hand Waving Gesture. Cryptology and Network Security (CANS), 2013 [32]

12. H Li, D Ma, N Saxena, B Shrestha, Y Zhu. Tap-Wave-Rub: Lightweight Malware Prevention for Smartphones Using Intuitive Human Gestures. Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks (WiSec) 2013 [40]

1.5 Organization

The rest of the dissertation is organized as follows.

In Chapter 2, we first introduce some preliminaries that will be discussed throughout the dissertation. Then, we provide threat models and design goals. In Chapter 3, we discuss the state-of-the-art work in the field of utilizing the context to provide security. We discuss the work that tries to address the problem of insider and outsider attacks. In Chapter 4, we introduce the design, implementation and evaluation of the explicit as well as implicit gesture detection to prevent the unauthorized access of resources/services by a malware. In Chapter 5, we introduce the design, implementation and evaluation of the implicit context, transparent to users, to authenticate users based on their behavioral biometrics (NFC tapping and gait based pattern). In Chapter 6, we present the design, implementation and evaluation of using context to enhance the detection of two co-present devices to prevent relay attacks. Chapter 7 introduces novel attacks on the system which utilizes the context. We critically evaluate the performance of the context based system in the face of context manipulating adversary. We provide an attack on *Sound-Proof*, cutting edge technology using audio to detect co-presence and provide second factor authentication. Finally, we conclude our work presented with the closing remarks in Chapter 8.

CHAPTER 2

BACKGROUND

In this chapter, we first present the preliminary concepts that will be discussed in the following chapters, and then we present our threat models and assumptions, followed by an overview of our contextual authentication and authorization approaches.

2.1 Preliminaries

2.1.1 Zero-Interaction Authentication ($\mathcal{ZIA}$)

A $\mathcal{ZIA}$ scheme involves a user $\mathcal{U}$ who intends to authenticate to a verifier terminal $\mathcal{V}$ (e.g., a PC, car or gate) using a prover device $\mathcal{P}$ (e.g., a phone or smart key). $\mathcal{U}$ does not explicitly take part in the authentication process other than by approaching $\mathcal{V}$ while carrying $\mathcal{P}$. $\mathcal{ZIA}$ is triggered by the devices sensing each other over a short-range wireless communication channel like Bluetooth. $\mathcal{V}$ will authenticate $\mathcal{U}$ by running a standard challenge−response based entity authentication protocol with $\mathcal{P}$ over the proximity communication channel. $\mathcal{P}$ and $\mathcal{V}$ pre-share a key K, which allows $\mathcal{P}$ to authenticate to $\mathcal{V}$ in the entity authentication protocol.

2.1.2 Relay Attacks

$\mathcal{ZIA}$ systems as well different payment systems are vulnerable to relay attacks. In relay attacks, an attacker (leech) relays the information from the $\mathcal{P}$ device such as phone or credit card of $\mathcal{U}$ to a colluding adversary (ghost). The attacker duo, a ghost and a leech, can succeed in impersonating as $\mathcal{P}$. Another scenario relates to payment tokens and point-of-sale readers. It involves a malicious reader and an unsuspecting payment token owner intending to make a transaction [41, 42]. In this scenario, the malicious reader, serving the role of a leech and colluding with the ghost, can fool the owner of the payment token $\mathcal{P}$ into approving to $\mathcal{V}$ a transaction which she did not intend to make (e.g., paying for a diamond purchase made by the adversary in a jewellery store while the owner only intends to pay for food at a restaurant). The main difference in the two scenarios relates to user awareness – in the first scenario, the user does not intend to authenticate at all,

whereas, in the second scenario, the user does intend to authenticate but ends up authorizing a different transaction than the one she intends to.

2.1.3 Biometrics Authentication

Users can be authenticated using different approaches such as "something you know", "something you have", and "something you are". Biometrics based authentication fall into the last one where users are authenticated using their intrinsic characteristics. In biometrics authentication, users can again be identified in two different ways, namely, 1. Physical biometrics authentication and 2. Behavioral biometrics authentication. In physical biometrics authentication, users are identified using their physical characteristics such as face recognition, iris recognition, fingerprint recognition, etc., while in behavioral biometrics authentication, they are authenticated based on the behavioral pattern such as the way they walk (gait based pattern), move (keystroke pattern, phone tapping pattern), talk (voice authentication).

2.1.4 Two Factor Authentication

Most of the systems currently use "something you know" approach where users remember password or pin along with their user name/id to authenticate to a system. However, such approach has been proven to be vulnerable as users tend to choose weak password to easily remember. To address this problem, multi-factor authentication has been deployed which provides an additional layer of security. As a second factor, commonly deployed approach is "something you have", where users have an additional device as a security token such as phone. This device provides the users with an additional information needed for the second factor authentication. The use of a general-purpose smartphone as a token [43–45], as opposed to a dedicated device [46, 47], helps improve usability and deployability of 2FA, and is currently a commonly used approach on the Internet.

2.1.5 *SMASheD*

The current Android sensor security model either allows only restrictive read access to sensitive sensors (e.g., an app can only read its own touch data) or requires special install-time permissions (e.g., to read microphone, camera or GPS). Moreover, Android does not allow write access to any of the sensors. Sensing-based security and non-security applications therefore crucially rely upon

the sanity of the Android sensor security model. *SMASheD* [34] (an abbreviation for Sniffing and Manipulating Android Sensor Data) is a framework under the current Android ecosystem that can be used to stealthily sniff as well as manipulate many of the Android's restricted sensors. *SMASheD* exploits the Android Debug Bridge (ADB) functionality and enables a malicious app with only the INTERNET permission to read, and write to, multiple different sensor data files at will. *SMASheD* can sniff and manipulate protected sensors on unrooted Android devices, without user awareness, without constant device-PC connection and without the need to infect the PC.

SMASheD framework comprises of three components: *SMASheD* server: a native service that provides the sensor data reading and injection capabilities, *SMASheD* scripts: two simple scripts used to copy the *SMASheD* server to the device and to start the server, and *SMASheD* app: an app that runs a status detection module in the background, and depending on the phone's status and the desired functionality, it sends requests to the *SMASheD* server to read or inject sensor events.

2.1.6 *Sound-Proof*

In the two-factor authentication described in Section 2.1.4, the need to look-up and interact with the phone, and copy the pin during a 2FA authentication session lowers the system's usability, which may prevent users from adopting this approach for authentication [30]. *Sound-Proof* leverages *ambient sounds* to detect the proximity between the phone and the login terminal (browser). Specifically, during the login session, the browser and the phone each record a short audio clip, and the login is deemed successful only if the two recorded audio samples are highly correlated with each other (and the correct password is supplied). Except of entering the password, *Sound-Proof* does not require any user action (e.g., transferring PIN codes or even looking-up the phone) – mere proximity of the phone with the terminal is sufficient to login. It may also work even if the phone is inside a purse or pocket.

The main security goal of *Sound-Proof* is to defeat a *remote attacker*, who has learned the user's password (e.g., by hacking into a password database server of the web service in question), and is attempting to login to the user's account, and possibly multiple user accounts. As argued in [30], given the prominence of remote attacks on the web today, this is a very legitimate goal. In order to login to the user's account, the remote attacker against *Sound-Proof* would have to predict the ambient sounds in the environment of the phone and possibly be in a very similar

environment as the user, which may be a difficult endeavor in practice, as shown in the security analysis reported in [30]. In other words, if the attacker can not predict the user's environment and is in a different environment than the user, the audio samples at the browser's end and the phone's end would not correlate, thereby preventing the attacker from logging in [30].

2.2 Threat Model and Assumptions

2.2.1 Insider Attacks

In the case of the insider attack, we assume that the device has already been infected by the malware without any user suspicion. The malware can be hidden in a normal application and can spread through various paths to the phone via various communication channels such as Bluetooth, Wi-Fi, and GSM. A user may download an app from an untrustworthy source that looks like a normal app such as game but contains malicious code. Preventing malware from being installed on the phone is beyond the scope of our model. The attackers use the malware to access sensitive services (such as phone call, SMS, or camera) for various malicious intentions. The malware can further access services such as NFC to scan confidential information from NFC/RFID cards/devices around the device's periphery.

We also assume that the malware has not infected the OS kernel of the device. Strengthening the kernel is again an orthogonal problem [48, 49]. To be specific, the kernel is immune the malware and the malware cannot alter the kernel control flow. We also assume that hardware is immune from the malware and the malware cannot manipulate the device's on-board sensors. The malware capable of manipulating the sensors can produce an appropriate context corresponding to the activity.

The ultimate goal of the malware is either to steal confidential information, compromise the privacy of the user or directly earn money by trying to access various sensitive resources/service such making premium phone calls, taking pictures of user surrounding [11], or read nearby NFC enabled credit cards/tags [15]. The malware may try to access such services frequently or rarely to remain stealthy. We do not impose any restriction as to how frequently the malware attempts to access such services.

We assume the attacker may be physically near the user. The attacker is unable to persuade the user to perform the gesture to access a particular service. However, he may coerce/fool

the user into moving a particular manner with a hope that such movement can generate similar motion as a valid gesture. In case of insider attack, we also assume that the device is not with the attacker. If the attacker has physical access to the phone, then he can simply grant permission to malware by providing the necessary gesture. In other words, our mechanisms, so far, are not meant for user authentication and do not provide protection in the face of loss or theft of phone.

2.2.2 Outsider Attacks

In the case of the outsider attack, the devices themselves are not infected. However, the attacker is trying to exploit the vulnerability of the NFC/RFID cards/tags which allows any reader to read this sensitive information stored in the device promiscuously. The Zero Interaction Authentication ($\mathcal{ZIA}$) systems and the NFC payment systems are prone to such attacks. Attacker can exploit the $\mathcal{ZIA}$ or the NFC payment systems in two different ways; either by stealing/possessing the device or by relaying the signals from the device.

2.2.2.1 Device Theft

As mentioned in Section 2.1, $\mathcal{ZIA}$ systems rely upon the authentication factor "something you have". In $\mathcal{ZIA}$ system, $\mathcal{U}$ does not need perform any explicit action or gesture for the authentication, and simply walking towards $\mathcal{V}$, while carrying $\mathcal{P}$ is enough to establish authentication.

In $\mathcal{ZIA}$ threat model, $\mathcal{P}$ and $\mathcal{V}$ are assumed to be honest (i.e., uncompromised and non-malicious). The communication channel between $\mathcal{P}$ and $\mathcal{V}$ is protected with encryption and authentication tools.

In a realistic threat model, an attacker should be assumed to be in possession of the $\mathcal{P}$ device. The attacker may obtain the $\mathcal{P}$ device either by stealing it or via a lunchtime attack [50, 51]. In this model, existing $\mathcal{ZIA}$ systems are completely broken since the attacker can just access $\mathcal{V}$ by making use of $\mathcal{P}$. The attacker can simply go near the $\mathcal{V}$ device and unlock it in $\mathcal{ZIA}$ system or go to NFC POS terminal and perform the transaction.

We further assume that the phone's OS kernel is healthy and the attacker is unable to alter the kernel control flow. Strengthening the kernel is an orthogonal problem [48, 49]. We also assume that attacker cannot manipulate device's onboard sensor hardware. In other words, the attacker only has physical access to the device but does not have internal control of the device.

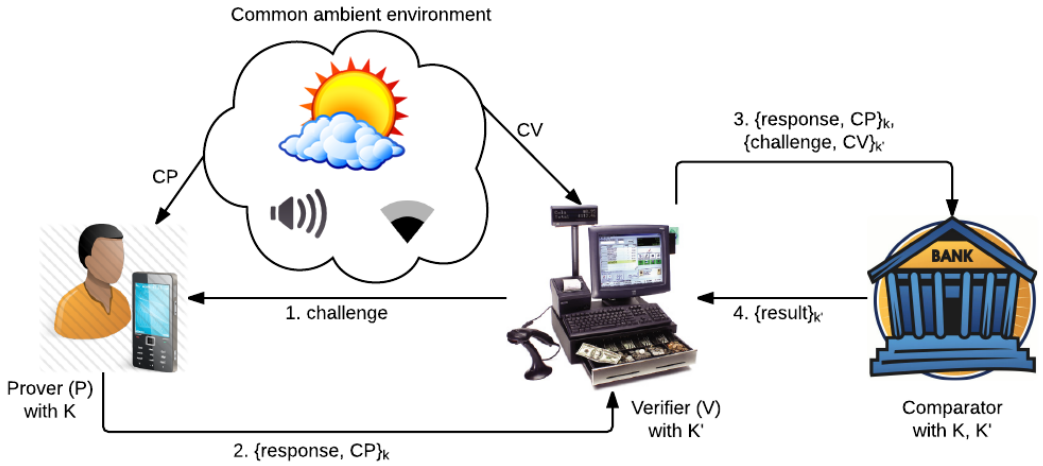


FIGURE 2.1: Proximity based authentication

2.2.2.2 Relay Attack

The attacker can relay the signals from $\mathcal{P}$ to $\mathcal{V}$ without stealing the $\mathcal{P}$ device. In such case, $\mathcal{U}$ would be authenticated as having the $\mathcal{P}$ device even though $\mathcal{P}$ is not in the proximity of $\mathcal{V}$. Such vulnerability of the $\mathcal{ZIA}$ system calls for the proximity detection system. The Figure 2.1 shows a general model of our proximity-based authentication. In this model, $\mathcal{P}$ wants to authenticate itself to $\mathcal{V}$ and convince $\mathcal{V}$ that it is close to $\mathcal{P}$. Typically, when $\mathcal{P}$ is close to $\mathcal{V}$, the authentication process between $\mathcal{P}$ and $\mathcal{V}$ is run. $\mathcal{V}$ makes use of a back-end "comparator" function to make the authentication decision (it could reside on $\mathcal{V}$ device or on a remote machine such as a bank server in the case of payment transactions). $\mathcal{P}$ and $\mathcal{V}$ have pre-shared secret keys K and K', respectively, with the comparator. When $\mathcal{P}$ wants to authenticate to $\mathcal{V}$, $\mathcal{V}$ sends a *challenge* to $\mathcal{P}$. The $\mathcal{P}$ computes a *response* based on the *challenge* and K. $\mathcal{P}$ returns the *response* to $\mathcal{V}$ which then uses the comparator function to decide if the *response* is acceptable.

This model is applicable to various real-world scenarios such as payment at a point-of-sale (POS) terminal and $\mathcal{ZIA}$ systems. In the payment scenario, the payment card plays the role of $\mathcal{P}$, and the POS terminal plays the role of $\mathcal{V}$. The issuer of the payment card plays the role of the comparator. In a $\mathcal{ZIA}$ systems, the user token (key or mobile phone) acts as $\mathcal{P}$ and the terminal (car or desktop computer) plays the role of $\mathcal{V}$. The comparator functionality is integrated in the terminal itself and therefore K' is not needed.

In this model, we assume that the devices are not infected and will follow the standard protocol for communication upon the request for the authentication. We assume a standard

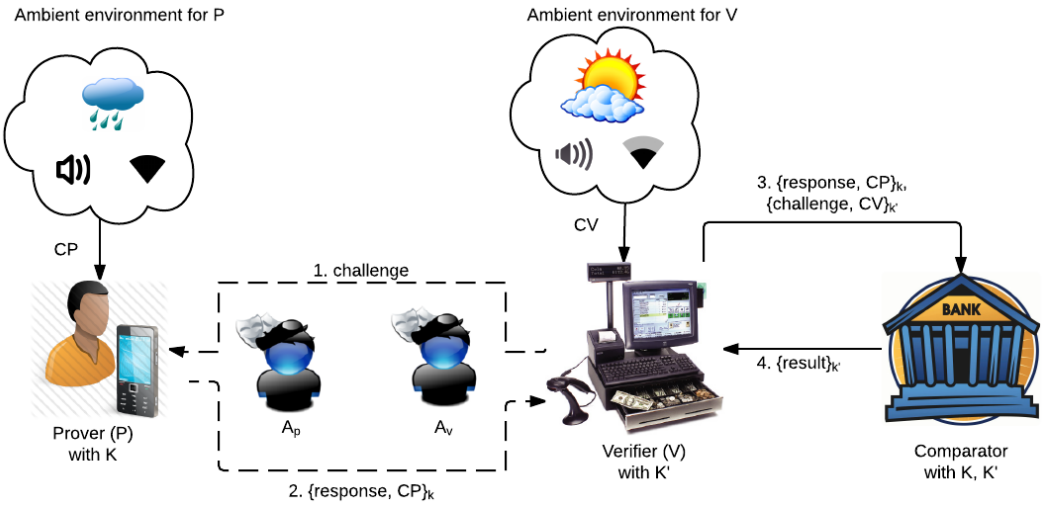


FIGURE 2.2: Relay attack in proximity based authentication

Dolev-Yao adversary [52] who has a complete control over the communication channel over which the authentication protocol between $\mathcal{P}$ and $\mathcal{V}$ is run. The adversary $\mathcal{A}$ does not physically possess any of the legitimate devices. The goal of $\mathcal{A}$ is to carry out the relay attack such that $\mathcal{V}$ is convinced that $\mathcal{P}$ is in close proximity when in fact $\mathcal{P}$ is far. $\mathcal{A}$ can take the form of a "ghost-and-leech" [26] duo ($\mathcal{A}_p$, $\mathcal{A}_v$) such that $\mathcal{A}_p$ is physically close to $\mathcal{P}$ and $\mathcal{A}_v$ is physically close to $\mathcal{V}$, and $\mathcal{A}_p$ and $\mathcal{A}_v$ communicate over a high-bandwidth connection as shown in the Figure 2.2.

2.3 Design Goals

For each of our defense mechanism to be useful in practice, it must satisfy the following properties:

- ***Lightweightness:*** The scheme should be lightweight in terms of memory, computation, and power consumption.
- ***Efficiency:*** The users should not have to wait for a noticeably long time while using the scheme. The longer delays caused due to system will affect the usability of the system.
- ***Robustness:*** The approach should be tolerant to errors. Both the False Negative Rate (FNR) and the False Positive Rate (FPR) should be quite low. A low FNR means that the system grants the permission to app during benign case with high probability. The low FNR infers high usability. On the other hand, a low FPR means that the system denies

access to requesting apps/devices during malicious case with high probability. The low FPR infers high security.

- ***Usage Model Consistency:*** The approach should require a little or no change, to the usage model of existing application. Although the context that is transparent to the users are preferred, there can be cases when accessing particular services/resources may not have definite context that can be extracted. For such services, explicit context needs to be used. Such context should still be intuitive and easy for users to perform. Nevertheless, our system should require minimal effort from users if not transparent.

CHAPTER 3

LITERATURE REVIEW

In this chapter, we review state-of-the-art research that use context to enhance the security of the system thwarting insider and/or outsider attacks. Basically, we explore the work that tries to address the insider and outsider attacks in different ways.

3.1 Malware Detection and Prevention

The most common approach to defend against the malware are static analysis [53–56] and dynamic analysis [57–60]. Static analysis, also known as signature-based detection, is based on source or binary code inspection to find suspicious patterns (malware) inside the code. This approach has been used as current solutions by many anti-virus apps. However, it can be evaded by malware authors through simple obfuscation, polymorphism and packing techniques. Also it cannot detect zero day attacks. Dynamic analysis, also known as behavior-based detection, monitors and compares the running behavior of an application (e.g., system calls, file accesses, API calls) against malicious and/or normal behavior profiles through the use of machine learning techniques. It is more resilient to polymorphic worms and code obfuscation and has the potential to defeat zero-day worms. Permission models have become very common on smartphone operating systems to provide access control to sensitive services for installed third party application. The previous research focuses on optimizing desktop solutions to fit on mobile devices. These techniques for desktop computers are still considered too time consuming for resource-constrained mobile devices operated on battery. Venugopal et al. [61] try to speed up the signature lookup process in static analysis by using hashes. Several collaborative analysis techniques have been proposed to distribute the work of analysis by a network of devices [62, 63]. Burguera et al. [64] and Cheng et al. [65] have proposed remote server assisted analysis techniques to reduce the overhead of the computation on the individual devices.

Chaugale et al. [66] have proposed to use the hardware interruption generated by human initiated actions to differentiate pure software initiated actions which should not generate hardware interruption. They aim at detecting the malware specifically targeting SMS and audio services.

Whenever users press or touch the keypad or touchscreen to type and send the SMS, it generates the hardware interruptions for each key press event while when the malware tries to send the SMS, it will not explicitly generate such interruption, hence, providing some context to the system. Our approach differs from their approach in a way that they check if there is (any) user-activity involved when the request is being made while we check if there is a special user-activity. So our approach has advantage as our approach provides more fine-grained access control to sensitive services and thus can detect even sophisticated malware. For example, a sophisticated malware which is hidden underneath a tic-tac-toe game, which already involves touch screen activity generating hardware interruption, can easily send SMS while our approach will look for corresponding context to provide SMS permission.

Roesner et al. [67] have proposed user-driven access control to grant the permission to the app which requests for the access to the resource only when user's permission granting intent is captured. They encourage the operating system to provide Access Control Gadgets (ACGs). ACGs are UI elements exposed by each user-owned resource for applications to embed. The user's UI interaction with the corresponding ACG grants the permission to the application to access the corresponding resource. Their design differs with ours as they grant permission to the app when user's authentic UI interaction with corresponding ACG is captured while our design grants permission to the app when a specific user gesture is captured. Their work requires kernel level changes and requires Resource Monitor (RM) to be incorporated for each resource such as device drivers. Moreover, if an application requires different resources to be accessed, they suggest using composition ACG (C-ACG) with composition RM (C-RM). If there are many resources that need to be accessed by an application, then the number of C-ACG and C-RM will be extremely large. Also, for services like NFC which may not have any specific UI elements or ACGs associated with them, a new ACG need to be designed which hampers the usability of such services.

Shebaro et al. [55] have presented context-based access control ($cbac$) for mobile devices. In this system, they restrict the apps from using the resources/ services according to the context based policies. The system tries to identify the context based on location such as if a user is in a meeting room or in a rest room. When the user is in such places, microphone recording or camera recording must be prohibited to prevent the user against the spyware. The $cbac$ system identifies where the user is based on GPS trilateration, cellular triangulation and Wi-Fi Positioning method.

It also uses time to enforce the context based policy. Their system differs from ours with respect to that our system checks for gestures to provide access while they use location and time to get the context and provide access. Our system has advantage over theirs as they need to frequently poll the $LocationManager$ service to observe if the user has moved to a sensitive place while our system is triggered only when an app makes request for sensitive resources/services. Also in $cbac$, the user needs to explicitly set policy in the beginning. This can result in a human error, i.e., he may either add more restriction or less restriction to the resources for particular context. Also, asking user to set policy every time a new location is detected can be burdensome for users.

3.2 Biometric Authentication

Conti et al. [68] authenticate users by analyzing the hand movements while making/answering phone calls. They investigated if such motion can be used as biometric authentication measure. They have used Dynamic Time Warping (DTW) algorithm to analyze and detect the gesture of making/answering phone calls. They have only considered accelerometer and orientation sensors. The experiment was done in a controlled setting and no real-world scenario has been captured.

Hong et al. [69] propose Waving Authentication (WA), a biometric authentication based on waving hand along with the phone. WA utilizes accelerometer sensor to extract 8 features, train SVM classifiers to build a model and authenticate users with this model. However, this approach is not transparent to the user.

Gascon et al. [70] have analyzed typing motion behaviour of the user to continuously authenticate a user on smartphones. It records the touch input along with the timestamps when the keys are pressed or released. They use different sensors such as accelerometer, gyroscope, and orientation sensors and extract 2376 dimensional vector representing the typing motion behaviour of the user. They use linear Support Vector Machine (SVM) classifier to identify if the typing motion belongs to user or not. Other work [71–74] share the similar philosophy to authenticate users based on the touch gesture. They either use only the touch sensor or use the touch sensor in conjunction with different inertial sensors.

Many researches have explored the use of accelerometer to authenticate the users based on their walking pattern. These work mostly use electronic motion recording (MR) devices such as MR100 wearable sensor [75], ZSTAR [75, 76], ADXL202JQ accelerometers [77],

MMA7260 [78], etc. These work analyze the accelerometer reading by attaching such MR sensors at different location of the body such as waist [75, 77–79] (device wore in a belt), lower leg [80, 81], shoe [82–85], pockets (chest/hip) [86, 87], upper limb/forearm [86], gloves [88–90], and so on. In most of these work, the MR device was tied on the particular parts of the body as most of these devices were not wearable.

Vildjiounaite et al. [87] used accelerometer module (MR sensor) and placed it in chest pocket, hip pocket and hand to authenticate users based on their walking pattern. To perform their experiment, they made mock-ups of "clothes with pockets" from pieces of textile which the users put on over their normal clothes. They reported that since it was not the real pocket, shifting of the mock-ups of clothes affected the accelerometer readings while the accelerometer module itself was not shifting as it was attached to the mock-ups of clothes.

Gafurov et al. [86, 91] used a "Motion Recording Sensors" (MRS) to collect accelerometer data. In their work [91], they tried to spoof the user's walking pattern by performing the experiment in two rounds. First, the targeted user walked in front of the attacker twice. Then, the attacker walked alone twice mimicking the user. They showed that such minimal effort impersonation attack on gait pattern does not increase the chances of imposters being accepted significantly. Further they used MRS attached to the belt.

Stang et al. [76] also explored the gait based authentication approach using ZSTAR accelerometer sensor and analyzed if the imposters could imitate the walking pattern. They recruited 13 participants to imitate users. Each participant was given 15 attempts on each template to attack. The imposters did not see the original walking but they were given a simple description of the gait. The participants were provided with the visual feedback such that they could see the template gait graph and their gait graph continuously plotted on a big screen. The walk duration was 5 second long for each walk sample. After each attempt a match score between 0 and 100 was displayed based on correlation such that 100 is a perfect match. They reported 3 persons exceeded the correlation threshold once, 2 persons exceeded the threshold twice, 1 person exceeded it three times and 1 person managed to exceed as much as 9 times in 15 attempts. Therefore, they concluded that it is easy to walk like another person.

Another attempt to mimic walking pattern was made by Mjaaland et al. [75]. They trained seven imposters to imitate a specific victim. They used two wearable sensors: the Motion Recording 100 (MR100), and the Freescale ZSTAR sensor to record the accelerometer sensor

values. They attached these sensors on belt and asked the participants to wear the belt which could be mounted to any person's hip regardless of what they were wearing such that the device would always have the same-orientation. They conducted short-term hostile scenario and long-term hostile scenario. In the former scenario, they trained six participants for two weeks, five hours every day while in the latter scenario, they trained the seventh participants for six weeks. In both scenarios, the imposters were not able to imitate the victim's walking pattern. They concluded that there is a physiologically predetermined boundary to every individual's mimicking performance and also that if one successfully adopted gait characteristics improved an attacker's performance, other characteristics worsen in a chain-like effect.

One of the works in line with ours is by Kumar et al. [92] as they also used an Android smartphone with an app to record sensor data. In this work, they only used features extracted from accelerometer sensors. They recorded accelerometer sensor only to authenticate users based on their walking pattern. In this work rather than imitating a victim walking pattern, they used different gait characteristics to match the victim's walking pattern. They assumed that the attacker has the victim's gait samples. This means that the attacker knows the values for each feature which would satisfy the classifier threshold to authenticate. From the 47 features extracted from accelerometer sensor only, they ranked their features based on information gain based attribute evaluator [93] and selected 17 top ranked features only. Since they were using features extracted from accelerometer sensor only, the features might be highly correlated and reported that that their system's FAR increased from 5.8% to 43.66%.

Researchers have also explored accelerometer and/or gyroscope sensors available on current smartwatches for the purpose of gait detection. Johnston et al. [94] used the accelerometer sensor embedded in the smartwatch, while Kumar et al. [95] used the accelerometer and the gyroscope sensor. They only used the sensors from smartwatch and did not consider the use of multiple devices (both phone and watch). The authors extracted a total of 76 features (32 features from the accelerometer readings and 44 features from the gyroscope readings).

3.3 Relay Attack Resilience

The most common approach to avoid relay attack is "Distance bounding techniques" [27, 96, 97]. However, because of its difficulty to deploy on commodity devices [98] and its dependence

on low-level implementation which is vulnerable to attackers [99, 100], it may not be realistic for commodity devices. An alternative approach to defend relay attacks is to use the ambient environment which is being investigated recently. This is based on the assumption that two legitimate devices performing transaction under benign scenario will be co-located and will detect the similar ambient environment at that location whereas when they are not co-located the ambient environment detected by them will be significantly different. Prior works rely on commodity devices which are equipped with various traditional sensors such as Wi-Fi, Bluetooth, and audio microphones.

Halevi et al. [98] have used ambient audio and light to detect if the two devices are co-located. They present the analysis using different methods such as time-based, frequency-based and time-frequency based similarity detection using raw audio data. They show that the ambient audio has better accuracy in determining the co-presence than the ambient light. A pattern based audio alignment was used by Nguyen et al. [101, 102] to detect and compare ambient audio to provide secure communication between mobile phones. Schurmann et al. [103] also used audio to detect the ambient context and provide secure communication.

Krumm et al. [104] have proposed "NearMe" which uses Wi-Fi similarity features for proximity detection. They build a model using data collected in an office building environment and tested in a cafeteria environment. Varshavsky et al. [105] have proposed the use of the common radio environment (Wi-Fi) as a basis to deriving shared secret between co-located devices. They introduce an algorithm Amigo that extends the Diffie-Hellman key exchange with verification of co-present devices. Another proximity detection approach by Narayanan et al. [106] also utilizes ambient Wi-Fi information. They studied the use of various modalities for private proximity detection and concluded that on Wi-Fi broadcast packets and Wi-Fi access point IDs are likely to perform best. Our systematic experiments confirm that Wi-Fi access point IDs perform well.

Czeskis et al. [107] proposed "secret handshakes" to avoid ghost-and-leech attack by limiting the context where the contactless card communicates with the reader. They used only accelerometer data as contextual information.

3.4 Two Factor Authentication

Most common and traditional form of Two Factor Authentication "2FA" employs hardware tokens such as RSA SecurID [46] and Yubico [47]. These hardware tokens are specialized devices used solely for the purpose of authentication. Such schemes require users to carry and interact with the token. These schemes may be expensive to deploy because the service provider must provide one such token per customer.

Many software tokens 2FA schemes are also available, including Google 2-Step Verification [43], Duo Push [44], and Celestix's HOTPin [45]. These schemes are both scalable and flexible as single personal device can be used with multiple services in such schemes. These schemes are also cost effective, since deploying software tokens are logistically much simpler. These schemes prompt the user with a push message on his phone with current login attempt information and the user interacts with his phone to authorize the login.

PhoneAuth [108] is a software token 2FA scheme that leverages Bluetooth communication between the browser and the phone, to eliminate user-phone interaction. The Bluetooth channel enables the server (through the browser) and the phone to run a challenge-response protocol which provides second authentication factor. This scheme requires browser to have Bluetooth communication capability which is currently not available on many browsers. Authy [109] is another approach that allows seamless 2FA using Bluetooth communication between the computer and the phone. However, Authy requires extra software to be installed on the computer.

Traditionally, these 2FA schemes increase resistance to online dictionary attacks. Shirvanian et al. [110] presented several 2FA schemes that are enhanced to strengthen security against both online and offline attacks. The main idea underlying all their 2FA protocols is for the server to store a randomized hash of the password, $h = H(p,\ s)$, and for the device to store the corresponding random secret s. The authentication protocol checks whether the user types the correct password p and also that it can access the device that stores s.

SlickLogin [111] (recently acquired by Google) minimizes the user phone interaction. It employs near-ultrasounds to transfer the verification code. The notion is to generate unique near-ultrasounds for each login attempt and use the very non-audible audio to authorize the attempt. To verify user's identity, a website plays a uniquely generated, nearly-silent sound through the computer's speakers. An app running on the user's phone picks up the sound, analyzes it, and

sends the signal back to the site's server. The server verifies the user with the possession of the phone as a second factor.

CHAPTER 4

CONTEXT ENHANCED AUTHORIZATION

4.1 Introduction

In this chapter, we show how the context (implicit and explicit) can be used to enhance the safety and security of the system to authorize an app requesting for sensitive permission. Especially, we show how we can use explicit and implicit context to authorize benign apps in smartphones requesting for permissions while block malwares requesting for such sensitive permissions. With this approach we focus on preventing insider attacks in the form of malwares.

To detect implicit and explicit contexts, we use the gestures provided by the users. The user gesture can be detected using various inertial and environment sensors embedded in the smartphone. As mentioned in previous sections, the users can either be asked to provide the gesture (explicit gesture), or the gesture can be extracted from the user movement when he is performing corresponding activity (implicit gesture). We carried out all of our experiments for gesture detection in Android smartphones. The Android OS, one of the most popular smartphone operating systems, provides APIs to support different categories of sensors such as those that measure motion, position and environment. It also provides support for NFC transactions. Our system architecture can be visualized in Figure 4.1.

4.2 Explicit Gestures

We detected hand waving/tapping/rubbing in front of phone using low powered sensors such as proximity sensor [35, 40], and light & accelerometer sensors [112].

4.2.1 Using Proximity Sensor

In our work [35, 40], we used a proximity sensor, which works by sending an electromagnetic signal and analyzing the change in the electromagnetic field or the returned signal itself. This is used in the smartphones primarily to detect if there is an object in the close proximity to the phone so that unintentional button press/touch events can be prevented in event such as when

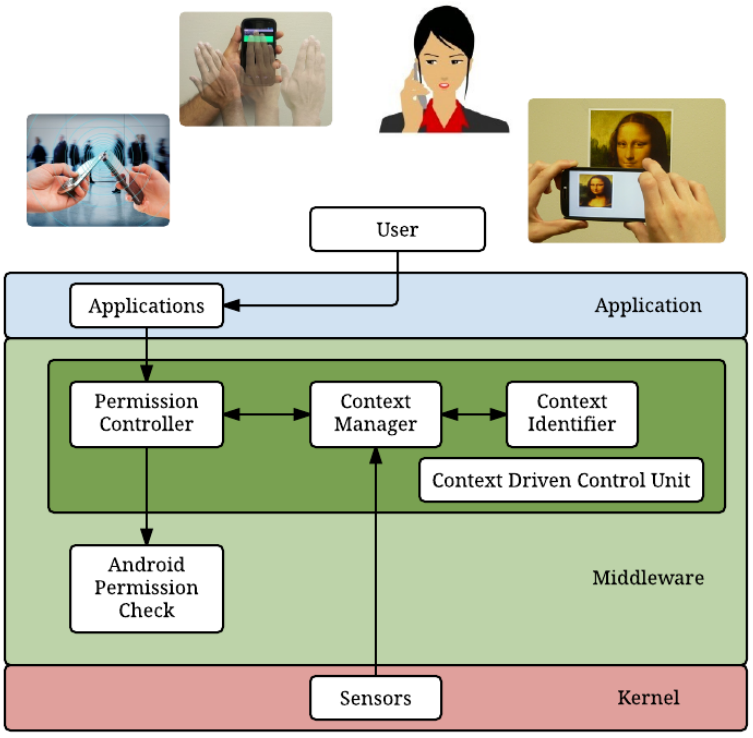


FIGURE 4.1: Context Driven System Architecture for Mobile Malware Prevention

user is making phone call. For this reason, the proximity sensor is primarily located near the ear piece and the phone's display facing towards the user. We leverage this property to develop our gesture detection mechanism.

The algorithm to detect these gestures using proximity sensor is simple and straightforward as illustrated in Algorithm 1. Also, the detection approach does not require any template to be stored which makes our approach lightweight, satisfying one of our design goals. When an app tries to access sensitive services/resources, we will check for the proximity sensor values for certain duration. We record the proximity sensors values along with their corresponding timestamps. If we find more than certain number of fluctuations in proximity value within certain duration, we deem the fluctuations as hand waving/tapping/rubbing gesture and the app will get access to the requested services/resources.

To evaluate our approach, we conducted an experiment with 16 volunteers from our department. Eight of the participants were requested to perform the waving based unlocking 10 times and the result was recorded automatically by the app. The average gesture recognition rate observed was 93.75%. While the other eight participants were requested to perform the tap or the rub gesture near proximity sensor. The average gesture recognition rate in this case was 96.25%.

Algorithm 1 Hand Tap-Wave-Rub Detection using Proximity Sensor

1: Set $proxIndex = 0$, $WIND\_SZ = 6$, $WAVE\_TIME\_LIMIT = 1500\ ms$, and $UNLOCK\_TIME\_FRAME = 1000\ ms$
2: Record the time whenever proximity sensor detects a change
    1. An array $ProxChangeTime$ with size equal to $WIND\_SZ$ was used to record the time of proximity sensor change in cyclic order.
3: Calculate the time difference between the current time with the time recorded in previous six sensor change value.
    1. $TimeDiff = ProxChangeTime[proxIndex] - ProxChangeTime[(proxIndex + 1)\%WIND\_SZ]$
4: IF TimeDiff is less than $WAVE\_TIME\_LIMIT$, Unlock for $UNLOCK\_TIME\_FRAME$
5: Increase $ProxIndex$ by 1, i.e., $proxIndex = (proxIndex + 1)\%WIND\_SZ$
6: Repeat Step 2.

4.2.2 Using Light & Accelerometer Sensors

We used light sensor to detect the hand wave gesture along with accelerometer [112]. The light sensor values are dependent on the brightness of the surrounding. The primary purpose of the light sensor is to sense the light of the environment and adjust the display accordingly to save the battery in the smart devices. This is the reason, the light sensor is placed near the display of the phone. The proximity sensor may not be available in all the devices such as tablets since the phone calling is not the primary purpose for such devices. However, to save the battery, the light sensor is ubiquitously available in most, if not all, smart devices. The algorithm to detect quick fluctuations in the reading of the light sensor is also very simple and straightforward as shown in Algorithm 2. Hence, unlike many other gesture recognition algorithms, pre-established templates are not needed. This makes our approach extremely lightweight, satisfying one of our design goals.

Whenever the access to the sensitive services/resources is requested by an app, we record light sensor values along with their timestamps. We then analyze the fluctuation in the light intensity. If the light value fluctuates beyond a given threshold for certain number of times within an allocated time, then we consider such fluctuations in light values as being triggered by the hand wave gesture. We calculated the threshold, used to determine if the light has fluctuated, according to the ambient light intensity. For example, when it is dark with the ambient light intensity around 200 lux, then optimal threshold to determine the fluctuation is 20 lux while when it is bright with the ambient light intensity around 60,000 lux, the optimal threshold is around 15,000 lux. We used eight different thresholds for eight different ranges of light intensity

Algorithm 2 Hand Wave Detection using Light Sensor (and Accelerometer)

1: IF sensors are locked THEN wait for $MOVEMENT\_LOCK\_TIME$
ELSE get accelerometer sensor readings x, y and z.
2: IF
$$\sqrt{x^2 * y^2 * z^2} > ACC\_THRESHOLD$$
THEN lock the sensors for $MOVEMENT\_LOCK\_TIME$ and RETURN to step 1.
3: IF sensors are not locked THEN get light sensors reading to check if wave gesture is detected.

1. Analyze $WINDOW\_SIZE\_FOR\_LIGHT$ data to find out how many extremas (maximas and minimas) were there using $LIGHT\_THRESHOLD$.
2. IF $extremaCount > CHANGE\_COUNT\_FOR\_LIGHT$ AND All the light data are recorded within $WAVE\_TIME\_LIMIT\_FOR\_LIGHT$ THEN
SET $unlockAttempted = true$,
RECORD first unlock attempted time
DISPLAY Message "Stop Waving" for $WAVE\_TIME\_LIMIT\_FOR\_LIGHT$.
3. IF $unlockAttempted$ THEN
   (a) IF another unlockAttempt is obtained within less than $WAVE\_TIME\_LIMIT\_FOR\_LIGHT$ THEN Do not unlock, reset everything and start over, i.e., return to Step 2.
   (b) IF another $unlockattempt$ is not obtained within $WAVE\_TIME\_LIMIT\_FOR\_LIGHT$ THEN $Unlock$ the phone for $UNLOCK\_TIME\_FRAME$.

to accurately determine the wave gesture. The fluctuation of light detected using above approach can be deemed as hand waving or it could have been triggered due to some environmental effects. So instead of permitting the app the access, we delay the unlock for certain time. If no fluctuation is detected in light then we can safely assume that it was human who performed the wave gesture.

We used the light sensor in conjunction with accelerometer sensor. The accelerometer sensor is used to reduce the false positive while classifying the light fluctuations as the wave gesture. The movement of the phone triggers the corresponding change in the position of the phone with respect to the light source. This in turn will be detected as the wave gesture. In order to reduce this effect, if the phone detects movement, greater than certain threshold, as per the accelerometer data, we do not consider this as the wave gesture. Further, it locks the light sensor data.

We conducted an experiment with 20 volunteers to determine the accuracy of our gesture detection algorithm. Each of them was requested to perform the hand wave based unlocking for ten times and the results were recorded automatically by our app. The average recognition rate was 90.5%. Most of the undetected wave gesture occurred when it was dark. When the light intensity was greater than 700 lux, the system recognized the wave gesture with 95.71% accuracy, when the light intensity was between 350 lux and 700 lux, the accuracy was 87%, whereas when the light intensity was less than 350 lux, the accuracy was 83.3%.

4.3 Implicit Gestures

In this section, we set out to defend against mobile malware that can exploit critical and sensitive mobile device services, especially focusing on the *phone's calling service, camera and NFC*. In order to remain stealthy, mobile malware attacks occur in scenarios where the device user has no intention to access the underlying services. Thus, if the user's intent to access the services can be captured in some way, these attacks could be prevented. We propose to elicit a user's intent via gestures that are transparently and naturally performed by the user prior to accessing the services. In other words, whenever the user wants to access the service, she will naturally exhibit a particular gesture. On the other hand, if the malware attempts to access the service, the gesture will be missing and the access request can be blocked. We focus on *authorizing an app* with the use of transparent human gestures, *not on authenticating users* (which is an independent problem).

4.3.1 Our Approach: Call-Snap-Tap

We can extract a gesture from the natural interaction of a user with the device when user wants an app to perform some operation that needs access to some sensitive services/resources. For example, when making a phone call, a user wants an app to make a phone call while he moves his hand along with the phone towards the ear.

We focused on three different kinds of services, namely calling, snapping and tapping, which involves such human interaction with the device [36]. The user needs to move his phone in certain ways to perform these activities. For example, when the user wants to make call, he presses the "dial" button and moves his phone to ear. Similarly when he wants to snap a picture, he opens the camera app, moves his phone to compose a picture, and clicks the snap button.

The system model for our approach is shown in Figure 4.1. Our approach adds another layer of permission control on top of the original Android permission granting system.

4.3.2 App Design

To develop and evaluate our gesture-centric malware defense mechanism, we first needed to collect data from users to recognize the various gestures exhibited by them in varying scenarios. To this end, we created a suite of four separate apps, each of which collects the data from users

while: (1) making/answering phone calls (the *Call App*), (2) taking pictures using the phone's camera (the *Snap App*), (3) reading NFC tags via tapping (the *Tap App*), and (4) various random movements and activities captured at random times or in controlled settings (the *Snoop and Control App*). The app supports Android OS 4.0 (API Level 14) or later. The data collected from this app system is used to recognize the Call, Snap and Tap gestures, and to determine the possibility of these gestures matching with one another and with other controlled activities.

To prevent the overuse of sensors and excessive drainage of the battery, the periods of recording data must be limited to the events identified as the capture periods. Android provides/throws an intent to all listening applications whenever the user intends to perform certain activities. Hence, we configured the app to trigger the sensor readings whenever a corresponding event occurs. Once the event is received, the data from sensors are recorded as shown in Figure 4.2. The specifics of our apps are discussed below.

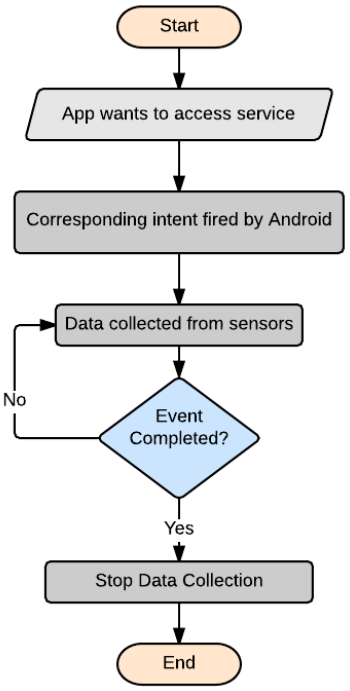


FIGURE 4.2: Data Collector Flowchart

4.3.2.1 Call App:

The phone calling intent is triggered when the call is initiated/answered and the state becomes "OFFHOOK". With this intent, the app starts recording the sensor data. This means that the user has made/answered the phone and the associated Call gesture, i.e., the motion to bring the phone to the ear, has been initiated. The app stops reading sensor data as soon as the proximity sensor value changes indicating the phone has reached the ear of the user. To preserve the integrity of the data, calls made using a headset or in the speaker phone are not considered since the motion expected for answering the phone is not likely to be exhibited during these events. In such cases, we need a fallback mechanism.

4.3.2.2 Snap App:

In the Android system, the camera hardware can be used by any application that is registered with the permission to use it in its manifest file, i.e., there is no system-wide intent that can be intercepted by another application any time the camera is in use. Because of this restriction, we developed a custom application which is to be used as an alternative to other camera applications to test our hand movement gestures. The camera is capable of capturing the intent to start the camera "MediaStore.ACTION_IMAGE_CAPTURE". Once the camera is started, the sensor data is recorded until the user either exits the screen or takes a picture. A flag is set while the camera is on, that is changed once either the application exits the screen or the user takes a picture, at which point the sensors are stopped.

4.3.2.3 Tap App:

For reading NFC, our app waits for the NFC intent and starts recording the sensor data as soon as this intent is captured. Since the gesture/motion between the first time a device detects NFC tag and stops motion is very short, we captured the data for four seconds after it detects NFC tag. In our scheme, the gesture detection may be too late to be captured while the transaction may have already taken place through NFC. For this, we can place the received NFC data in quarantine or stall the NFC command received until the gesture is fully detected and analyzed if it is a valid gesture.

4.3.2.4 Snoop and Control App:

Along with different gestures, the app system also records data from various sensors at random point of times in order to compare these gestures with other activities. We refer to this data as "Snoop". To verify that our classifiers are indeed robust, we need to test them with the active motion data (besides the random Snoop events) which is different from the Call, Snap and Tap gestures, but still may have a chance to match with these gestures. For this purpose, we added extra features in our app system to collect controlled data (referred to as *Control* gestures). It provides a text box for the tester to specify the activity being performed and once the button is pressed for recording, it records all the sensor data for ten seconds.

The app displays the count of how many times each gesture has been performed and provides a button to upload the recorded gesture data to our server. The app is designed to upload the data at regular intervals (24 hours). A user can also explicitly upload the data by pressing the upload button.

4.3.3 Data Collection

We used our data collection app system to extensively collect data for Call, Tap and Snap gestures as well as various Control activities. We distributed the apps to volunteers who were willing to provide the sensor data collected from their devices through their normal activities. We distributed the app to the users in our respective Universities in US and Finland. Before distributing the app to these users, we explained what sensor data was being recorded, for how long it was recorded, at what occasions it was recorded and for what purpose the data was being used. Those who consented to our explanation were provided with apk files for the installation. There were a total of **23 users** recruited for our study. They were students of the Computer Science departments of the two Universities. Due to the real-world nature of our data collection, it was challenging to recruit users for our study, and, as such, our sample size is slightly lower than a typical lab-study, but still sufficient to demonstrate the promising feasibility of our approach. The devices used by the participating students were popular Android smartphones which have all the sensors needed for our tasks. Their devices had Android OS 4.0 (API 14) or later versions. Not all users could provide the required amount of data corresponding to all three of our apps (30 calls, 30 snaps and 30 taps).

Our experiment for the Call gesture was performed in real-world settings, i.e., the data was recorded when the volunteers made or received calls under normal use. We collected the data from each user until we got minimum of 30 samples for each gesture. This took about a month for most users. The Call App is fired whenever the participant made/received phone calls as explained in Section 4.3.2. Along with recording the sensor data when user made/received a call, it also collected snoop data at regular intervals. The app notifies the user whenever the sensor data is being recorded by displaying an icon with a message in the notification bar.

In a similar manner, we provided the Snap App to the users to collect data in real-world settings. They were asked to take pictures using our Snap App instead of the original camera app conforming to real-world settings. However, following this data collection methodology,

we obtained a little amount of data during the first phase of data collection. We came up with two conjectures for collecting so a little data from such use of the Snap App. First, a user might take many pictures when she is on vacation whereas during normal routine she might not take any pictures for weeks. Second, the app we developed was inferior to the default camera app developed by smartphone manufacturers. Our app did not provide various features (e.g., face recognition, touch focus, zoom in/out or HDR mode), usually provided by the default camera app. Therefore, users might have preferred the default camera app to Snap App. Hence, although the Snap experiment could have been done in real-world settings, we had to ask volunteers to take pictures in a lab setting mimicking real-world scenarios with our app. We posted a photo of "Mona Lisa" in the lab and asked volunteers to take her mugshot as shown in Figure 4.3. Thus, our Snap experiment is a semi-controlled experiment. We observed that some users preferred to take snaps in landscape mode while other preferred portrait mode.

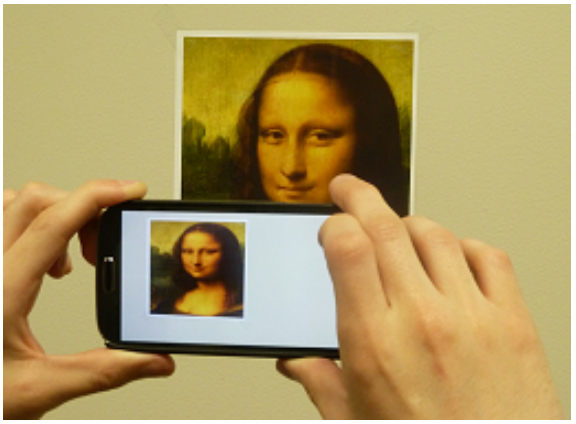


FIGURE 4.3: Taking a picture of Mona Lisa through Snap app

For the Tap experiment, we provided our Tap App to the volunteers. All of our participants possessed NFC enabled smartphones. However, since the NFC reader was not widely used in real life or by merchants, we had to limit our experiment to the lab settings. We attached a NFC tag in the student lab and asked users to tap on the tag at regular periods as shown in Figure 4.4. Whenever user tapped the NFC tag, Tap app handled the NFC intent as discussed in Section 4.3.2. User held back the device as soon as the app displays the toast message "NFC Detected". The App notifies about the data recording in notification bar.

Although we collected snoop data, we needed to make sure our app is robust against different kind of user activities. Since most of the random Snoop data might correspond to the device being stationary with no noticeable change in any sensor data, it might lead to false belief

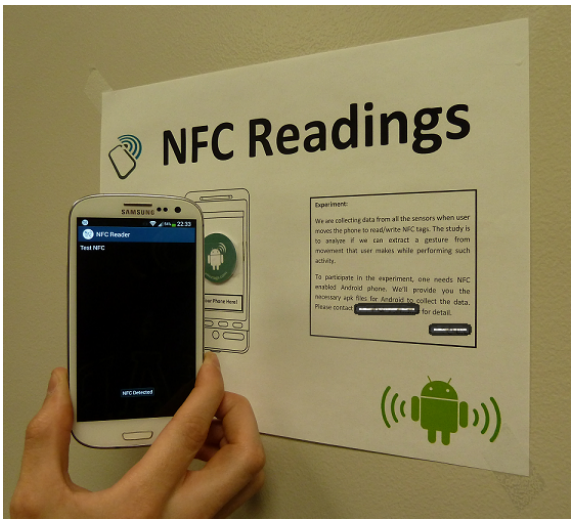


FIGURE 4.4: NFC Tap in Lab Settings

that our gesture classifiers were good enough to classify. Hence, we set out to evaluate the likelihood of false positives under different activities. We conducted several tests to emulate different user activities which might have a chance to match the Call, Snap or Tap gestures. For this experiment, our Control App was set to collect the sensor data for ten seconds as discussed in Section 4.3.2. We (volunteers) then performed different activities such as walking (upstairs/downstairs), running and jumping. We also mimicked reading phone's screen when there is a notification on device due to email or SMS, i.e., picking up the phone from table/desk to check message. Moreover, we mimicked writing email/SMS, and playing games in different orientations (landscape/portrait). We also performed a "drop test" to see if our classifier provides access to the permission requesting app when phone falls from pocket. For this, we dropped our phones from height of approximately 40 cm onto bed/couch. The activities could be exhibited by smartphone users in their day-to-day life (benign setting). An attacker could also coerce or fool the users into performing these activities with the hope that a false positive would occur allowing malware with access to the resource (adversarial setting).

4.3.4 Call-Snap-Tap Detection

In order to detect our Call, Snap and Tap gestures, we used the machine learning approach based on the underlying readings of the motion, position and ambient pressure sensors. We conducted several classification experiments to determine which off-the-shelf machine classifiers and which underlying sensor features provide the optimal performance. We used a total of nine sensors. The

TABLE 4.1: Sensors Utilized for Context Detection. Sensors marked with $^{\psi}$ were not used for the call/tap/snap detection. All sensors were used for NFC tapping. Sensors marked with $^{\omega}$ were not used for the walk biometrics.

Sensor	**Type**	**Function**
Accelerometer (A)	Motion	Acceleration force including gravity
Gravity (G)		Force of gravity on the device
Gyroscope (Gy)		Rate of rotation of the device
Linear Acceleration (LA)		Acceleration force excluding gravity
Rotation Vector (R)		Rotation vector of the device (uses geomagnetic field and gyroscope)
Game Rotation Vector (GR)	Position	Rotation vector of the device (does not use geomagnetic field)
Geomagnetic Rotation Vector (GMR)$^{\psi\omega}$		Rotation vector of the devices (uses magnetometer)
Magnetic Field (M)		Earth's magnetic field
Orientation (O)		Position of a device relative to the earth's frame
Pressure (P)$^{\omega}$	Environment	Ambient air pressure

sensors used in our analysis and their descriptions are depicted in Table 4.1.

All of the motion and position sensors used have three components corresponding to the three physical axes (X, Y, Z) at each instance. We calculated the scalar value, i.e., the square root of the sum of squares for each instance, which captures the significance of all the three axes. From this scalar value for each instance, we calculated mean and standard deviation for each of the sample for our different gestures, such as Call, Snap, Tap, Control and Snoop. This gave us eighteen features which we used for training and testing various off-the-shelf classifiers using Weka. We developed a Java program that utilizes Weka library to test different classifiers across different sensors subset that would result in best accuracy. We tested different machine learning algorithms provided by Weka: *Trees* – Logistic Model Trees (LMT), Random Forest (RF) and Random Tree (RT); *Functions* – Logistics (L) and Simple Logistic (SL), and *Bayesian Networks* – Naive Bayes (NB), on all sensors subsets.

The eighteen features were used as input to train each classifier to differentiate Call, Snap and Tap gestures from each other and from other gestures collected. We evaluated three training models for the classification task: (1) *user-specific model*, (2) *device-specific model* and (3) *generalized model*. The user-specific model requires each individual user to train a classifier herself before using the app. The device-specific model would require the app developer to build

specific classifiers for different phone models. The generalized model uses all the data from all different devices and all users to a build global classifier. These models have their own pros and cons. User-specific model would give better accuracy as it is tailored to an individual user but will require the user to train the classifier before using the system. The other two models do not have a user-centric training phase and will work right after the user installs the app, but the accuracy of this model might be lower than the user-specific model.

In our classification tasks, the positive class corresponds to Call/Snap/Tap and the negative class corresponds to other gestures (referred to as "Others"). Therefore, true positive (TP) represents Call/Snap/Tap that is correctly classified as Call/Snap/Tap, true negative (TN) represents Others that is correctly classified as Others, false positive (FP) represents Others misclassified as Call/Snap/Tap and false negative (FN) represents Call/Snap/Tap misclassified as Others.

As performance measures for our classifiers, we used Precision, Recall and F-measure (F1 score), as shown in Equations 4.1 & 4.2. Precision measures the security of the proposed system, i.e. the accuracy of the system in detecting the malware. Recall measures the system usability as low recall leads to high rejection rate of legitimate users' actions. To make our system usable, ideally we would like to have recall as close as 1.

$$precision = \frac{TP}{TP + FP}; \quad recall = \frac{TP}{TP + FN}; \tag{4.1}$$

$$F\text{-}measure = 2 * \frac{precision * recall}{precision + recall} \tag{4.2}$$

4.3.4.1 Call Detection

For the Call gesture, we could receive the desired data, i.e., the one corresponding to 30 (incoming/outgoing) phone calls, from **14 users**, as discussed in Section 4.3.3.

User-Specific Model: We divided the data into fourteen sets based on the users' ids. In order to build a classifier to distinguish Call from Other gestures for each set, we define two classes. The first class has the Call data from each user, and the other class contains the data collected from Snap, Tap, Snoop from the same user and the Control data collected from three out of the fourteen users. The average time for the collected Call readings was around one second, so we compared it with one second of every other gesture. To find the best subset of features and

TABLE 4.2: Results of using the optimal feature subset in the classification of Call and Others

Classification Model	User ID /Device	Classifier	Features Subset	Precision	Recall	F-Measure
User-Specific	1	L	A,G,LA,P	0.91	0.97	0.94
	2	SL	A,G,LA,O,P	0.97	0.93	0.95
	4	RT	GR,Gy,LA	0.87	0.90	0.89
	5	RT	GR,LA,O,P,R	0.87	0.83	0.85
	6	RF	G,LA,P,R	0.83	1.00	0.91
	7	SL	G,LA,M,P	0.88	0.97	0.92
	9	RF	A,Gy,M,P,R	0.94	0.97	0.95
	11	RT	A,G,Gy,M,P,R	0.85	0.97	0.91
	12	RF	G,Gy,LA,M,P	0.79	0.90	0.84
	13	RF	G,GR,M,P	1.00	0.97	0.98
	14	RF	GR,LA,M,R	0.97	1.00	0.98
	15	RF	LA,M,P,R	0.88	0.93	0.90
	16	L	G,Gy,M,P,R	0.97	1.00	0.98
	20	RF	G,Gy,P	0.88	0.93	0.90
Device-Specific	Google Nexus	RF	G,Gy,LA,R	0.90	0.83	0.86
	Samsung Galaxy	RF	A,GR,G,Gy, P,R	0.83	0.88	0.86
	HTC	RF	GR,LA,M,R	0.97	1.00	0.98
Generalized	ALL	RF	A,GR,LA,M,R	0.92	0.83	0.87

classifiers, we applied all the combination of sensors subsets and classifiers (specified above) to each of the fourteen user sets. The best features and the measurement values are calculated from running a 10-fold cross validation as shown in Table 4.2. The gesture detection performance can be termed as quite good. The recall and F-measure of the classifiers were on average 0.95 and 0.92 respectively. While the user is performing a Call gesture, he moves the phone which can be measured by the change in the acceleration and air pressure applied on the device. For this reason, Linear Acceleration and Pressure appear in almost all of the best sensors subsets. The Call gesture is unique per user, as shown in a prior work [68], which justifies why different sensors subset works well for different users.

Device-Specific Model: We grouped the data from the users who used the same devices. Seven users out of the fourteen users used Google Nexus, six used Samsung Galaxy and one used

HTC. As in the previous model, we applied all subset of features and different classifier onto these resulting three data sets. The best features and the measurement values are calculated from running a 10-fold cross validation are shown in Table 4.2. Combining the data from different users degrade the classifiers accuracy. The recall, precision and F-measure were above 0.83 for all the classifiers. The sensor subset for each device is subset of the sensors used by those users. For example, user 5, 6, 12, 15, 16 and 20 are the users who had Samsung Galaxy devices, therefore the best classifier subset for Samsung Galaxy is a subset of the sensors used by those users. User 14 is the only user who had HTC device. This is why the result for HTC is same as that for user 14.

Generalized Model: We combined the features from all the user into a single dataset. We used these features to generate a generalized classifier. The results are shown in Table 4.2 (last row). Similar to device-specific model, aggregating the data from different users degrades the classifier accuracy. The recall, precision and F-measure were all above 0.83.

4.3.4.2 Snap Detection

For the Snap gesture, we collected the data from **19 users** as described in Section 4.3.3. Each of these users performed 30 Snap operations. Similar to the Call gesture, we experiment three different classification settings for the Snap gesture.

User-Specific Model: We divided the data into nineteen sets based on the user id. Then, we built a classifier to distinguish the Snap gesture from other gestures collected by Call, Snap, Snoop and Control apps for each of the sets. The average time taken by the users to take a picture was four seconds. Therefore, we compared the Snap gesture with the four seconds of each other gesture. The best features and the measurement values are calculated from running a 10-fold cross validation as shown in Table 4.3. The precision, recall and F-measure are on average 0.98, 0.99 and 0.98, respectively. While the user is taking a picture, she moves the phone and adjusts the orientation of the phone, this can be measured by the Accelerometer, Pressure, Orientation, Gyroscope, Magnetic Field, Rotation and Game Rotation sensors. For that reason, all the sensors subset includes at least one of those sensors. The accuracy of Snap classifier is better than for the Call, which might be because the average time for Snap is four seconds and that for Call is one second.

Device-Specific Model: We aggregated the users' data based on their devices. The best features and the measurement values are calculated from running a 10-fold cross validation are shown in Table 4.3. The F-measure of the classifiers is above 0.92 for all the classifiers. We had only one user who had HTC phone in the Snap experiment; the result for the HTC is same as the results for user 14. For the other devices, the sensor subset is a subset of the best sensors subset of different users. Again, the classifier accuracy degraded when we combined data from different model phones.

Generalized Model: Here, we grouped the features from all the users into a single dataset. The results are shown in the last row of Table 4.3. The precision, recall and F-measure are 0.89, 0.93 and 0.91 respectively. The sensor subset consists of subset of the sensors used in the user-specific model.

4.3.4.3 Tap Detection

For the Tap gesture, we could collect the data from **20 users** as described in Section 4.3.3. Each of these users performed 30 NFC taps. We then performed three different experiments to evaluate our three classification models for the Tap gesture.

User-Specific Model: We divided the data into twenty sets according to the user id. Then, we built a classifier to distinguish the Tap gesture from other gestures collected by Call, Snap, Snoop and Control apps for each of the sets. The best features and the measurement values are calculated from running a 10-fold cross validation as shown in Table 4.4. The Tap gesture requires the user to move his device near to the NFC tag, the reading then starts, the user should keep the device on that position and then move it far from the NFC tag. Normally the users hold their devices in certain orientation while reading the NFC tag, finding the comfortable way to attach the NFC sensor in their device to the NFC tag. The phone movements can be measured by the change in Linear Accelerometer, Gravity, Accelerometer and Pressure sensors, and similar to Snap, the orientation can be measured by Orientation, Magnetic Field, Rotation and Game Rotation sensors. For that reason, each of the subsets includes at least one of each of those sets of sensors. The recall and F-measure were on average 0.98 and 0.97, respectively, which indicate very good performance.

TABLE 4.3: Results of using the optimal feature subset for the classification of Snap and Others

Classification Model	User ID /Device	Classifier	Features Subset	Precision	Recall	F-Measure
User-Specific	1	RF	P,Gy,A,R,LA	0.97	1.00	0.98
	2	SL	Gy,O,LA	0.97	1.00	0.98
	3	NB	M,P,GR	0.97	1.00	0.98
	4	L	G,P,M,GR	1.00	1.00	1.00
	5	NB	A,M,GR	1.00	1.00	1.00
	7	RF	A,P,M	0.97	1.00	0.98
	8	NB	O,A,M	1.00	1.00	1.00
	9	L	A,M,LA,GR	0.97	1.00	0.98
	10	RF	GR,M,O,P	0.97	1.00	0.99
	11	RF	G,Gy,P	1.00	1.00	1.00
	12	RF	A,G,Gy,P,R	1.00	1.00	1.00
	13	RF	GR,P	0.97	0.93	0.95
	14	RT	A,GR,G,R	0.91	1.00	0.95
	16	SMO	M,P,R	1.00	0.97	0.98
	18	L	A,LA,P	0.97	0.97	0.97
	19	RF	GR,Gy,P	0.97	1.00	0.99
	21	RT	G,LA,O,P,R	1.00	1.00	1.00
	22	RF	GR,Gy,P	1.00	1.00	1.00
	23	NB	A,GR,M,R	1.00	0.97	0.98
Device-Specific	Google Nexus	RF	A,GR,G,M, O,P	0.89	0.95	0.92
	Samsung Galaxy	RF	G,Gy,M,O,P	0.96	0.98	0.97
	HTC	RT	A,GR,G,R	0.91	1.00	0.95
	LG	RF	LA,M,P,R	1.00	0.98	0.99
Generalized	ALL	RF	A,LA,M,O,P	0.89	0.93	0.91

Device-Specific Model: We aggregated the users' data based on their devices. The best features and the measurement values are calculated from running a 10-fold cross validation are shown in Table 4.4. The average recall and F-measure are 0.96 and 0.94 respectively for all of the classifiers.

Generalized Model: We grouped the features from all the users into a single dataset and we used these features to generate a generalized classifier. The results are shown in Table 4.4 (last row).

TABLE 4.4: Results of using the optimal feature subset for the classification of Tap and Others

Classification Model	User ID /Device	Classifier	Features Subset	Precision	Recall	F-Measure
User-Specific	1	RT	G,M,GR	1.00	0.97	0.98
	2	NB	P,M	1.00	0.97	0.98
	3	RT	P,A,GR	1.00	0.97	0.98
	4	NB	GY,M,A	1.00	0.97	0.98
	5	NB	P,M	1.00	1.00	1.00
	6	RF	G,R,,GR	1.00	1.00	1.00
	7	RF	P,O,M	0.97	1.00	0.98
	8	NB	P,M	1.00	1.00	1.00
	9	NB	GR,M,P	0.94	1.00	0.97
	11	NB	A,Gy,M	0.97	1.00	0.98
	12	NB	A,G,O,P,R	1.00	1.00	1.00
	13	RF	O,P,R	0.92	0.96	0.94
	14	RF	A,LA,M	0.82	0.93	0.88
	16	RF	GR,M,O,P,R	1.00	0.97	0.98
	17	RF	A,GR,G,M,O	0.91	0.97	0.94
	18	RF	A,GR,G,M,O	0.97	1.00	0.99
	19	L	GR,G,O,P	1.00	1.00	1.00
	21	RT	M,P	0.97	1.00	0.98
	22	RF	Gy,O,P	1.00	1.00	1.00
	23	RT	GR,Gy,P	0.86	0.97	0.91
Device-Specific	Google Nexus	RF	GR,G,Gy,M, O,P,R	0.93	0.92	0.92
	Samsung Galaxy	RF	LA,M,O,P,R	0.96	0.97	0.96
	HTC	RF	A,G,LA,M	0.82	0.93	0.88
	LG	RF	A,G,M,O,P	0.96	1.00	0.98
Generalized	ALL	RF	GR,G,M,O,P	0.89	0.90	0.89

The precision, recall, and F-measure are above 0.89. The best sensors subset consists of the common sensors between all device models and some additional sensors.

4.4 Summary

In this chapter, we presented a novel approach to protecting sensitive mobile device services against many prominent attacks. The approach captures user's intent to access a given service

via user friendly explicit gesture or transparent implicit gestures. These gestures are either very simple, quick and intuitive or even transparent for the user, but would be very hard for the attacker to exhibit without user's knowledge. We presented the design and implementation of the hand waving gesture using different sensors, already available on most smartphones.

CHAPTER 5

CONTEXT ENHANCED AUTHENTICATION

5.1 Introduction

In the previous chapter, we have shown how the implicit and the explicit contexts can be used to authorize a benign app requesting for a sensitive permission. The implicit gesture/context has an advantage over the explicit gesture/context as the entire process is transparent to the user. This transparency is a key component when we need to authenticate users interacting with the Zero-Interaction Authentication ($\mathcal{Z}IA$) system.

In this chapter, we present how we can use two different implicit context provided by users in the form of implicit gestures to authenticate them from other users. First, we show that we can authenticate a person performing a tap gesture to process a transaction at the Point-of-Sale (POS) payment terminal using Near Field Communications (NFC) reader. Then, we show that a person can be authenticated using his gait and/or arm movement patterns so that he can access a $\mathcal{Z}IA$ system using multiple devices.

5.2 NFC Tap Authentication

NFC in smartphones allows a phone to communicate with any other NFC device (an external contactless reader or another NFC phone) when they are in close proximity, typically upon tapping to one another. This facilitates many important applications in day-to-day life including payments (using the phone essentially as a digital wallet), access control for buildings [113–115] and vehicles [116, 117], and public transit ticketing [118, 119], to name a few. The NFC technology, especially mobile payments, is already popular in many countries (e.g., China and Japan) [120] and has been gaining momentum in many other countries (e.g., the US). Introduction of Apple Pay [3], Android Pay [121] and Samsung Pay [122] have further boosted the growth of NFC payments.

With the rise of NFC deployments, a natural concern pertains to the security of NFC phones and NFC applications. One obvious and serious threat is that of loss or theft of NFC phones –

an unauthorized entity in physical possession of an NFC phone can fully compromise the NFC functionality leading to severe consequences (e.g., making hefty purchases on behalf of the user or entering the user's office premises). Given many current mobile users do not lock their phones (e.g., with a PIN or pattern) [123], the abuse of NFC services becomes a real threat. A report by Boyles et al. [124] estimates that nearly one third of cell phone owners have experienced a lost or stolen phone, and 12% have had another person access the contents of their phone in a way that made them feel their privacy was invaded. Lookout [125], from 2011 lost phone data of 15+ million users worldwide, reported that more than $11 million dollars worth of phones were lost during the Christmas alone and estimated that lost phone could cost $30 billion dollars in 2012 for the U.S. consumers alone.

To address this problem, many NFC apps (e.g., Google Wallet) authenticate the user prior to making an NFC transaction with a PIN or password. This approach, however, has two major problems. First, given PINs or passwords are only short and weak secrets (especially in the context of mobile phones with small form factors), they can be easily guessed or brute-forced [126–128]. Second, typing in the PIN or password for *each NFC transaction* can be tedious and potentially annoying for the user, thereby significantly undermining the usability of NFC technology as it was inherently designed for easy and fast transactions [129, 130].

Given the rather poor security and usability offered by PINs/passwords for the purpose of NFC user authentication, we set out to investigate a fully transparent and hard to compromise authentication mechanism. In short, we propose a *transparent behavioral biometrics* [131, 132] mechanism drawn from the gesture involving the tapping of a phone with a transaction terminal (e.g., an external NFC reader at point of service) while completing an NFC transaction.

5.2.1 Our Approach: Tap Biometrics

When a user makes an NFC transaction using her NFC-enabled device (let's say an NFC phone), she taps her phone to the transaction terminal and holds it for a while. When the transaction completes or gets interrupted, she removes her phone away from the terminal. These steps are illustrated in Figure 5.1. Tapping a phone to an NFC transaction terminal involves a particular motion of her phone which can be measured using different embedded sensors on the phone. The motion sensors and the position sensors can give us information about how the phone was moved. Also, there may be significant changes in the pressure as detected by the device when moved.

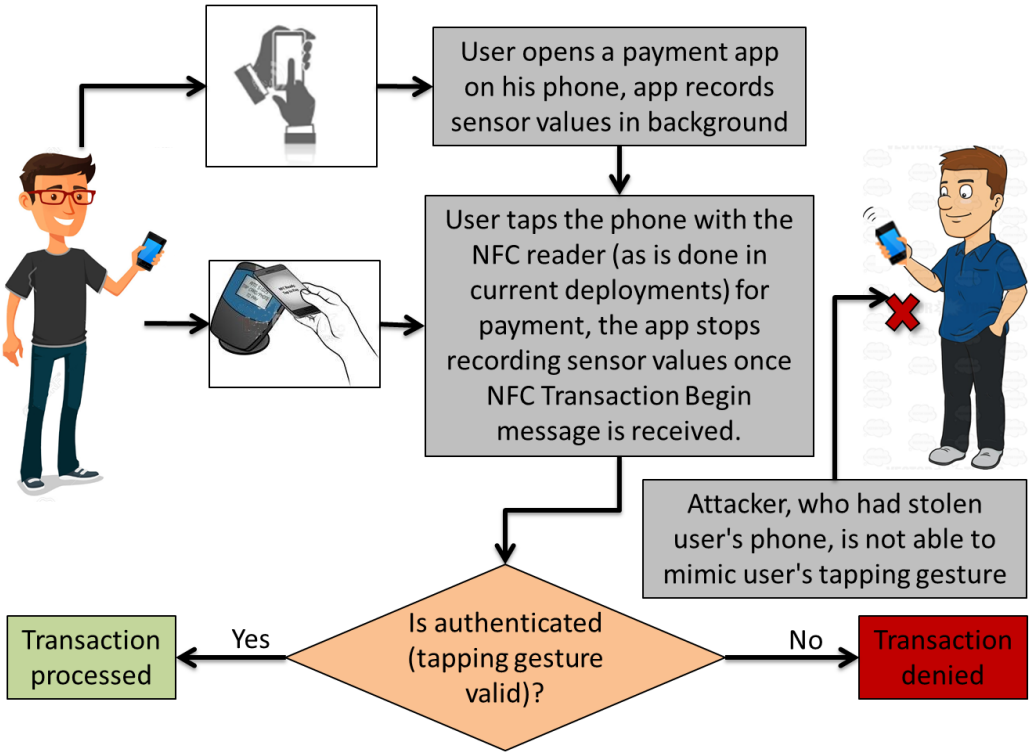


FIGURE 5.1: Overview of our system. The user gets authenticated just based on the uniqueness of his tapping gesture, a form of behavioral biometrics. The process is completely transparent to the user – no additional work is needed beyond what is currently done in NFC systems.

This can also be used to analyze how the device was moved.

In this dissertation, we show that the tapping gesture performed by a user before making NFC transactions is unique to the user and can be detected in a robust manner using machine learning classifiers and multiple sensors available on the phone. In the following section, we will demonstrate that our approach meets all of our design goals introduced in the prior section.

In our model, we add another layer of a security check on top of the default authentication system of Android and that of an NFC transaction app. Android can authenticate a user via different options such as passwords, PINs, face recognition, or fingerprint scanner. However, many users do not prefer to lock their phones. Also, using PINs or fingerprint scans for each transaction can be burdensome. We provide a way to authenticate the user before making the transaction in a way that does not require any explicit user action – just tapping the phone to the terminal (as is done currently) is sufficient. Our approach is invisible to users and requires no additional actions from the users. Our approach accurately identifies legitimate users and prevents unauthorized NFC transactions. It can also work seamlessly with other authentication

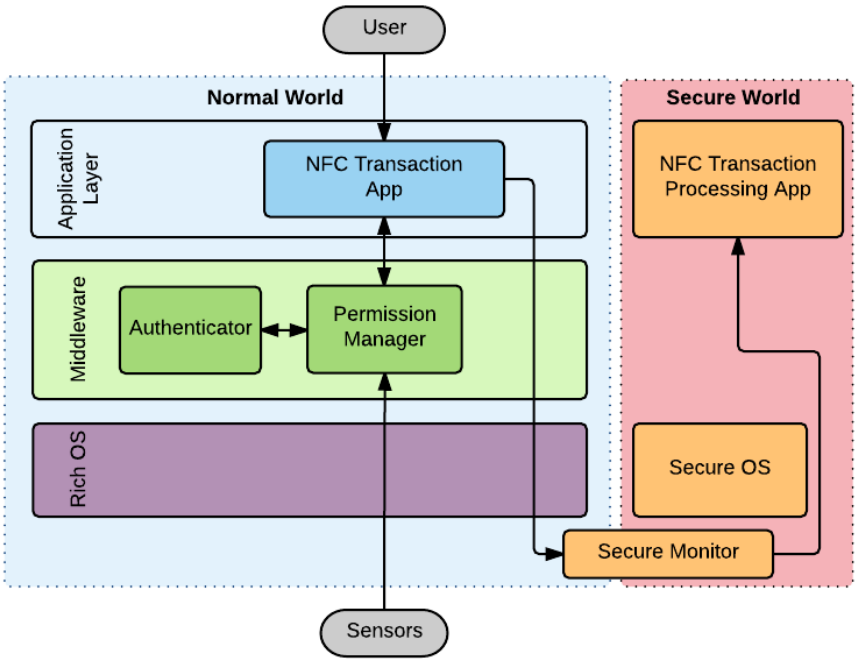


FIGURE 5.2: NFC Tap Biometrics Detection System Architecture: Control Flow

methods, such as PINs or fingerprint scans when used, to achieve strong two-factor security.

Figure 5.2 depicts the control flow for our approach. Our system analyzes the collected sensor values and compares with a pre-registered template of the user's tapping gesture. Our system grants permissions to complete NFC transactions if and only if the sensor values match with the user's tapping gestures. Our system includes four modules: (1) *NFC Transaction App* which provides the user interface and handles NFC communication, (2) *Transaction Processing Module* which processes the NFC transaction messages, (3) *Authenticator Module* which is a trained classifier that uniquely identifies the user's tapping gesture, and (4) *Permission Manager* that reads the sensor values, communicates with the Authenticator Module and grants the NFC Transaction App with the permission to interact with the Transaction Processing Module.

We assume that the Transaction Processing Module executes as a *Trusted application* inside trusted execution environment (TEE), e.g., ARM TrustZone [133]. ARM TrustZone divides a device platform into two execution environments, namely, *normal world* and *secure world*. The normal world is used to host rich Operating Systems (OS), like Android OS, and user applications while it allows processing of security sensitive codes in isolation within the secure world. The two worlds communicate with each other via secure monitor.

In our approach, the trusted application is responsible for processing transaction specific messages, handling necessary cryptographic operations and maintaining secrets like keys required for NFC transactions. On the other hand, NFC Transaction App running on the normal world handles user interactions and NFC communication. To authenticate a user based on her tapping gesture, our system begins collecting information from different sensors as soon as the user opens NFC Transaction App. Our system also records the time when the phone receives the first NFC message from the NFC transaction terminal. At this point, the user must have tapped her phone to the NFC transaction terminal and she is holding her phone towards the terminal to complete the NFC transaction.

Whenever the NFC Transaction App starts, it informs the Permission Manager to indicate that it has started. Permission Manager immediately starts collecting the sensor values. When the NFC Transaction App requires to process transaction messages, it requests the Permission Manager by sending NFC event begin time. The Permission Manager sends the set of appropriate sensor values to the Authenticator. Once the Authenticator confirms the tapping gesture as belonging to the user, the Permission Manager permits the NFC Transaction App to interact with the transaction module to complete the NFC transaction.

5.2.2 Application Design

To develop and evaluate our authentication mechanism based on tap gesture biometrics, we first needed to collect the tap gesture data from different users. After the data collection, different features were to be generated to robustly identify individual user data from other user data. We chose to implement our system in the Android OS. For the data collection, we created two modules: (1) *NFC Transaction Module* for a user to perform the tap gesture on a NFC transaction terminal which simulates NFC transactions, and (2) *Sensor Module* to record sensor values when the user performs the tap so that underlying data can be analyzed and later used to identify the user.

5.2.2.1 NFC Transaction Module

Android provides NFC Host Card Emulation APIs that allows the NFC-enabled phone to acts as a contactless card and allows NFC applications to communicate with external contactless readers. We designed our NFC module to simulate a real-world NFC transaction application. For this, we

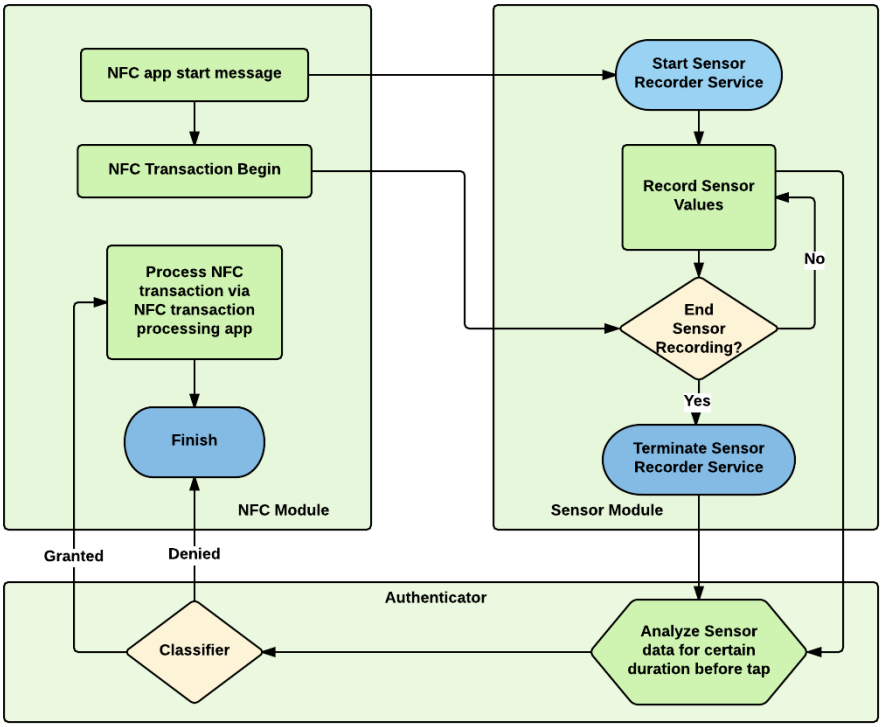


FIGURE 5.3: Sensor data collection flowchart.

chose to implement an NFC based public transit ticketing system. We designed and implemented both NFC ticketing application on the phone and the ticket reader application that controls the NFC transaction terminal. Both applications use a shared 128-bit AES key to authenticate each other during an NFC transaction. Specifically, we used three-pass mutual authentication protocol of MIFARE DESFire EV1 [1] as Kasper et al. [134] elaborated. Ticketing applications based on Mifare DESFire are widely used by public transit authorities around the world. NFC ticketing is only one aspect of an NFC transaction, nevertheless, it can be used as an analogy to understand user's NFC tapping gesture during any NFC transaction (e.g., for payments or building entry).

5.2.2.2 Sensor Module

Android platform provides several sensors that allow developers to monitor the motion of the device, the position of the device or the environment in which the device is. To be specific, the Android platform provides three broad categories of sensors, namely, motion sensors which

[1]MIFARE DESFire EV1: http://www.nxp.com/products/identification_and_security/smart_card_ics/mifare_smart_card_ics/mifare_desfire/series/MIFARE_DESFIRE_EV1_4K.html

measure acceleration forces and rotational forces along three axes, position sensors which measure physical position and orientation of the device, and environmental sensors which measure various environment parameters such as humidity, light illumination, ambient temperature, pressure and so on.

We created an Android service such that whenever the service is called by another activity or service, the service starts recording selected sensor values. The sensors we considered in our app are listed in Table 4.1. The sensors values are logged along with the timestamps so that they can be used for statistical analysis later on. When the calling app sends the stop service command to the service, the service stops recording the sensor data. The flow chart of this data collection process is shown in Figure 5.3.

5.2.3 Data Collection

To develop and evaluate our approach, we first needed to collect data from multiple users. We also wanted to capture various types of gestures that users make while tapping their NFC-enabled phone to the terminals installed for different types of NFC applications. There is no standard instruction on how NFC transaction terminals should be placed, e.g., they can be placed horizontally, vertically or at certain angle from the surface where they are placed at the NFC transaction terminals. We designed our data collection engine to capture four different scenarios based on how the NFC transaction terminals may be installed: (1) *Waist-Flat*: horizontally at the height of 0.75-1 meter above the ground, (2) *Waist-Angular*: at 45 degree angle with horizontal surface at the height of 0.75-1 meter above the ground, (3) *Chest-Angular*: at 45 degree angle with the vertical surface at the height of 1-1.5 meter above the ground, and (4) *Chest-Vertical*: vertically at the height of 1-1.5 meter above the ground. A user tapping an NFC reader in the waist-flat scenario is shown in Figure 5.4. We implemented an app as discussed in the Section 5.2.2 and collected data using Google Nexus 5 as our phone model. We used NFC reader ACR 122U as the transaction terminal.

As the user opens the app to make NFC transactions, our system runs in the background as a service as mentioned in the Section 5.2.2.2. We continuously recorded the sensor values for the experiment and detailed analysis, however, in the real-life implementation, the sensors can be turned off as soon as the transaction success message is received or shortly thereafter.

For data collection, we invited volunteers to our lab via word of mouth. These volunteers

FIGURE 5.4: A user tapping an NFC reader at waist-flat position. In waist-flat scenario, the NFC reader is kept at the height of 0.75-1 m from the ground and horizontally on the table.

were university students from different countries situated in the US and Finland. There were a total of 20 volunteers (17 male and 3 female, between the age of 25-35) who participated in our study. We only observed four left handed users, while rest of them were right handed, and none of them swapped their phone from one hand to other during the experiment. The experiment was performed in lab settings. We provided a smartphone to the volunteers and asked them to tap it to the reader. Each user opened the app, tapped to the reader to initiate an NFC transaction and held it there until he/she was notified about the transaction complete message as displayed on the phone. Then the user brought the phone away from the reader. We asked each user to pause for a few seconds before he/she tapped again for another transaction.

In one session, we asked the user to tap and perform the transaction five times for each of the four different reader positions mentioned above, i.e., after the user tapped the reader five times, we changed the position of the reader to a different setting. Hence, in a session, we collected 20 tap gesture samples from each user (five each for four different reader positions). We conducted six sessions collecting 120 tap gesture samples for each user (30 samples of data for each of the four positions of the reader). These six sessions were conducted in time spans ranging from either one day to six days depending upon the availability of the volunteers. However, each session had sufficient gap to break the user's rhythm of tapping and add variation to the user's hand motion.

5.2.4 Tap Biometrics Detection

5.2.4.1 Set-Up and Design

In order to evaluate the feasibility of the proposed tap gesture biometrics as an authentication scheme, we utilized the machine learning approach based on the underlying readings of the motion sensors, the position sensors and the ambient pressure sensors (the different sensor employed are listed in Table 4.1).

Classifier: We utilized the Random Forest classifier in our analysis. Random Forest is an ensemble approach based on the generation of many classification trees, where each tree is constructed using a separate bootstrap sample of the data. In order to classify a new input, the new input is run down on all the trees and the result is determined based on majority voting. Random trees have been shown to be a strong competitor to Support Vector Machine (SVM), and its performance frequently outperforms SVM [135]. Random Forest is efficient, can estimate the importance of the features, and is robust against noise [135].

Features: For each of the position and the motion sensor instances, we calculated the square root of the sum of squares for that instance's axes components (X, Y, Z), such that it captures the significance of all the three axes. Then, we calculated the mean and the standard deviation of all the instances in the sample that corresponds to a single tap. This gave us twenty features, which we used for training and testing of the Random Forest classifier.

The twenty features were used as input to train the classifier to differentiate a user from other users. We evaluated two training models for the classification task: (1) *scenario-specific model*, and (2) *general model*. The scenario-specific model requires each user to train a classifier on all reader (transaction terminal) positions (described in Section 5.2.3) before using the app. This model assumes that the classifier knows or is informed about the position of the reader (i.e., the scenario for the transaction). The generalized model, in contrast, uses all the data from all different scenarios of the user and builds a global classifier per user regardless of the reader position. Moreover, we have tested multiple gesture duration by utilizing the sensor data of one, two and three seconds before the transaction begins. Our goal was to determine the optimal duration of the tapping gesture which can uniquely identify each user.

In all of the classification tasks, the positive class corresponds to the tap gesture of the

legitimate user and the negative class corresponds to impersonator (other user). Therefore, true positive (TP) represents the number of times the legitimate user is granted access, true negative (TN) represents the number of times the impersonator is rejected, false positive (FP) represents the number of times the impersonator is granted access and false negative (FN) represents the number of times the correct user is rejected.

As performance measures for our classifiers, we used Precision, Recall and F-measure (F1 score), as shown in Equations 4.1 and 4.2. Precision measures the security of the proposed system, i.e., the accuracy of the system in rejecting impersonators. Recall measures the usability of the proposed system as low recall leads to high rejection rate of the legitimate users. F-measure considers both the usability and the security of the system. To make our system both usable and secure, ideally, we would like to have F-measure as close as 1.

5.2.4.2 Classification Results

General Model: As mentioned in Section 5.2.3, we collected data from 20 users. Each user performed a total of 120 taps. We divided the collected data into 20 sets based on the users' identities (ids). In order to build a classifier to authenticate a user based on her tapping biometrics, we defined two classes. The first class contains the Tap data from a specific user, and the other class contains randomly selected Tap data from other users. We analyzed three different duration of the tapping gesture, by considering one, two and three seconds before the transaction begins.

After running a 10-fold cross validation, we obtain results for different duration and different scenarios. The results show that one second of sensor data is enough for authenticating the user, shown with high F-Measure, recall and precision. Increasing the gesture duration did not improve the accuracy; it would rather decrease the accuracy as it may incorporate random user movement before the actual tapping gesture starts. We summarize the results for different scenarios with different durations of the tap gesture in Table 5.1. These results suggest that increasing the tap duration does not seem to increase the accuracy and therefore one second duration seems optimal. Hence, the rest of the experiments reported after this section are conducted with the one second duration of the tap gesture.

In our experiment, 12 out of the 20 users performed all the tapping in one day, and, for this sub-group of users, the average and standard deviation (for tapping duration of 1 second before)

TABLE 5.1: Performance of the classifier for Generalized and Scenario-specific models. Each column shows average (Avg) and standard deviation (S.D.) for F-Measure, Recall and Precision for tapping duration of one second, two seconds and three seconds respectively. Precision captures the security of the system while recall captures the usability of the system. F-measure accounts for both precision and recall. The model seems to perform equally well for all the three different duration of the tapping gestures.

	One Second			**Two Seconds**			**Three Seconds**		
	F-Measure	Recall	Precision	F-Measure	Recall	Precision	F-Measure	Recall	Precision
	Avg (S.D.)	Avg (S.D.)	Avg (S.D.)	Avg (S.D.)	Avg (S.D.)	Avg (S.D.)	Avg (S.D.)	Avg (S.D.)	Avg (S.D.)
Generalized	0.93 (0.05)	0.97 (0.03)	0.91 (0.08)	0.93 (0.05)	0.95 (0.04)	0.9 (0.07)	0.92 (0.06)	0.96 (0.03)	0.89 (0.08)
Chest-Angular	0.89 (0.06)	0.92 (0.06)	0.87 (0.07)	0.89 (0.06)	0.93 (0.05)	0.86 (0.09)	0.90 (0.05)	0.93 (0.06)	0.87 (0.05)
Waist-Flat	0.91 (0.06)	0.95 (0.05)	0.88 (0.07)	0.92 (0.05)	0.95 (0.06)	0.90 (0.06)	0.90 (0.06)	0.93 (0.05)	0.87 (0.08)
Chest-Vertical	0.92 (0.07)	0.94 (0.05)	0.89 (0.09)	0.91 (0.04)	0.93 (0.04)	0.89 (0.05)	0.90 (0.06)	0.93 (0.05)	0.88 (0.08)
Waist-Angular	0.91 (0.06)	0.95 (0.04)	0.88 (0.08)	0.89 (0.07)	0.92 (0.06)	0.86 (0.08)	0.89 (0.06)	0.91 (0.07)	0.87 (0.07)

were 0.97 (0.03) for these users. The data collection from the rest of the users spanned between 4 and 22 days, and, for these users, the average and standard deviation of the F-Measure dropped to 0.88 (0.03). In practice, the classification models can be re-trained as the user makes new successful transactions such that the accuracy does not drop as the time gap between the testing and training data increases.

Scenario-Specific Model: In our scenario-specific model, we divided the collected data into 80 sets based on the user's ids and the scenario's (reader positions) id. In order to build a classifier to authenticate the user based on the tapping in a given specific scenario, we define two classes. The first class has the tap gesture data from a specific user in a specific scenario, and the other class contains randomly selected data from other users corresponding to the same scenario.

The classification results are calculated after running a 10-fold cross validation and shown in Table 5.1. The classification accuracy for the scenario-specific model is less than its correspondent in the general model. This may be due to the reduced number of instances in each of the files (30 versus 120 in the general model). However, both models seem to perform about equally well in detecting the tap biometrics.

5.2.4.3 Summary of Results

The results obtained from both the classification models show that the tap gesture can be detected in a robust manner and thus will serve as an effective method for authenticating the users of NFC devices. This is reflected in high precision, recall and F-measure for both models. The general model can be used in applications where the user can train the model with tapping gestures in different scenarios (reader positions). The scenario-specific model can be used in practice

when the phone can acquire the knowledge about the reader position. This knowledge can be acquired either by asking the user about the reader position, although this will require some user involvement in the authentication process, or the terminal can send its position to the phone.

5.2.4.4 Power Analysis

Since our app records sensor values, we set forth to analyze if our system is lightweight. To measure the battery power consumption, we used $PowerTutor$ [136]. $PowerTutor$ is an app readily available on Google PlayStore[2] which estimates the power/energy consumed by different apps installed on the phone. The app provides the power/energy consumed by apps based on various parameters such as screen brightness, CPU usage, Wi-Fi polling and so on. We compared the energy consumed by our app with $NFCtools$[3], one of the most popular apps for NFC in Google PlayStore. We logged the energy consumption for both apps accounting for CPU usage only.

We ran $PowerTutor$ app to monitor the power consumption of all the apps on the phone. Then, we performed 20 taps with our app against the NFC reader, and then we performed 20 taps with $NFCtools$ against an NFC tag. We observed that our app consumes 0.2 J of energy per tap compared to 0.13 J of energy per tap by $NFCtools$. This shows that our system is lightweight as it only uses an additional 0.07 J of energy for the sensor recordings.

5.2.5 Active Adversarial Attack

From the analysis presented in Section 5.2.4, we can see that our approach is robust and can authenticate users with a high accuracy. That is, the approach can be effectively used to differentiate one user from the other. However, it is possible that the attacker may deliberately attempt to mimic the tapping gesture exhibited by a victim user. In this section, we assess our tap biometrics system against such an active adversary.

If the attacker tries to authenticate himself as the victim user, he has to move his hand in such a way that his hand motion as sensed by different sensors correlates significantly with the tapping gesture exhibited by the legitimate user. Even when the attacker observes how the user taps, it may still be difficult for the attacker to reproduce the tapping gesture as our gesture is

[2] https://play.google.com/store/apps/details?id=edu.umich.PowerTutor
[3] https://play.google.com/store/apps/details?id=com.wakdev.wdnfc

sensed by multiple sensors and *all* of the sensor values should match with the user's template. Mimicking multiple sensor events simultaneously would be harder for the attacker and so our approach should provide better resistance to active attacks compared to systems that use single or fewer sensors. While robotic attacks have been reported against other authentication systems (such as the one developed by Serwadda et al. [137], such attacks will not apply to our system since authentication is to be performed by a real human user in the presence of retail personnel and using a robot to make a purchase at the terminal would clearly raise a suspicion.

We proceeded to evaluate the robustness of our system against human-based observation and active adversarial attacks. For our evaluation, we designed an active attack that aimed at maximizing the attacker capabilities in defeating our system. If our system could defeat this attacker, it could also defeat other weaker attackers. To this end, we asked one of our users to serve the role of the victim while a researcher served the role of an expert attacker. The victim and the attacker had similar body structures, i.e., their height and weight were similar, which would have facilitated the attacker to better mimic the victim's tapping gesture. We asked the victim user to perform his tapping for 30 times in each of the four reader orientation scenarios while the attacker recorded a clear video of him tapping. After the total 120 taps were collected from the victim, we built the classifier for this user following the procedure described in Section 5.2.4. The attacker then closely watched the previously recorded video and practiced to re-create the victim's tapping gesture against a dummy reader several times while receiving a feedback from his friend (a colluding attacker). This simulated the attacker's training phase performed at home (i.e., not at the retail store in the presence of the authentication terminal/reader). Finally, during the actual attack phase, the attacker performed 20 taps and each of these taps was tested against the victim's classifier built earlier.

The success rate of this active attacker against our authentication systems is shown in Table 5.2. From these results, we can claim that even when an attacker practices and mimics the hand motion of the victim, he cannot succeed. Also, we can claim that we have a strong attacker as the attacker is fully trained watching the victim's tap video recording and getting feedback from a colluding attacker. Moreover, the victim we chose matched the body structure of the attacker which may further facilitate the attack. Since our system can defeat such strong attacker, it can, therefore, defeat attacks in other scenarios where the victim's structure is different from the attacker's and/or where the attacker cannot fully observe the victim and train.

TABLE 5.2: The results for the active attack with tap duration of one second. The performance of the classifier built using 120 taps for generalized classification model as well as using 30 taps for different scenario specific classification model for the particular victim is shown. The last column shows the attack success rate FPR (False Positive Rate) for the corresponding classifier. FPR represents the rate at which the attacker was falsely classified as the victim. The attacker was not successful at all in mimicking the victim's tap gesture.

	Victim			Attacker
	F-measure	**Recall**	**Precision**	**FPR**
Generalized	0.983	0.975	0.992	0
Chest-Angular	0.984	1	0.968	0
Waist-Flat	0.967	0.967	0.967	0
Chest-Vertical	0.968	1	0.938	0
Waist-Angular	0.984	1	0.968	0

5.2.6 Summary

In this section, we presented an approach to authenticate a user transparently before making an NFC transaction. The approach captures the user's hand movement and identifies the user based on the sensor data recorded by the device. The gesture is very unique to the user and is difficult for the attacker to mimic. We presented the design and implementation of the proposed authentication approach. Our results suggest that our approach could be very effective in authenticating users and preventing misuse of NFC services in case of theft or loss of NFC phone, without necessitating any additional user burden.

5.3 *WUZIA*: Walk Unlock ZIA

Zero-interaction authentication ($\mathcal{ZIA}$) [29] represents a rapidly emerging paradigm, in which a verifier device authenticates a prover device in physical proximity of the verifier while requiring *no interaction* by the user of the prover device. The user, carrying the prover, usually just walks towards the verifier and the verifier gets unlocked automatically. In this approach, the prover and verifier devices pre-share a security association, and simply execute a challenge-response based protocol for the verifier to authenticate the prover.

The zero-interaction requirement is intended to improve the usability of the authentication process, which may increase the chances of adoption. Indeed, $\mathcal{ZIA}$ systems are already getting deployed in many real-world application scenarios. For example, BlueProximity [138] allows a user to unlock the idle screen lock in her computer merely by physically approaching the

computer while in possession of a mobile phone, without having to perform any other action, such as typing in a password. Other $\mathcal{ZIA}$ systems include: "Passive keyless entry and start" systems like "Keyless-Go" [139], PhoneAuth [108], and access control systems based on wearable devices [140].

However, the zero-interactive nature of $\mathcal{ZIA}$ systems opens up a fundamental vulnerability — unauthorized physical access to the prover device, e.g., during lunch-time or upon theft, would allow an attacker to have unfettered access to the verifier device. Since the prover device does not require any authorization from the user prior to responding to the verifier device in a $\mathcal{ZIA}$ authentication session, mere possession of a lost or stolen prover device is sufficient to gain access to the verifier device. Since users' personal devices and items (e.g., smartphones or car keys) are prone to loss or theft, this issue makes the $\mathcal{ZIA}$ systems inherently weak and insecure. Speaking about statistics, digital trends [141] reports that Americans lost $30 billion worth of mobile phones in 2011. Moreover, the trend has been increasing as reported by Lookout [142] that 3.1 million Americans consumers were victims of smartphone theft which is double the number reported in 2012 by Consumer Reports [143].

This raises an important research challenge: *how to protect the $\mathcal{ZIA}$ systems in the face of loss or theft of prover devices, while still keeping the authentication process transparent and zero-interactive for the user?*. In this section, we set out to address this challenge by the use of walking or gait pattern biometrics prior to authorizing a $\mathcal{ZIA}$ authentication session. In other words, the prover device carried by the user will respond to the authentication session with the verifier device only when it (the prover device) detects that it is being carried by the legitimate user. As the user walks towards the verifier device, the prover device first detects the walking pattern of the user, and only then gets unlocked and responds to the verifier device. Since a user's walking pattern is believed to be unique, only that user (no imposter) would be able unlock the prover device to gain access to the verifier device in a $\mathcal{ZIA}$ session. Since the user has to nevertheless walk towards the verifier device as part of the $\mathcal{ZIA}$ authentication process, no additional effort is imposed on the user, thereby preserving the zero-interactivity and user-transparency requirement.

While walking-based biometrics schemes have been studied in prior literature for other application settings (e.g., [75, 76, 80, 81, 83, 86, 91, 92, 94]), our main novelty lies in two important aspects:

1. The use of *multiple sensors* available on the current breed of devices (e.g., accelerometer, gyroscope and magnetometer).

2. The use of *multiple devices* carried by the user, in particular, an "in-pocket" smartphone and a "wrist-worn" smartwatch. Each of these devices capture unique physiological and behavioral facets of the user's walking pattern (e.g., phone captures hip movement and watch captures hand movement).

5.3.1 Our Approach: Walk Unlock ZIA

To protect the unlocking of $\mathcal{V}$ in the face of loss or theft of $\mathcal{P}$ in a $\mathcal{ZIA}$ scheme, we propose to authenticate the user based on a gait-based authentication system [32]. In other words, we propose to authenticate the user with her unique walking pattern. Different categories of sensors are embedded nowadays into smartphones and smartwatches such as motion, position and environment sensors. Android OS, one of the most popular smart device operating systems, provides APIs to support different categories of these sensors. We leverage these sensors, especially motion and position sensors, to identify that the $\mathcal{P}$ device is undergoing a particular activity, in a specific motion and orientation, as if the prover device is being carried/worn by the legitimate user. This activity detected by the $\mathcal{P}$ device is transparent to the user since it is performed implicitly while the user walks towards $\mathcal{V}$.

While many types of $\mathcal{P}$ devices may be used to detect the user's walking activity prior to authorizing a $\mathcal{ZIA}$ session, in this work, we capture the walking biometrics using an "in-pocket" device and/or a "wrist-worn" device, both devices having multiple on-board sensors. Specifically, in such a walk-unlock $\mathcal{ZIA}$ (*WUZIA*) scheme, we aim to authenticate the user in a robust manner using machine learning classifiers based on data drawn from multiple sensors from multiple devices such as smartphone (in-pocket) and smartwatch (wrist-worn). The *WUZIA* authentication process has been visualized in Figure 5.5. As shown in Figure 5.5, *WUZIA* requires changes only in the $\mathcal{P}$ devices. The $\mathcal{V}$ device in an existing $\mathcal{ZIA}$ system is transparent to the authentication process and requires no modification. Hence, *WUZIA* can be implemented in traditional $\mathcal{ZIA}$ system such as BlueProximity [138] by just changing the smartphone app, without changing the terminal software.

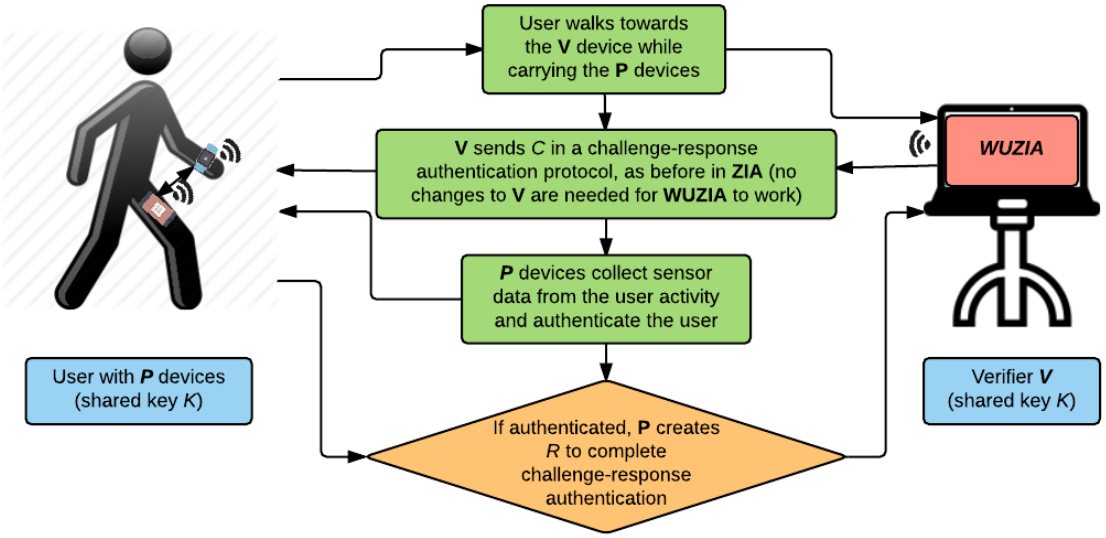


FIGURE 5.5: *WUZIA* system overview: the $\mathcal{P}$ devices respond to the $\mathcal{V}$ device in a challenge-response *WUZIA* authentication only if the $\mathcal{P}$ devices detect the valid walking pattern of the user. We consider a smartphone and a smartwatch as the $\mathcal{P}$ devices.

In our *WUZIA* system, we use multiple devices, i.e., a smartphone and a smartwatch, to authenticate the user. However, to analyze the efficiency and robustness of our system systematically, we show:

1. walking pattern extraction using the in-pocket smartphone,
2. walking pattern extraction using the wrist-worn smartwatch, and
3. combination of the above two.

The second setting is suitable for situations where the user may leave her phone on the desk space or the car dashboard, and will need to be logged in just by using her watch. Although currently most of the smartwatches work along with companion devices (smartphones), we believe that in the future such devices would be usable as stand-alone devices.

The threat model of *WUZIA* (Section 2.2.2.1) is in line with that of $\mathcal{ZIA}$, except that the former aims to be secure even under the adversarial possession of $\mathcal{P}$. Since in the proposed scheme, $\mathcal{P}$ can be either a smartphone or a smartwatch or both, the attacker may therefore possess only one of the devices or both devices. After the attacker possesses $\mathcal{P}$ (one or both devices), it will try to unlock $\mathcal{V}$. Further, a *WUZIA* attacker may be active in the sense that it may try to authenticate itself as the valid user by mimicking the walking pattern of the user as measured by

$\mathcal{P}$ device(s). We allow such an attacker to observe (and record) the user in an attempt to imitate the user's walking habits.

In the *WUZIA* system, we assume that a relay attack prevention technique has already been deployed (like in a $\mathcal{Z}IA$ system). That is, no relay attacks are possible between $\mathcal{P}$ and $\mathcal{V}$. Similarly, we assume that no relay attacks are possible between the $\mathcal{P}$ devices (phone and watch). Also, we assume that the two $\mathcal{P}$ devices are securely paired with each other and that all communication between them has been protected with traditional cryptographic mechanisms.

Given this threat model, in the following sections, we will show that our *WUZIA* system satisfies all of our design goals, i.e., being lightweight, efficient, robust and transparent.

5.3.2 Application Design

To develop and evaluate our system for authenticating the users based on their walking pattern, we need to collect the sensors data from the users' smartphones and smartwatches while they are walking. We developed a framework that encompasses two Android apps and a web app. The web app utilizes Google Cloud Messaging (GCM) to send commands to the smartwatch. One of the Android apps is installed on the smartphone and the other is installed on the smartwatch.

5.3.2.1 Web App

We used GCM to send start/stop commands to the smartphone, which upon receiving start/stop recording the sensors data and send start/stop recording trigger to the smartwatch. We created a simple HTML page with a text box to record the user information, a start recording button, and a stop recording button. The experimenter first inputs the user information in the text box and hits the start recording button when the user starts walking towards $\mathcal{V}$. When the user touches $\mathcal{V}$, the experimenter hits the stop recording button. We used GCM for the purpose of data collection only (in real-life implementation, GCM is not needed).

5.3.2.2 Smartphone App

The app on the smartphone waits for the GCM commands. As soon as it receives the GCM start command, it sends a start recording trigger to the smartwatch and starts recording the sensors

value. As soon as it receives the GCM stop command, it sends a stop recording trigger to the smartwatch and stops recording the sensors value.

5.3.2.3 Smartwatch App

The app on the smartwatch waits for the smartphone's triggers. Once it receives a start recording trigger, it starts recording the sensors values and keeps on recording until it receives a stop recording trigger. The recorded sensor values by the smartwatch are stored in the smartwatch.

The sensors utilized in our implementation, from both the smartphone and the smartwatch, are listed in Table 4.1.

5.3.3 Data Collection

For data collection, we recruited 18 students in our University through the word of mouth. Among these participants, 15 were male while 3 were female. To avoid any kind of inconsistency, we used only one smartphone (LG Nexus 5 (D820) [144]) and one smartwatch (LG G watch R (W110) [145]). Both devices have Android OS version 6.0.1. The participants were clearly informed about the experiment such as the data being collected, the purpose of the experiment, and that they can refuse to participate in the middle of the experiment or even request to delete their collected data during or after the experiment has been conducted. Our University's Institutional Review Board approved the project.

After the participants were detailed about the experiment, we asked these volunteers to wear the smartwatch on their (left/right) hand where they normally wear their watch and put the smartphone in their (left/right) pocket where they normally put it during walking. We asked each volunteer to walk from a door to the computer (distance of around 7 meters) as if they are trying to log in. The experimenter sent the GCM command to the smartphone to start the sensors recording when the user started walking. As soon as the user touches the keyboard as if the user is trying to log into the computer, the experimenter sent another GCM command to stop the sensors recording. We noticed that some of the users log into the machine standing while others sit on a chair before they touch the keyboard. One of the participants even placed his phone on the desk before he logged into the machine.

We collected the data from these volunteers for a period of time ranging from 30 to 60 days based on their availability. We asked each user to walk for around 7 meters (from the door to the machine) for five times each day. We collected the data from each user for 10 days resulting in 50 samples of walking data from each user.

5.3.4 Walk Biometrics Detection

In order to evaluate the performance of the proposed gait biometrics as an authentication scheme, we utilized the machine learning approach based on the underlying readings of the motion sensors, and the position sensors from both of the phone and the watch.

5.3.4.1 Design

In order to evaluate the feasibility of the proposed gait based biometrics as an authentication scheme, we utilized the machine learning approach based on the underlying readings of the motion sensors, the position sensors and the ambient pressure sensors (the different sensor employed are listed in Table 4.1).

Classifier: In our analysis, we utilized the Random Forest classifier. Random Forest is an ensemble approach based on the generation of many classification trees, where each tree is constructed using a separate bootstrap sample of the data. To classify a new input, the new input is run down on all the trees and the result is determined based on majority voting. Random Forest is efficient, can estimate the importance of the features, and is robust against noise [135]. Random Forest outperforms other classifiers including support vector machines which are considered to be the best classifier currently available [135, 146, 147].

Features: For each of the used sensor instances, we calculated the mean, the standard deviation and the range of each of the axis (X, Y, Z), the square of each axis (X^2, Y^2, Z^2) and the square root of the sum of squares for that instance's axes components (X, Y, Z) of all the instances in the sample that corresponds to a single walk instance. Twenty one features are extracted from each of the used sensors, which give us a total of 336 features.

The 336 features or subset of them were used as input to train the classifier to differentiate a user from other users. In the classification task, the positive class corresponds to the gait of the legitimate user and the negative class corresponds to impersonator (other user). Therefore, true

TABLE 5.3: Performance for the classifier for three different categories. The first three rows show the performance of the classifier using all the sensors. The next three rows show the results of using the sensors subset that provides the best average results. The last three rows show the result of using the best sensors subset for each user. Highlighted cells emphasize the most interesting results.

		FNR	FPR	F-Measure	recall	precision
		Avg (std. dev.)				
Overall	**Phone Only**	0.058 (0.037)	0.068 (0.034)	0.937 (0.026)	0.942 (0.037)	0.934 (0.031)
	Watch Only	0.085 (0.050)	0.105 (0.045)	0.906 (0.036)	0.915 (0.050)	0.899 (0.040)
	Both	0.038 (0.047)	0.042 (0.031)	0.960 (0.030)	0.962 (0.047)	0.960 (0.029)
General	**Phone Only**	0.040 (0.035)	0.051 (0.033)	0.954 (0.025)	0.960 (0.035)	0.950 (0.031)
	Watch Only	0.080 (0.049)	0.095 (0.043)	0.913 (0.030)	0.920 (0.049)	0.909 (0.038)
	Both	0.022 (0.027)	0.030 (0.027)	0.974 (0.021)	0.978 (0.027)	0.971 (0.025)
Individual	**Phone Only**	0.018 (0.023)	0.036 (0.020)	0.973 (0.013)	0.982 (0.023)	0.965 (0.019)
	Watch Only	0.046 (0.034)	0.063 (0.044)	0.947 (0.024)	0.954 (0.034)	0.941 (0.039)
	Both	0.002 (0.006)	0.003 (0.008)	0.997 (0.005)	0.998 (0.006)	0.997 (0.008)

positive (TP) represents the number of times the legitimate user is granted access, true negative (TN) represents the number of times the impersonator is rejected, false positive (FP) represents the number of times the impersonator is granted access and false negative (FN) represents the number of times the correct user is rejected.

As performance measures for our classifiers, we used false positive, false negative, precision, recall and F-measure (F1 score), as shown in Equations 4.1 and 4.2. FP/precision measures the security of the proposed system, i.e., the accuracy of the system in rejecting impersonators. FN/recall measures the usability of the proposed system as high FN leads to high rejection rate of the legitimate users. F-measure considers both the usability and the security of the system. To make our system both usable and secure, ideally, we would like to have FP and FN as close as 0 and recall, precision and F-measure as close as 1.

5.3.4.2 Classification Results

As mentioned in Section 5.3.3, we collected data from 18 users. From each user, we collected 50 samples of walking data. We divided the collected data into 18 sets based on the users' identities (ids). In order to build a classifier to authenticate a user based on her gait biometrics, we defined

two classes. The first class contains the walking data from a specific user, and the other class contains randomly selected walking data from other users.

The classification results are obtained after running a 10-fold cross validation, and are summarized in Table 5.3. The first part of Table 5.3 shows the results of using all the features extracted using sensors from the phone, the watch and both devices. We found combining the features from the phone and the watch sensors decreases the false negative from 5.8% in case of only phone, 8.5% in case of using only watch to 3.8% and decreases the false positive from 6.8% in case of only phone, 10.5% in case of using only watch to 4.2%.

The second part of Table 5.3 shows the results obtained by finding the sensor subset that provides the best overall average. We found that utilizing only accelerometer, gyroscope, magnetometer and orientation sensors from phone rather than using all phone sensors decreases the false negative and the false positive by around 2%. Similarly, using only accelerometer, gravity, gyroscope, linear acceleration and magnetometer sensors from watch instead of using all watch sensors decreases the false positive rate from 10.5% to 9.5% and the false negative rate from 8.5% to 8.0%. Furthermore, we found utilizing only phone accelerometer, phone gyroscope, phone magnetometer, phone orientation, watch accelerometer, watch magnetometer, and watch orientation sensors improves the classification accuracy (i.e., decrease both the false positive and the false negative rate by 1.2% and 1.6%, respectively). These features subset also contained the subset of features which were not correlated to each other. We leverage these uncorrelated features to prevent our *WUZIA* system against a sophisticated form of active impersonation attack [92], as we will describe in Section 5.3.5.2.

Finally, we checked the classification accuracy by selecting for each user the subset of sensors that provides the best results. The results of this model are shown in the last three rows of Table 5.3. We found out that the classifier performance improved over the previous two models. Moreover, both the average false positive and the average false negative rates dropped to around 0% when we used the best subset from both of the devices.

5.3.4.3 Summary of Results

In summary, the results obtained from the classification models show that the gait biometrics can be detected in a robust manner and thus will serve as an effective method for authenticating the

users. The results show that the fusion of the phone and the watch sensors significantly enhances the performance of detecting the gait biometrics. This is reflected in very low false positives and false negatives.

5.3.5 Active Adversarial Attack

5.3.5.1 Human Imposter Attack

In a human-based imposter attack, the adversary tries to manually mimic a victim's walking pattern so that it can fool the *WUZIA* system. Our model assumes that the attacker already has the physical possession of the $\mathcal{P}$ devices (phone and/or watch). Such kinds of attacks have been explored in the literature by a few researchers [75, 76, 91]. However, most of these works use accelerometer devices (e.g., MR100 wearable sensor) (not a phone or a watch used in our scheme), and these devices are worn on the waist tied to the belt [75, 91] or on the limbs near the shoes [76]. Therefore, we analyze how our system will perform when an attacker with similar physical characteristics attempts to learn and imitate an individual's walking pattern.

During the walking biometrics data collection, we recorded videos of eight different users. The attacker (a researcher, serving the role of an expert attacker) chose two of the users as victims (we call them V_1 and V_2) who exhibited the simplest walking pattern or distinctive visible characteristics, upon careful visual inspection. If the attacker can not succeed in attacking such simplistic walking patterns, then it would be harder for the attacker to succeed in attacking more complex walking patterns.

In our experiment, the attacker watched the video several times so as to learn the feet and the hand movement pattern of the user. While practicing, the attacker also tried to match the time duration from the start to the end of the victim's walk, using the video. After the attacker felt comfortable with the timing and the walking pattern, we collected the data for the attacker with the $\mathcal{P}$ devices walking towards the $\mathcal{V}$ device. The attacker was provided the visual feedback while imitating the walk pattern.

To measure the performance of the imposter in mimicking the victim, we first trained a random forest classifier with the victim's data using 10-fold cross validation. First, we trained the classifiers with the subset of features that provided the best average results, as mentioned in Section 5.3.4. We analyzed the classifier's accuracy with features from the phone only, the watch

TABLE 5.4: Performance for the imposter attack on two different victim users for two different types of classifier categories. The first three rows show the performance of the imposter attack against the classifier trained with the subset that provides best average results, as mentioned in Table 5.3. The last three rows show the result of the imposter attack against the classifier trained with the best subset for the individual victim user. Highlighted cells emphasize the most interesting results.

		Victim V_1		**Attacker**	**Victim V_2**		**Attacker**
		F-measure	**FPR**	**FPR**	**F-Measure**	**FPR**	**FPR**
General	**Phone Only**	0.931	0.100	0.000	0.936	0.100	0.917
	Watch Only	0.887	0.100	0.909	0.935	0.080	0.000
	Both	0.980	0.020	0.091	0.989	0.000	0.833
Individual	**Phone Only**	0.970	0.040	0.182	0.968	0.060	0.917
	Watch Only	0.960	0.060	0.000	0.968	0.060	0.000
	Both	1.000	0.000	0.091	1.000	0.000	0.000

only and both devices. We also trained our classifiers with the subset of features that provided the best performance for the individual user (victim). Then, we tested these classifiers against the imposter attacker's data to determine the success rate of the attacker. The results are shown in Table 5.4.

As expected, we found that the individual classifier performed better than the general classifier. When the general classifiers were tested against the imposter attacks, the attacker was able to imitate the hand motion (captured by watch) of V_1 (FPR = 0.909), while he could not imitate the hip motion (captured by phone) of V_1 (FPR = 0.000). On the other hand, the attacker was able to imitate the hip motion of V_2 (FPR = 0.917) while it could not imitate the hand motion of V_2 (FPR = 0.000). When both devices were used, we can see that the FPR for V_1 is low (0.091) but still high for V_2 (0.833). This suggests that the classifier trained with the features from both devices was dominated by the features from the phone, and hence the results of impersonation are more similar to that of the phone only. Similarly, when the individual best subset features were used to train the classifier, the attacker could not imitate the hand motion resulting low attack success rate when both devices' features were used. In other words, *WUZIA* could resist the imposters to a high degree when both devices' features and the best subset of features were used for each individual user.

In summary, these results show that the *WUZIA* system that leverages both phone and watch, and employs individualized classifiers can be highly resistant to walking imitation attacks. This is a significant security advantage of a multi-device *WUZIA* scheme.

5.3.5.2 Treadmill Attack

To perform a more powerful attack on the victim's walking pattern so as to successfully fool the *WUZIA* system, we followed the work by Kumar et al. [92]. This research represents the state-of-the-art attack against gait biometrics and is therefore an ideal platform to evaluate our system against. In this attack, the attacker already has the sample of a victim's gait pattern. First, the authors extract different features from the accelerometer sensor of the smartphone to authenticate users based on their walking pattern to create a baseline model called Gait Based Authentication System ($GBAS$). Then, they attack on the $GBAS$ system using a treadmill. In this attack, instead of imitating the victim's walking pattern, the attacker uses treadmill to control different gait characteristics (GCAT) such as speed, step length, step width and thigh lift to match the features extracted from the victim's walking pattern. To setup this attack, the attacker first analyzes the feature subsets that dominates the decision making process of the machine-learning classifiers [92]. Among these dominant features subset, the attacker then analyzes how these features are correlated with each other. From this analysis, the attacker tries to manipulate only one feature among the correlated features set. Now the attacker has final set of five features which it needs to manipulate to fool the classifier. The experimenter creates an imitator profile based on these final five features mapped to the four GCAT. This mapping is also created using correlation between GCAT and the dominating feature set. For example, if speed is directly correlated with the mean of X-axis of the accelerometer (ACC_{X_M}) then to increase or decrease the ACC_{X_M}, the imitator needs to increase or decrease the walking speed, respectively.

To thwart such attacks using sophisticated devices like treadmill to control different gait characteristics, we calculated the correlation values among each pair of features. The detail regarding the calculation of the correlation among features is explained in Appendix A.1 and the results are shown in Appendix Figure 1. From this analysis, we observe that the features from the phone are more correlated with the features from the phone while the features from the watch are more correlated with the features from the watch. This means that the attacker cannot use one device to alter the feature of the other device, however, it may be able to alter the features from a single device if it knows the correlation among the features from the same device.

We next analyzed how the features from a single device are correlated with the other features from the same device. The correlations among the features from the same device are depicted in

Appendix Figures 2 and 3. From these plots, we can see that the features extracted from a single sensor were more correlated to each other than the features extracted from different sensors. For example, mean, standard deviation and range of the accelerometer sensor were more correlated with each other, compared to those taken from gyroscope or magnetometer. We wrote a script to find out the best feature subset such that each feature is correlated to each other in a given feature subset by less than -/+ 0.1 (i.e., the subset of uncorrelated features). More the number of uncorrelated features in this subset, harder it will be for the attacker to correlate/match all the features with different gait characteristics [92]. Further one gait characteristics may influence more than one feature vector which do not have any correlation, increasing the difficulty of the treadmill attack.

Further, to increase the performance of the classifier in defending the treadmill attack, we wrote another script to find out the super set of the subset containing maximum number of uncorrelated features set. The best feature subset for the general classifier in Section 5.3.3 that is trained with features from both devices consists of eight uncorrelated features. This increased the accuracy of the classifier during the benign case while still being robust to the treadmill attackers. Further, the treadmill attackers may use more sophisticated devices to provide better gait characteristics that may alter different features. We can defend this by increasing the correlation threshold (currently set to 0.1) for finding uncorrelated feature set. This will provide larger number of features that are correlated to each other by that threshold value. Note that the correlation of 0 to 0.1 is considered near-zero correlation while that between 0.1 and 0.3 is considered weak correlation [148, 149]. Hence, using the correlation threshold of 0.3 will still give the feature subset with weak correlation that attacker may not be able to attack using the treadmill technique.

5.3.6 Summary

In this section, we proposed the use of walking-based biometrics to protect zero-interaction authentication systems in the event of loss or theft of authentication tokens. Our approach transparently authenticates the user to her authentication token as she walks towards the authentication terminal in order to unlock it. Our system leverages a smartphone and/or a smartwatch, and multiple embedded sensors therein, to reliably detect the unique walking pattern of the user. Our results suggest that especially when using both devices together, the system offers almost

error-free detection and makes it very difficult for even a powerful attacker to imitate a user's walking habit. Consequently, we believe that our approach can significantly enhance the security of current zero-interaction systems without degrading their usability.

5.4 Summarizing Context Enhanced Authentication

In this chapter, we presented different approaches to authenticate a user transparently during an NFC payment transaction and $\mathcal{Z}IA$ based authentication. The approach captures the user's movement and identifies the user based on the sensor data recorded by the device(s). The gestures are very unique to the user and are difficult for the attacker to mimic. We presented the design and implementation of the proposed authentication approaches. Our approach could be very effective in authenticating users and preventing misuse of services in case of theft or loss of the device, without calling for any additional user burden, significantly enhancing the security of such NFC payment systems as well as $\mathcal{Z}IA$ systems.

CHAPTER 6

CONTEXT ENHANCED CO-PRESENCE DETECTION

6.1 Introduction

In proximity-based "zero interaction authentication" (ZIA) [29] systems, a verifier device $\mathcal{V}$ authenticates the presence of a prover device $\mathcal{P}$ in physical proximity of the verifier while requiring *no additional interaction* by the user of the prover device. The zero interaction requirement is intended to improve usability of access control systems. For example, BlueProximity [138] allows a user to unlock the idle screen lock in her computer merely by physically approaching the computer while in possession of a mobile phone, previously paired with the computer, without having to perform any other action, such as typing in a password. Motivated by these usability considerations, there are many examples of ZIA systems, such as "Passive keyless entry and start" systems like "Keyless-Go"[1] PhoneAuth [108], and access control systems based on wearable devices [140].

Although the security research community no longer takes security and usability to be mutually contradictory goals [150], simultaneously accomplishing security and usability goals continues to be a challenge. Under the standard Dolev-Yao adversary model [52], an attacker is assumed to have complete control over the communication channel. In such a model, naïve ZIA schemes are vulnerable to *relay attacks* where a pair of colluding attackers relays messages between a legitimate user and verifier, thereby fooling the verifier into incorrectly concluding that the user is in close proximity. Relay attacks have been demonstrated to be practical for various short range wireless communication technologies like Bluetooth [26, 151], RFID [28] and NFC [99], making this vulnerability a serious threat.

The commonly proposed defense against such relay attacks, while preserving zero-interaction, is to use *distance bounding* techniques [96]. Distance bounding assumes that the prover and verifier share a security association. The prover is required to respond to a series of rapid-fire challenges from the verifier, which can then calculate a lower bound for the distance to the prover

[1] http://techcenter.mercedes-benz.com/_en/keylessgo/detail.html

by measuring the elapsed time between sending a challenge and receiving a correct response. Distance bounding needs to be implemented at the lowest possible layer in the communication stack because even a small error in estimating processing time at the prover side can lead to large deviations in the distance bound. Therefore implementing distance bounding on commodity devices like ordinary smartphones might be a challenge.

An alternative approach is to leverage the fact that two co-present devices will "see" (almost) the same ambient environment. Modern computing devices are equipped with many "sensors" like microphones, wireless networking interfaces, global positioning system (GPS) receivers, temperature, humidity and so on. A device can extract information from such a sensor that is characteristic of that context. By having two mutually trusting devices exchange and compare context information, they can determine if they are co-present or not. This approach has recently been proposed for *single* sensor modalities, including Wi-Fi [104, 105], audio [98, 103], Bluetooth and GPS [152].

Although these prior works constitute an important step towards addressing the hard problem of resisting relay attacks using off-the-shelf hardware, they leave several important questions unexplored, which we address in this chapter. *First*, we compare the performance of different sensor modalities in resisting relay attacks against ZIA based on contextual co-presence. Although standalone evaluations of different modalities individually have been reported in prior work, they cannot be used for a fair comparison given that the data assessing each modality was collected in disparate settings. *Second*, we investigate whether the combination ("fusion") of multiple sensor modalities will perform better than using individual modalities in isolation. Prior work did not address this question.

In this chapter, we use the ambient information to detect the co-presence of two devices and prevent relay attacks. This ambient information can either be acoustic or Radio Frequency (RF) signals [37, 38], such as Wi-Fi signal, Bluetooth signal, ambient audio or be naturally occurring [39], such as temperature, altitude, gas ratio, humidity. Both of these ambient information, which can be detected by various sensors either embedded in the smartphone or off-the-shelf sensing device, can be used to analyze the context of the surrounding environment and identify the co-presence or non-co-presence of two devices.

6.2 Background

In this section, we review the proximity-based authentication approach that forms the focus of this chapter and the underlying threat model, followed by an overview of our relay attack defense based on ambient multi-sensing.

6.2.1 Functional Model for Proximity-based Authentication

Figure 2.1 shows a general model of proximity-based authentication. The model consists of a prover $\mathcal{P}$ who wants to authenticate itself to verifier $\mathcal{V}$ and convince $\mathcal{V}$ that it is close to $\mathcal{P}$. The authentication process between $\mathcal{P}$ and $\mathcal{V}$ is typically run when they are in close proximity to each other. $\mathcal{V}$ makes use of a back-end "comparator" function to make the authentication decision (it could reside on the verifier device or on a remote machine such as a bank server in the case of payment transactions). $\mathcal{P}$ and $\mathcal{V}$ have pre-shared secret keys K and K', respectively, with the comparator. In an authentication session, $\mathcal{V}$ sends a *challenge* to $\mathcal{P}$ which computes a *response* based on the *challenge* and K. $\mathcal{P}$ returns the *response* to $\mathcal{V}$ which uses the comparator function to decide if *response* is acceptable.

This functional model is applicable to various real-world scenarios such as payment at a point-of-sale (POS) terminal and zero interaction authentication (ZIA) for access control to locking/unlocking a car or a desktop computer. In the payment scenario, the payment card plays the role of $\mathcal{P}$, and the POS terminal plays the role of $\mathcal{V}$. The issuer of the payment card plays the role of the comparator. In ZIA the user token (key or mobile phone) acts as $\mathcal{P}$ and the terminal (car or desktop computer) plays the role of $\mathcal{V}$. The comparator functionality is integrated in the terminal itself and therefore K' is not needed.

6.2.2 Threat Model

We assume a standard Dolev-Yao adversary model [52] where the adversary $\mathcal{A}$ has complete control over all communication channels. However, $\mathcal{A}$ is not able to compromise $\mathcal{P}$, $\mathcal{V}$ or the comparator, i.e., none of the legitimate entities have been compromised. The goal of $\mathcal{A}$ is to carry out relay attack by convincing $\mathcal{V}$ that the $\mathcal{P}$ is nearby when in fact $\mathcal{P}$ is far away. Figure 2.2 shows how $\mathcal{A}$, in the form of a relay-attack duo ($\mathcal{A}_p$,$\mathcal{A}_v$) can relay messages between the legitimate $\mathcal{P}$ and $\mathcal{V}$ with $\mathcal{A}_p$ and $\mathcal{A}_v$ acting as a dishonest prover and verifier, respectively.

6.2.3 Our Approach: Relay Attack Defense with Ambient Multi-Sensing

Figure 2.1 shows our countermeasure against relay attack which is based on the natural assumption that two entities will sense similar ambient environments when they are co-present. When $\mathcal{P}$ sends an authentication trigger to $\mathcal{V}$, they both start sensing their respective contexts using ambient physical sensor modalities, resulting in CP and CV, respectively, as the sensed data. This sensor data may be acquired using an additional (uncompromised) device, connected over a secure channel, to $\mathcal{P}$ and $\mathcal{V}$ (such as Sensordrone) or via the sensors embedded within $\mathcal{P}$ and $\mathcal{V}$. We consider RF sensor modalities, such as *Wi-Fi*, *Bluetooth*, *GPS*, as well as *audio* as sensors embedded within $\mathcal{P}$ while physical ambient sensor modalities, such as *temperature*, *precision gas*, *humidity* and *altitude* as sensors from an additional device (Sensordrone). $\mathcal{P}$ will attach CP to *response*. Similarly $\mathcal{V}$ will convey CV along with *challenge* in its message to the comparator. In case multiple sensors are used (say n), CP would be the vector $CP_1, CP_2, ..., CP_n$, and similarly, CV would be the vector $CV_1, CV_2, ..., CV_n$.

Using the keys K, K', the comparator can recover and validate CP and CV, and compare them (in addition checking that *response* matches *challenge*). We recall that in scenarios where the comparator is integrated with $\mathcal{V}$, K' is not used.

Figure 2.2 illustrates the presence of the relay attack duo $\mathcal{A}$ =($\mathcal{A}_p$,$\mathcal{A}_v$). Assuming that $\mathcal{A}$ cannot subvert the integrity of context sensing and the comparator can reliably tell the difference between co-presence and non co-presence by examining CP and CV, our countermeasure based on context sensing will thwart a Dolev-Yao $\mathcal{A}$. In the rest of this chapter, we describe our experiments to evaluate whether a comparator can reliably distinguish co-presence and non co-presence based on context information CP and CV sensed using RF, ambient audio and physical ambient sensors.

6.3 RF and Audio Sensors for Co-presence Detection

6.3.1 Data Collection

As mentioned before current smartphones not only provide different sensors such as Wi-Fi, Bluetooth, Audio, GPS but also sensors to measure ambient physical environment properties such as ambient temperature, relative humidity and relative pressure sensors. In this section, we

provide the co-presence detection and relay attack prevention based on the RF and audio sensors embedded on the current smartphone.

6.3.1.1 Sensor Data

For RF and audio, we currently use GPS, Wi-Fi, Bluetooth and audio modalities. These modalities were chosen as they are widely available on contemporary smartphones.

GPS Raw Data: We record the identifiers of visible GPS satellites and the "signal strength" for each of them in the form of signal-to-noise ratio (SNR). The identifier is the "pseudo-random noise code" (PRN) which is an integer ($1 \ldots 32$). The SNR ranges from 0 to 100.

Wi-Fi: For each visible Wi-Fi access point (AP), we record the list of link-layer addresses (BSSID) and the associated received signal strength indicators (RSSI), supported capabilities and the frequency of the Wi-Fi channel advertised by that AP. RSSI ranges from -100 to -20 dBm.

Bluetooth: For each visible Bluetooth device, we record the identifier (BDADDR) and received signal strength indicator (RSSI). RSSI ranges from -100 to -20 dBm.

Audio: Ambient audio is recorded in standard PCM format (wav file) without compression. Each PCM wave is sampled in 44100Hz with 16-bit encoding. Because raw audio is sensitive, by default, we do not store raw audio on Server. Instead, we extract certain features (as described in Section 6.3.2). Users however have the option of changing this default to let their client(s) upload raw audio to Server.

6.3.1.2 Dataset

We collected data for 15 days in mid 2013. Hardware variations across devices are well-known to cause significant changes in sensor measurements. To ensure robustness of results with respect to device variations, we collected data using tablets and phones from different manufacturers and with different models: tablets (Google Nexus 7, Samsung Galaxy Tab, Acer Iconia, Asus Transformer) and phones (Samsung Galaxy SII, SIII).

We gave no specific instructions to the testers about what scenarios or locations in which they should collect data. Consequently, the resulting dataset is *uncontrolled*, consisting of data collected in various everyday settings and locations (e.g., university campus, labs, libraries, cafeteria, home, streets), Data collection was done in two different cities: Birmingham, Alabama,

USA and Helsinki, Finland. This dataset contains 2303 samples, of which 1140 samples (49.5%) are from co-present devices and 1163 (50.5%) from non co-present devices. Each sample contains data from sensor modalities available at the time on the respective devices (2117 with audio, 1600 with Bluetooth, 782 with GPS and 2269 with Wi-Fi). For each sample, we scan all available sensors simultaneously: 2 minutes for GPS scanning, 10 scans for Wi-Fi (about 30 seconds), 10 seconds for recording ambient audio, and 10 scans for Bluetooth (up to 12 seconds for each scan).

6.3.2 Co-presence Detection

6.3.2.1 Features

We investigated various possible features that can be extracted from the data in different sensor modalities, finally settling on the most promising features as discussed below.

Features for Bluetooth, Wi-Fi, GPS: Although the three sensors (GPS, Bluetooth, Wi-Fi) involving radio-frequency (RF) emissions considered in our analysis are different, fundamentally they have the same inherent characteristics. We therefore chose to represent them by a common set of features. Let a record from an RF sensor modality be of the form (m, s) where m is an identifier of a sensed device and s is the associated signal strength. Let S_a and S_b denote the set of records sensed by a pair of bound devices A and B, and let n_a and n_b denote the number of different beacons (i.e., Wi-Fi access points, GPS satellites or Bluetooth devices) observed by devices a and b. We define the following sets:

$S_a = \{(m_i^{(a)}, s_i^{(a)}) \mid i \in \mathbb{Z}_{n_a-1}\}$.

$S_b = \{(m_i^{(b)}, s_i^{(b)}) \mid i \in \mathbb{Z}_{n_b-1}\}$.

$S_a^{(m)} = \{m \; \forall (m, s) \in S_a\}$, $S_b^{(m)} = \{m \; \forall (m, s) \in S_b\}$.

$S_\cap = \{(m, s^{(a)}, s^{(b)}) \; \forall m | (m, s^{(a)}) \in S_a, (m, s^{(b)}) \in S_b\}$.

$S_\cup = S_\cap \cup \{(m, s^{(a)}, \theta) \; \forall m | (m, s^{(a)}) \in S_a, m \notin S_b^{(m)}\}$

$\quad \cup \{(m, \theta, s^{(b)}) \; \forall m | (m, s^{(b)}) \in S_b, m \notin S_a^{(m)}\}$,

θ is modality-specific (see below).

$S_\cap^{(m)} = \{m \; \forall m | (m, s^{(a)}, s^{(b)}) \in S_\cap\}$.

$S_\cup^{(m)} = \{m \; \forall m | (m, s^{(a)}, s^{(b)}) \in S_\cup\}$.

$L_a^{(s)} = \{s^a | (m, s^{(a)}, s^{(b)}) \in S_\cap\}$.

$L_b^{(s)} = \{s^b | (m, s^{(a)}, s^{(b)}) \in S_\cap\}$.

$S_\cap$ consists of devices seen by both A and B; $S_\cup$ represents all devices seen by A or B with θ filled in as the "signal strength" for devices that are *not* seen by either device.

We consider a total of six features, five of which have been selected from state-of-the-art co-presence detection systems (NearMe [104], Amigo [105] and RF-based place learning schemes [153].

1. Jaccard distance: $1 - \frac{|S_\cap^{(m)}|}{|S_\cup^{(m)}|}$
2. Mean of Hamming distance: $\frac{\sum_{k=1}^{|S_\cup|} |s_k^{(a)} - s_k^{(b)}|}{|S_\cup|}$
3. Euclidean distance: $\sqrt{\sum_{k=1}^{|S_\cup|} (s_k^{(a)} - s_k^{(b)})^2}$
4. Mean exponential of difference: $\frac{\sum_{k=1}^{|S_\cup|} e^{|s_k^{(a)} - s_k^{(b)}|}}{|S_\cup|}$
5. Sum of squared of ranks: $\sum_{k=1}^{|S_\cap|} (r_k^{(a)} - r_k^{(b)})^2$

   where, $r_k^{(a)}$ (respectively $r_k^{(b)}$) is the rank of $s_k^{(a)}$ ($s_k^{(b)}$) in the set L_a (L_b) sorted in ascending order.

6. Subset count: $\sum_{i=1}^{T} f_i$. Here T is the scanning time (seconds)

   $f_i = 1$ if $S_{a_i}^{(m)} \neq \emptyset$, $S_{b_i}^{(m)} \neq \emptyset$,
   $(S_{a_i}^{(m)} \subseteq S_{b_i}^{(m)}$ *or* $S_{a_i}^{(m)} \supseteq S_{b_i}^{(m)})$

   $f_i = 0$ otherwise. S_{a_i}, S_{b_i} are the set of records by A and B respectively at the i^{th} second

Wi-Fi: Features 1-5 are used. Since we do multiple scans in each sample, in line with current best practices, we use the mean value of RSSI for a BSSID (m) from all of the scans as the signal strength (s) value. θ is -100.

Bluetooth: Features 1,3 are used with BDADDR as identifier (m) and average RSSI as signal strength (s). θ is -100.

GPS: All features are used with PRN as identifier (m) and mean SNR as signal strength (s). θ is 0.

Note that feature 6 is used only for GPS. This is because the set of satellites visible to a device varies greatly depending on the sensitivity of GPS hardware. Thus, one device may see a

subset of the satellites seen by the another co-present device. In such cases, metrics like Jaccard distance perform poorly whereas the subset count could perform better. When GPS co-ordinates are available for A and B in a sample, we also use the orthodromic distance [154] as a feature.

Features for Audio: We consider two features proposed by Halevi et al. [98], which were found to be the most robust among all algorithms tested: Schurmann and Sigg [103], SoundSense [155], Shazam audio fingerprinting [156], and Sound of Silence [157]. The other features either required careful synchronization between the two audio samples or were highly sensitive to variations in the microphone characteristics of the devices. The two features that we consider are defined as follows:

- Max cross correlation:
  $M_{corr}(a,b) = Max(cross\ correlation(X_a, X_b))$
- Time frequency distance:
  $D(a,b) = \sqrt{(D_{c,time}(a,b))^2 + (D_{d,freq}(a,b))^2}$
  where, $D_{c,time}(a,b) = 1 - M_{corr}$, $D_{d,freq}(a,b) = ||FFT(X_a) - FFT(X_b)||$ is the Euclidean norm of the distance.

Here X_a and X_b denote the raw (16-bit PCM) audio signals recorded by A and B and FFT(X_a), FFT(X_a) denotes the Fast Fourier Transforms of the corresponding signals.

6.3.2.2 Analysis and Results

TABLE 6.1: Overall performance vs. time budget

Time Budget (s)	5	8	10	12	15
%FN	8.95	2.19	1.67	1.40	1.49
%FP	7.14	2.67	1.98	2.15	2.15
MCC	0.841	0.951	0.966	0.964	0.964
Fm	0.921	0.976	0.983	0.982	0.982

We use machine-learning classifiers to detect the contextual co-presence. We performed classification using ten-fold cross-validation and Multiboost [158], a state-of-the-art algorithm widely used for different types of context recognition tasks, as the classification algorithm. In all experiments, decisions trees (J48 Graft) are used as the weak learners. From each experiment, we

record the 2x2 confusion matrix, containing the number of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). Positive and negative classes represent co-presence and non-co-presence, respectively.

For the evaluation, as discussed in Section 4.3, we used False Positive Rate (FPR), False Negative Rate (FNR), $precision$, $recall$ and $F\text{-}measure$ (Fm) as performance measure of the classifiers as defined in Equations 4.1 and 4.2. Along with these, we also used Mathhews' correlation coefficient (MCC) to evaluate the classifier's performance. MCC is an approximate statistical measure for deciding whether the prediction is significantly more correlated with the data than a random guess. It can be calculated as shown in Equation 6.1.

$$|MCC| = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \tag{6.1}$$

The performance of our classifier is shown in Table 6.1. In this table, we show here how the classifiers would perform w.r.t time. The performance improves significantly when we increase the time budget from 5 seconds to 15 seconds, however, increasing scan time means more delay, and hence decreases usability.

In this work, we also analyzed how the classifier performance will change when we fused different sensor modalities as shown in Table 6.2. Among individual modalities Wi-Fi performs best (Fm = 0.989, MCC = 0.978) and GPS worst (Fm = 0.776, MCC = 0.550). Bluetooth and Audio exhibit similar performance with the former (Fm = 0.885, MCC = 0.773) slightly better than the later (Fm = 0.857, MCC = 0.715).

6.4 Physical Sensors for Co-presence Detection

In this section, we explore purely ambient physical sensing capabilities present on upcoming devices to address the problem of relay attacks in authentication systems. More specifically, we consider the use of four new sensor modalities, *ambient temperature*, *precision gas*, *humidity*, and *altitude*, for $\mathcal{P}$-$\mathcal{V}$ proximity detection. Using an off-the-shelf ambient sensing platform, called Sensordrone[2], connected to Android devices, we show that combining these different modalities provides a robust proximity detection mechanism, yielding very low false positives (security against relay attacks) and very low false negatives (good usability). Such use of multiple

[2] http://www.sensorcon.com/sensordrone/

TABLE 6.2: Individual modalities vs Fusion of modalities; (A) Audio, (B) Bluetooth, (G) GPS, (W) Wi-Fi

	All samples containing Audio (sample size = 2117)							
	A only	A+B	A+G	A+W	A+B+G	A+B+W	A+G+W	A+B+G+W
FN(%)	19.9	12.49	20.41	1.52	12.59	1.52	1.73	1.62
FP(%)	9.28	5.21	7.07	1.59	4.33	1.77	1.59	1.77
MCC	0.715	0.829	0.736	0.969	0.837	0.967	0.967	0.966
Fm	0.857	0.914	0.866	0.984	0.918	0.983	0.983	0.983
	All samples containing Bluetooth (sample size = 1600)							
	B only	B+A	B+G	B+W	B+A+G	B+A+W	B+G+W	B+A+G+W
FN(%)	15.54	7.64	18.25	0.74	6.78	0.49	0.49	0.37
FP(%)	7.35	3.55	4.18	1.27	2.66	1.01	1.14	1.01
MCC	0.773	0.888	0.782	0.980	0.906	0.985	0.984	0.986
Fm	0.885	0.944	0.886	0.990	0.952	0.992	0.992	0.993
	All samples containing GPS (sample size = 782)							
	G only	G+A	G+B	G+W	G+A+B	G+A+W	G+B+W	G+A+B+W
FN(%)	23.6	14.89	25.28	1.97	18.54	1.69	2.53	1.97
FP(%)	21.36	14.32	13.85	3.52	12.91	3.99	3.52	3.76
MCC	0.55	0.707	0.615	0.944	0.688	0.941	0.938	0.941
Fm	0.776	0.854	0.808	0.972	0.845	0.971	0.969	0.971
	All samples containing Wi-Fi (sample size = 2269)							
	W only	W+A	W+B	W+G	W+A+B	W+A+G	W+B+G	W+A+B+G
FN(%)	0.36	0.27	0.45	0.45	0.18	0.18	0.27	0.18
FP(%)	1.83	1.83	1.83	1.83	1.83	1.83	1.92	1.83
MCC	0.978	0.979	0.977	0.977	0.980	0.980	0.978	0.980
Fm	0.989	0.989	0.989	0.989	0.990	0.990	0.989	0.990

ambient sensor modalities offers unique security advantages over traditional sensors (Wi-Fi, GPS, Bluetooth or Audio) because it requires the attacker to simultaneously manipulate the multiple characteristics of the physical environment. These ambient sensors also yield rapid response times and very low battery consumption, whereas traditional sensors can have noticeable scanning times and battery drainage. These ambient sensors may also be seamlessly combined to work with traditional sensors to further improve security.

To demonstrate the feasibility of our approach, we use an additional environmental sensing platform (Sensordrone). However, the devices participating in the protocol themselves ($\mathcal{P}$ and $\mathcal{V}$) may be equipped with various environmental sensors in the future [159, 160]. Android platform already supports broad category of environmental sensors that includes barometer, photometer and thermometer [161] such that phones and other devices that come equipped with these sensors will already have an interface to provide data to corresponding application.

6.4.1 Data Collection

6.4.1.1 Sensor Modalities

We explore the use of various ambient sensor modalities to determine whether two devices are co-present or not. In this section, we are focusing on ambient temperature, precision gas, humidity and altitude, and combinations thereof, which are readily provided by Sensordrone (see Figure 6.1). In this section, we describe the functioning details of these sensors.

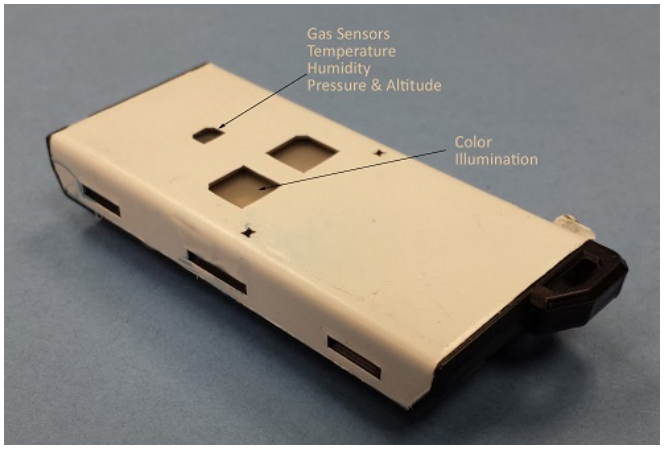


FIGURE 6.1: Sensordrone device with different sensors (ambient temperature, precision gas, humidity and altitude are utilized in this work). Device dimensions: 2.67 x 1.10 x 0.49 inch^3.

Ambient temperature: It is the temperature in a given localized surrounding. Ambient temperature of different locations might be different as it changes with sensor being indoors or outdoors, and differs from one room to another with Air Conditioning adjusted at different levels. We recorded the current temperature, in Celsius scale, at different locations. Sensordrone uses silicon bandgap sensor to record the ambient temperature. The principle of the bandgap sensor is that the forward voltage of a silicon diode is temperature-dependent [162].

Humidity: It is the amount of moisture in the air which is used to indicate the likelihood of precipitation or fog. Humidity can serve as the contextual information about the location since the amount of water vapor present in the environment may differ when moving from one location to the other. Capacitive Polymeric Sensor is used to detect the humidity of the surrounding. It consists of a substrate (glass, ceramic or silicon) on which a thin film of polymer or metal oxide is deposited between two conductive electrodes. The change in the dielectric constant

of a capacitive humidity sensor is nearly directly proportional to the relative humidity of the surrounding environment [163].

Precision Gas: Ambient air consists of various gases, primarily Nitrogen and Oxygen. The gaseous content of a particular location may differ from that of another location. The Sensordrone device comes with pre-calibrated Carbon Monoxide (CO) sensor, which measures the CO content of the atmosphere. We used the default calibration of the device that monitors CO to get the context information of the location. The values were measured in "ppm (parts per million)".

Altitude and Pressure: Atmospheric Pressure of a particular location is the pressure caused by the weight of air at that location above the measurement point. With increase or decrease in elevation, the weight of air above the location changes and so does the pressure at that location. Although the variation of pressure can be obtained from the altitude, it changes drastically with the weather. Hence, pressure at a location can serve as an indicator for that location and time. In our experiments, the pressure was recorded in "mmHg (millimeter of Mercury)" using Micro electromechanical (MEMS) Pressure Sensor. When there is a change in pressure from the air on a diaphragm within the sensor, the piezoresistive sensors senses the change with alternating piezoelectric current which is used to determine the actual pressure.

This is also used to determine the altitude. Since the pressure value at any given location is directly proportional to the amount of gases above the device and the amount of gases above the device is inversely proportional to the altitude, the altitude value can be derived from the pressure sensor using the equations 6.2. The units for station pressure must be converted to millibars (mb) or hectopascals (hPa) before using following expression to convert the pressure values into altitude [164].

$$h_{altitude} = \{1 - (\frac{P_{station}}{1013.25})^{0.190284}\} * 145366.45 \tag{6.2}$$

The $h_{altitude}$ measurements are in feet, and are multiplied by 0.3048 to convert them to meters.

Although Sensordrone provides both pressure and altitude readings, we only use altitude to classify the location as altitude is derived from pressure. We found that as the readings are taken at a more precise scale, the classifiers result improves. In our dataset, we measured pressure in *mmHg* and altitude in *m*. The pressure values did not vary much and were not very useful

in providing accuracy to the classifier while altitude provided a clear difference between two locations allowing classifier to more accurately make predictions.

Excluded Sensors: Although there are other sensors available on the Sensordrone device, we did not use the data from those sensors for two reasons: either they did not convey information about the ambient context or may not work when blocked. The sensors excluded from our experiments are as follows.

- *Object Temperature:* This sensor uses Infrared to obtain the temperature of a nearby object (line of sight object temperature). This measures the information about a specific object but not about the ambient environment.
- *Illuminance (Light):* It measures the ambient light luminosity and may seem like a useful modality to convey the environmental information. However, its use suffers from the fact that light intensity greatly varies depending upon the position of the source of light and the light sensor facing towards it. Also, the devices will not provide light measurements when their sensors are blocked, such as when the devices are stowed inside purses or backpacks.
- *Proximity Capacitance and External Voltage:* The proximity capacitance sensor measures changes in capacitive flux and is basically used for touch sensing or proximity detection like when used on touch pads or capacitive touch screens. The external voltage sensor gives the measure of a battery voltage level.

6.4.1.2 Dataset

The main goal here is to identify if two devices are co-present or not using the sensor data. We collect the data from two devices and use a classifier to determine if these devices are at the same location or at different locations. For this, we needed to collect the sensor data when the devices are in close physical proximity as well as when they are at different locations.

To collect the sensor data described in Section 6.4.1.1, we modified the original app provided in [165] to *record* the data to a file for further analysis (UI is shown in Figure 6.3). The data from all the sensors used in our experiments (ambient temperature, precision gas, humidity, and altitude) was recorded and labeled according to the location and time of the place. The data was also marked how the device was held, i.e., either in hand or in pocket (although this information

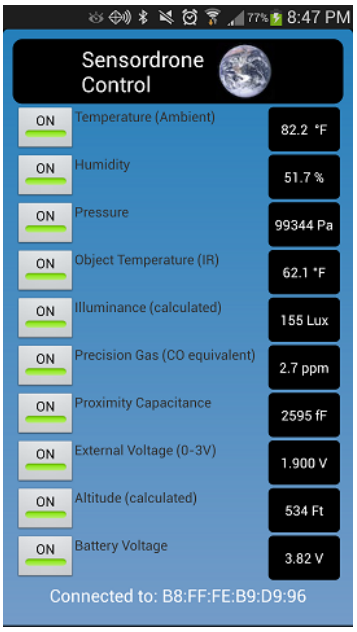


FIGURE 6.2: Original Sensordrone app displaying sensor values

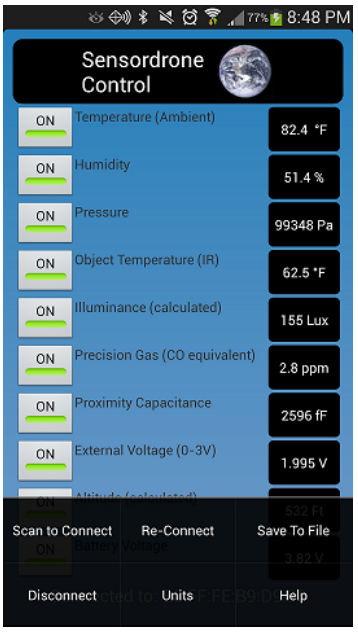


FIGURE 6.3: Modified Sensordrone app to record the sensor values

was not used in our current experiments; it can be useful when working with the light sensor in the future). The experiment was conducted in a variety of places, not just confined to labs and typical university offices. The locations included: parking lots, office premises, restaurants, chemistry labs, libraries as well as halls with live performance and driving on interstate highways. We collected a total of 207 samples at 21 different locations. The different samples collected from the same place are "paired" to generate co-presence data instances whereas those from different places are paired to generate non-co-presence data instances. We ended up with 21320 instances of which 20134 instances belonging to non co-presence class and 1186 instances belonging to co-presence class.

6.4.2 Co-presence Detection

6.4.2.1 Features

Let L_i and L_j be a sensor reading captured by two devices at locations i and j. The Hamming distance is calculated as follows:

$$D(i,j) = |L_i - L_j| \tag{6.3}$$

Given a sensor modality k (k is in range of (1, n) where n is the number of sensor modalities)

TABLE 6.3: Classification results for different combinations of environmental sensors

	FNR(%)	FPR(%)	Precision	Recall	Fm
			Single sensor modality		
Temperature (T)	23.74	32.40	0.705	0.763	0.733
Precision Gas (G)	15.26	30.36	0.739	0.847	0.790
Humidity (H)	16.25	29.81	0.740	0.838	0.786
Altitude (A)	8.57	16.25	0.851	0.914	0.881
			Combination of multiple sensor modalities		
HA	7.93	9.85	0.905	0.921	0.913
HGA	5.30	6.83	0.934	0.947	0.940
THGA	2.96	5.81	0.944	0.970	0.957

and $L_i^{(k)}$ and $L_j^{(k)}$ from two samples, we have $D^{(k)}(i,j) = |L_i^{(k)} - L_j^{(k)}|$. With the data corresponding to n modalities, we obtain a feature vector of n elements of $D^{(k)}(i,j) \mid 1 \leq k \leq n$.

We consider co-presence detection as a classification task and carry out our investigation using the Weka data mining tool [166]. All experiments have been performed using ten-fold cross validation and Multiboost [158] as the classifier. We choose Random Forest [167] as the weak learners in all experiments since it performs best among different base learners we have tried with our dataset (e.g., Simple Logistics, J48, and Random Forest). From each experiment, we record the 2x2 confusion matrix, containing the number of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). We denote co-presence class to be the positive class, and non co-presence to be the negative class.

We use the *F-measure* (Fm), false negative rate (FNR), and false positive rate (FPR) to measure the overall classification performance (as shown in Equations 4.1 and 4.2).

Classifiers produce reliable results when the data is balanced over all classes. Our dataset is highly biased towards the non co-presence class which is 17 times larger than the co-presence class. Therefore, we generate balanced data for classification by randomly partitioning the non co-presence class into 17 subsets. Each such subset together with the co-presence class constitutes a resampled set for classification. We run experiments with 10 resampled sets, chosen randomly.

Each of the different sensors alone may not be fully effective for the purpose of co-presence detection, and therefore, we also explore whether combinations of different sensors improve the classification accuracy. To analyze which combination provides the best result, we would

need to analyze all 15 different combinations of four different sensors. However, to reduce the underlying computations, we first analyze the accuracy provided by each individual sensor. Then we combine best two modalities and view how the accuracy of the classifier changes. We keep on adding the modalities to see the change in the accuracy until all the modalities are fed into the classifier for co-presence detection.

6.4.2.2 Analysis and Results

The results of experiments for different combinations of modalities are provided in Table 6.3. They suggest that, although each individual modality on its own does not perform sufficiently well for the purpose of co-presence detection, combinations of modalities, especially combining all the modalities together, is quite effective, with very low FNR and FPR, and high overall Fm. Altitude performs the best in classifying single modality, and also ranked the best by Chi-squared attribute evaluation but still has unacceptable FNR and FPR ($FNR = 8.57\%$, $FPR = 16.25\%$, $Fm = 0.881$) for our targeted applications demanding high usability and high security. The result of the combination of all modalities is clearly the best ($FNR = 2.96\%$, $FPR = 5.81\%$, $Fm = 0.957$. The intermediary combinations of different modalities used in experiments are also based on the ranks of each modality (evaluated by Chi-squared test). The results for the best combinations, Humidity-Altitude and Humidity-Gas-Altitude, are also presented in Table 6.3.

6.5 Summarizing Co-presence Detection

In this Chapter, we developed a co-presence detection approach based on information collected from multiple different environmental sensors either embedded on the smartphones or additional device connected to the smartphones. Further, we addressed the issue of using different sensor modalities for co-presence detection to be used in applications that need ZIA. This approach is geared for preventing relay attacks, a significant threat to many proximity-based authentication systems without significantly degrading its usability. Although individually RF and Audio sensors seems to be strong candidate to be chosen for the proximity detection, combination of physical sensors also formed a robust relay attack defense. While each individual physical sensor does not seem sufficient for the security and usability requirements of the targeted applications. The

key advantages of using physical sensors compared to RF sensors are: security (manipulating multiple environmental attributes simultaneously could be a challenging task for the attacker), efficiency (fast response time and negligible power drainage), and privacy (user-specific sensitive information may not be leaked or may be hard to infer).

CHAPTER 7

ATTACKS ON CONTEXT ENHANCED SYSTEM AND STRONGER MODELS

7.1 Introduction

Most of the approaches relying on the context assume that the context cannot be manipulated in one way or another. In this chapter, we assess whether this assumption is meaningful. In case of our contextual malware defense, we assumed that the kernel is healthy and the malware cannot manipulate the sensor data. We implement a malware *SMASheD* as proposed by [34] with strong capability such that it can, at the systems-level, manipulate the on-board sensor embedded on the device. We critically analyze how such assumption can be exploited. Further, we show how these malware must execute in order to break the security provided by the system relying on the context detection.

In our threat model in Section 6.2.2 [39], we assumed that it might be very difficult to manipulate the environment without getting noticed. For example, to change the temperature/altitude of the surroundings, the attacker may need tamper with the thermostat in the given locality, which may not be easy. However, the environmental sensor modalities can be vulnerable to manipulation as the attacker does not require manipulating the entire surrounding but just the small local space near the sensing device. In this chapter, we show that such manipulating attacks are possible in many $\mathcal{ZIA}$ systems as $\mathcal{V}$ device is mostly left unattended, such as a car parked in underground/remote parking or a laptop during lunch time. We formally examine the feasibility of these attacks and assess the security of our defense mechanisms under such attacks. In addition, we investigate context manipulation attacks against RF modalities as well as audio. We will then come up with defensive approaches and insights to counter such advanced adversaries.

Finally, we present an attack on *Sound-Proof*, two factor authentication system which uses ambient audio as the second factor authentication. From our analysis on contextual system, we successfully attack *Sound-Proof* by restricting ourselves within the threat model of *Sound-Proof*.

7.2 Insider Attack: *SMASheD*

Sensing-based security and non-security applications therefore crucially rely upon the sanity of the Android sensor security model. In this section, we show that such a model can be effectively circumvented. Specifically, we implement *SMASheD* [34] to stealthily sniff as well as manipulate many of the Android's restricted sensors. As mentioned in Section 2.1, *SMASheD* exploits the Android Debug Bridge (ADB) functionality and enables a malicious app with only the INTERNET permission to read, and write to, multiple different sensor data files at will.

Following the design provided in [34], we implement the *SMASheD* framework which encompasses three components: *SMASheD* server: a native service that provides the sensor data reading and injection capabilities, scripts: two simple scripts used to copy the *SMASheD* server to the device and to start the server, and *SMASheD* app: an app that runs a status detection module in the background, and depending on the phone's status and the desired functionality, it sends requests to the *SMASheD* server to read or inject sensor events.

7.2.1 *SMASheD* Attacks

In this section, we show how we can use *SMASheD* to attack our context enhanced authorization and authentication systems described in Chapters 4 and 5.

7.2.1.1 Attacking Authorization Systems

In Chapter 4, we present "Tap-Wave-Rub" [35, 40] and "WaveToAccess" [112]. We propose multiple gestures that can be used for the purpose of authorization. An implicit gesture, such as tapping the phone with another device (*tap*), is used to provide NFC permission to the requesting app. The system uses accelerometer sensor to detect the *tap* gesture. An explicit gesture, such as waving a hand in front of the phone (*wave*) or rubbing a finger near the proximity sensor (*rub*), is used to grant permissions for the services where no implicit gesture can be used. To detect *wave* and *rub* gestures, the system uses proximity sensor. Later, we also detected *wave* using light and accelerometer sensors; the light sensors to infer the fluctuation in light due to hand waving and the accelerometer sensor to reduce the possibility of detecting other events as hand wave. Both Tap-Wave-Rub and WaveToAccess assume that the kernel is immune and the sensor data cannot be manipulated by the malware.

SMASheD attacks the assumption made by our systems. To generate the *tap*, *wave* or *rub* gesture, the attacker can record his own gesture and later inject the recorded values via *SMASheD*. Alternatively, *SMASheD* can record the gesture provided by the user during the benign case and replay it later. A simpler attack can be performed on *wave* and *rub* gestures in Tap-Wave-Rub, in which *SMASheD* fluctuates the proximity sensor in quick succession so that the system infers the corresponding gesture.

To test the validity of our attack, as a proof of concept, we implemented the algorithm used to detect Tap-Wave-Rub's *wave* and *rub* gestures using our implementation described in Chapter 4. *SMASheD* system detects the *wave* and *rub* gestures, when the proximity sensor changes for certain number of times (6 times) within certain period (1.5 seconds). In our attack, we recorded the valid *wave* and *rub* gestures and replayed them. *SMASheD* was able to deceive the system successfully. A demo is available at `https://androidsmashed.wordpress.com/demos/`.

In Section 4.3, we also present a similar defense to mobile malware using *transparent human gestures* [36]. Our system uses the hand movement gesture to prevent unauthorized access of the services such as phone calling, picture snapping and NFC tapping. It looks for multiple, motion, position and environmental, sensor data to detect the calling, snapping and tapping gestures. The assumption that the system makes is the device is already infected with malware. However, the device kernel is healthy and is immune to the malware infection, and also that the malware is not capable of manipulating the sensors.

SMASheD can attack the assumptions made by this systems as well. The attacker can record the sensor data that is being used by these systems to detect the gesture during call, snap or tap. Now, when malware is trying to make a call, snap photo or tap NFC tag, *SMASheD* can replay all these sensor data fooling the system to believe that the user is performing the activity.

7.2.1.2 Attacking Authentication Systems

In Chapter 5, we show that our system can be used to identify using the sensors provided by Android. We present "NFC Tap Authentication" (Section 5.2) and "*WUZIA*" (Section 5.3) to authenticate users. We assume that to authenticate users using our approach, the device has not been compromised and that the device kernel is healthy such that sensors data or flow cannot be manipulated by the adversary.

SMASheD can attack the assumptions made by this systems as well. The attacker can record the sensor data that is being used by these systems to detect the implicit gesture. Once it learns all the sensor values during one genuine authentication, it can replay the same sensor values fooling the system to authenticate the users even when there is no real physical motion.

7.2.2 Summary

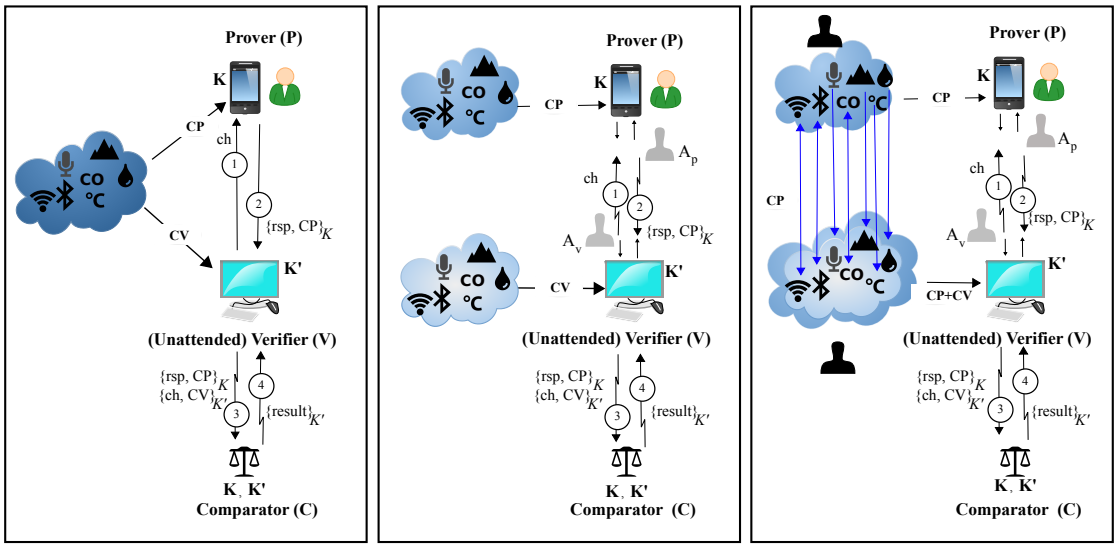
In this section, we called the Android's sensor security model into question. We exploited Android's ADB workaround using *SMASheD* that can effectively sniff and manipulate many sensors currently protected by Android's access control model. We called our security system into question by challenging the assumption made. The strong attack model as such introduces a wide spectrum of potentially devastating attacks that can compromise user privacy and subvert many security and non-security applications that rely upon different sensors like ours. We advocate the importance of raising people's awareness of the possible security risks associated with installing services through the ADB shell.

7.3 Outsider Attack: Environment Manipulation

In Chapter 6, we showed that different sensors embedded on the mobile device as well as additional sensor device paired with mobile device can be used for the co-presence detection of two devices. The presence of ubiquitous and low-cost sensing capabilities on many modern mobile devices has further facilitated a potentially more viable relay attack defense [98, 99, 105, 106]. One of the basic assumptions made in these works is that it is very hard to manipulate the contextual environment (i.e., it considered only a Dolev-Yao attacker [52]).

In this section, we are extending this model to the realm of a context-manipulating attacker. We show that it is feasible to manipulate the readings of different sensors (and combinations thereof) using low-cost, off-the-shelf equipment, representing a realistic attacker. We demonstrate attacks against a variety of modalities studied in prior work including audio, radio (Bluetooth/Wi-Fi), and physical (temperature, humidity, gas and altitude).

Based on the above manipulation capabilities, we comprehensively examine and quantify a zero-modality attacker, who manipulates one sensor only, as well as a multi-modality attacker, who can manipulate multiple sensor modalities simultaneously.



(a) P and V see the same environment

(b) Dolev-Yao attackers

(c) (Uni/bi)-directional single-modality and multiple-modality attackers

FIGURE 7.1: System model of proximity based authentication with contextual co-presence.

For systems that use multiple modalities, we investigate two different sensor fusion approaches – *features-fusion* (proposed in [38]) and *decisions-fusion* based on majority voting (equal voting and weighted voting), and show that both approaches are vulnerable to contextual attacks but the latter can be more resistant in some cases, at the cost of slight degradation in usability.

7.3.1 Environment Manipulation: Background and Threat Model

7.3.1.1 Overview

The goal of the Adversary $\mathcal{A}$ against a challenge-response authentication system is to fool Verifier $\mathcal{V}$ into concluding that Prover $\mathcal{P}$ is nearby and thus needs access to $\mathcal{V}$ even when $\mathcal{P}$ is actually far away. As mentioned in Section 2.2.2.2, the attacker possesses standard Dolev-Yao capabilities [52]: it has complete control of the communication channel over which the authentication protocol between $\mathcal{P}$ and $\mathcal{V}$ is run but does not have physical possession of $\mathcal{P}$ nor is able to compromise (e.g., through malware) either $\mathcal{P}$ or $\mathcal{V}$. The attacker could take the form of a "ghost-and-leech" [26] duo (A_p, A_v) such that A_p (respectively A_v) is physically close to $\mathcal{P}$ ($\mathcal{V}$), and A_p and A_v communicate over a high-speed connection as discussed in Section 2.2.2.2.

In Chapter 6, co-presence detection schemes aim to address such relay attacks. Figure 7.1(a)

shows a typical system model of an authentication/authorization protocol using contextual co-presence, adapted from Chapter 6. Figure 7.1(b) shows how contextual co-presence can thwart a Dolev-Yao relay attacker as described in Chapter 6. Figures 7.1(a) and (b) are comparable to Figures 2.1 and 2.2.

Prior work has proposed the use of different sensor modalities for such co-presence detection: ambient audio – Au [98], and radio context including Wi-Fi – W. In chapter 6, we used Bluetooth – B [38], and physical environmental attributes, temperature – T, humidity – H, concentration of gases – G and altitude – Al [39].

7.3.1.2 Threat Model for a Contextual Attacker

Our focus is on a context-manipulating attacker against co-presence detection (going beyond a Dolev-Yao attacker). In our threat model, an attacker cannot compromise $\mathcal{P}$ and $\mathcal{V}$ devices. However, based on the rationale that $\mathcal{V}$ is often unattended, whereas $\mathcal{P}$ is in the possession of a human user, we assume that the context attacker can manipulate context without detection only in one direction. More precisely, a contextual attacker is modeled as follows:

- A_p, A_v can measure the context information that $\mathcal{P}$, $\mathcal{V}$ would sense, respectively.
- A_v can fool $\mathcal{V}$ into sensing the context information A_v chooses. Specifically A_v can receive context information from A_p and reproduce it near $\mathcal{V}$.
- A_v (A_p) cannot suppress any contextual information from being sensed by $\mathcal{V}$ ($\mathcal{P}$).

Figure 7.1(c) illustrates this threat model. Later in Section 7.3.3, based on our context-manipulation attacks presented in Section 7.3.2, this current model will be extended, to incorporate multi-modality attackers, who can perform the above (single-modality) tasks corresponding to multiple modalities simultaneously.

7.3.2 Environment Manipulation: Attacks

In this section, we present our context manipulation attacks against audio, radio and physical sensor modalities, and their various combinations. "Modality" refers to the raw input used by the sensors [168, 169].

7.3.2.1 Manipulating Audio Sensor Modality

To manipulate ambient audio, an adversary must find a way to make ambient audio on one side similar to that on the other side. Recall from Section 7.3.1 that our threat model allows the attacker to add to the ambient audio at $\mathcal{V}$'s side without being noticed, allowing him to relay/stream the ambient audio in real-time from $\mathcal{P}$'s side to $\mathcal{V}$'s side thereby causing the features used for audio correlation almost match at both sides. The assumption that manipulating audio at $\mathcal{V}$'s side can go undetected is valid since $\mathcal{V}$ may be unattended in many scenarios (as our model in Section 7.3.1 assumed). The attacker duo can use any reliable audio streaming tool to stream the audio from $\mathcal{P}$'s side to $\mathcal{V}$'s side. They can execute this attack conveniently using mobile phones and wireless data connection. We evaluated how well such an attacker can succeed in fooling audio-based co-presence detection by streaming ambient audio using Skype. We use the features and classifier described in prior work [98]. Our results are presented in Section 7.3.3.2.

7.3.2.2 Manipulating Radio-Frequency Sensor Modalities

Prior work suggests that manipulating the radio context is possible in general. The work presented in [170] describes attacks on a public Wi-Fi based positioning system. They used a Linux laptop as an Access Point (AP) with the Scapy packet manipulation program [171] to spoof Wi-Fi APs. Similarly, spoofing Bluetooth device addresses has already been demonstrated in prior work [38, 151], both of which reported Bluetooth-based relay attacks. An attacker can control the received signal strength by controlling the transmission power of his masquerading devices. Therefore, we conclude that our threat model 2.2.2 is reasonable. Furthermore, in the case of Radio Frequency (RF) sensor modalities, it is reasonable to assume that an attacker can also manipulate the RF environment at $\mathcal{P}$'s end without being noticed (since radio waves are imperceptible to human users). Therefore, limiting the attacker to unidirectional manipulation only is too restrictive.

We tested the feasibility of Wi-Fi spoofing ourselves, and studied how it can be used to match the Wi-Fi context at two ends. In our experiment, we used a Linksys router (WRT54G) to create a spoofed hotspot. We flashed DD-WRT firmware [172] to the router since the default firmware did not allow us to spoof the Basic Service Set Identifier (BSSID). The router used in

our experiment is portable, easily available in the market, and much cheaper than other devices which can also be used to spoof the hotspot such as laptops or smartphones.

The DD-WRT control panel also provides an option to change the transmission power with which we can increase/decrease the signal strength. The normal signal strength for the router detected by our target device (a MacBook Air laptop) was around -39 dBm. The router and the target device were located around 30 cm apart. Merely by adjusting router settings, we were able to vary the signal strength of the router, as sensed by the target device, between -25 dBm and -48 dBm. By changing the distance between the target device and the spoofed router, we were able to further reduce the signal strength down to -87 dBm. This suggests that the adversary has a high degree of control in manipulating sensed signal strength. Based on this spoofing and Received Signal Strength Indicator (RSSI) manipulation capability, the Wi-Fi context matching attack becomes rather straightforward. The attacker can even have advantage in environments where number of Wi-Fi APs is low. For example, we observed that there are less than five APs in outdoors such as parking lot. In such cases, the attacker would only need to spoof $\mathcal{P}$'s side.

7.3.2.3 Manipulating Physical Environment Sensor Modalities

As discussed in [39], it may seem hard to manipulate physical modalities, Temperature T, Humidity H, Gas G and Altitude Al. For example, it appears that an adversary has to change the temperature or humidity of the entire environment surrounding the victim device which may be quite challenging or detected easily. However, in this section, we show that, by using off-the-shelf devices, manipulating physical context is not only feasible but also realistic and effective by tampering with the "local" environment close to one of the devices (e.g., an unattended $\mathcal{V}$). Our attacks do not require the compromise of the devices ($\mathcal{V}$ or $\mathcal{P}$), but rather only manipulation of environment close to their sensors. In order to monitor the current ambient readings as they are being changed, the attacker has to use his sensors. These ambient readings serve as a feedback for the attacker while he attempts to change the current $\mathcal{V}$'s ambience. The feedback sensor needs to be placed very close to the victim sensor so that the two provide similar readings.

Our experiments demonstrate how different contextual modalities can be *manipulated, controlled* and *stabilized* to enable successful relay attacks. Arbitrarily changing a sensor's readings, at the verifier's side, based on a physical activity may be straightforward but consistently maintaining and controlling these readings to match those at the prover's side, is non-trivial. For

example, it may be obvious that temperature can be increased using a hair dryer (a simple tool used in our temperature manipulation experiments), but how to maintain it at a desired level for a reasonable period of time (during which the attack can be launched) is not obvious. While we present several direct/explicit ways to manipulate many modalities, we also demonstrate some indirect/implicit techniques. For example, we show how altitude can be manipulated by changing pressure (i.e., without relocating the device to a different altitude). When performing the attacks, we need to consider that the attacker will not have access to the direct readings from the actual ($\mathcal{V}$) device and hence has to use his own sensors to monitor the current ambient readings during the attack. These ambient readings serve as a feedback for the attacker while he attempts to change the current $\mathcal{V}$'s ambience. The feedback sensor needs to be placed close to the victim sensor so that both provide similar readings.

Temperature Manipulation

We were able to successfully alter the temperature to a desired level using various household items, such as a hair dryer, a coffee mug, and ice cubes. All of our experiments were performed with Sensordrone devices serving as both $\mathcal{V}$ and the attacker's feedback sensor.

Increasing the Temperature: In situations where $\mathcal{P}$ (e.g., a car key indoors) is at a higher temperature than $\mathcal{V}$ (e.g., a car parked outside in winter), the attacker must increase the temperature. We first used a hair dryer to heat-up the area around the Sensordrone such that the temperature is increased to a desired level. To monitor how the temperature increases as we bring the hair dryer closer to $\mathcal{V}$, we first placed the hair dryer far enough and then brought the hair dryer closer to the sensors in a way that we can handle the increase in temperature gradient. In our experiment, we first tried to increase the temperature to 40 °C and then to 35 °C. After a few attempts, we could successfully increase the temperature to a desired level and stabilize for almost 2 minutes (Fig. 7.2). The lab temperature when the experiments were performed was around 26 to 27 °C. The hair dryer we used [173] had a power of 1875 watt AC. A video demonstration of our attack has been uploaded to YouTube [174].

Our next set-up uses two sensors, $\mathcal{V}$ sensor (VS) and feedback sensor (FS), to change the temperature. Depending on whether or not the attacker knows where the sensor is precisely located on $\mathcal{V}$ device, he may place FS either exactly on top of VS or away from it. We performed the hair dryer test such that: (1) FS is placed at the same place as VS; (2) FS is placed such

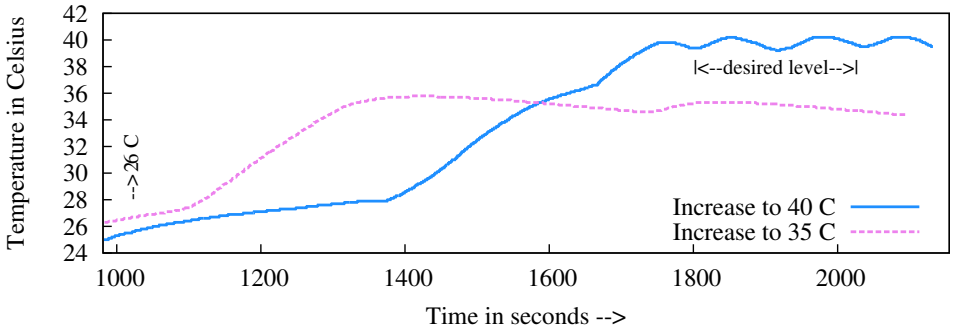


FIGURE 7.2: Increasing T to desired level (35 °C and 40 °C)

that VS is closer to hair dryer than FS; and (3) FS is placed such that FS is closer to hair dryer than VS.

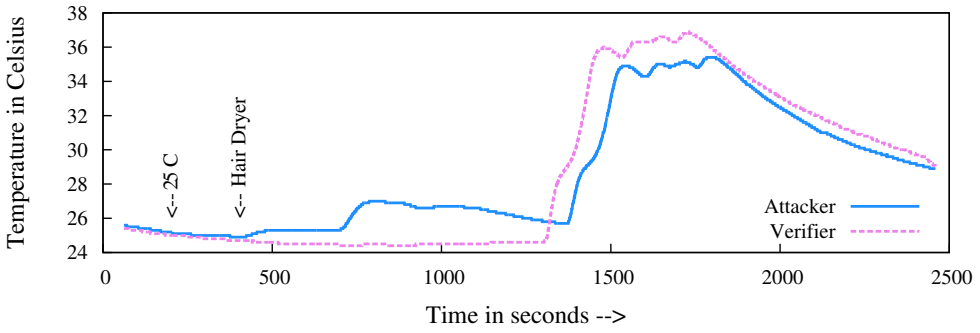


FIGURE 7.3: VS and FS on same location; the attacker trying to increase temperature to 35 °C.

For the first case, we were able to match the temperature on both sensors to a large extent when performing the heating activity (Figure 7.3). However, if the attacker does not know the location of VS then the sensor device closer to the hair dryer ends up getting more heated. These attacks are described in Appendix B.1 in detail. Hence, the attacker should heat up the whole area as he may not be able to place his FS exactly on top of VS. Subsequently, we tried to apply the heat not just focusing on one particular area but rather heating the entire area within a range of 15 cm. Using this approach, we could effectively change the temperature around VS with feedback from FS as the two temperature curves move side by side (Figure 7.4). We were able to control the temperature to a desired level within a variance of +/-0.3 °C for more than one minute in FS device.

Decreasing the Temperature: In some scenarios, it might be necessary for the attacker to reduce the temperature recorded by $\mathcal{V}$ (e.g., when $\mathcal{P}$ is indoors and $\mathcal{V}$ is outdoors during summer conditions). To decrease the temperature readings, we used an ice cube and rubbed it against the

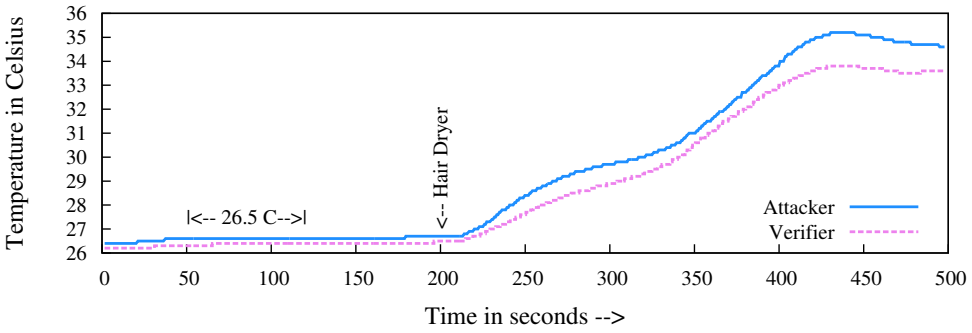


FIGURE 7.4: Heating an area; VS and FS within a range of 15 cm; the attacker trying to increase temperature to 35 °C.

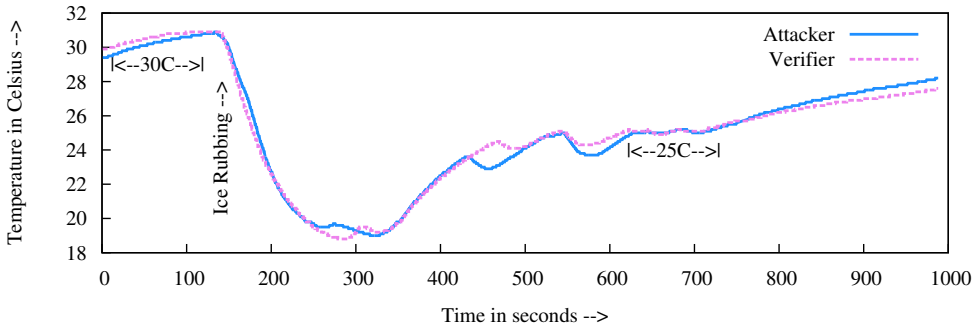


FIGURE 7.5: Decreasing temperature with an ice cube; the attacker trying to decrease to 25 °C.

sensor. The environment on the other hand increased the temperature. By using the ice cube, we first tried to drop the temperature below 20 °C and then let the environment increase the temperature naturally. This natural increase of the temperature was very slow, and when the temperature started increasing beyond the desired temperature level, we gently rubbed the ice again to stabilize the temperature. We conducted experiment in a parking deck where the ambient temperature was around 30 °C. Our goal was to change the temperature down to 25 °C. We rubbed the ice cube on the sensors (both $\mathcal{V}$ and feedback sensors) until the temperature decreased to less than 20 °C. Afterwards, the temperature started rising slowly naturally. When it reached around 25.2 °C, the ice cube was rubbed gently again on the sensors such that the temperature drops slightly. We were able to decrease the temperature and stabilize it at 25 °C for more than a minute after a few trials within a variance of +/-0.3 °C as shown in Figure 7.5.

Humidity Manipulation

To alter humidity, we used common household items such as hot coffee (for increasing humidity) and hair dryer (for decreasing humidity).

Increasing the Humidity: Coffee fumes when brought close to VS would increase the humidity level. An attacker has to move the hot coffee cup nearer to, and farther away, from the sensors to control the humidity level. Using this strategy, we were able to increase the humidity by 10%, i.e., from normal humidity of 55% to 65% (Fig. 7.6). The attacker needs to use FS to control the humidity. On our first attempt, we were able to control the humidity with a variance of +/-3% for almost 30 seconds. In the second attempt, we could raise the humidity to the desired level for more than one minute (106 seconds) with the same threshold.

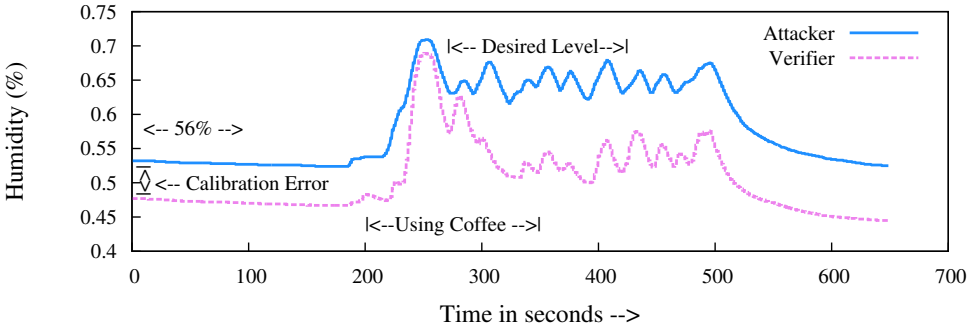


FIGURE 7.6: Increasing humidity with hot coffee; the attacker trying to increase to 65%.

Decreasing the Humidity: A hair dryer can be used to dry-up the air around the sensor to reduce the humidity. The setup of this experiment is similar to the hair dryer temperature increase experiment. We tried to decrease the humidity of VS by monitoring the humidity change on FS. When two devices are placed exactly at the same location, the humidity decreases and matches consistently between the two devices (Fig. 7.7). Even when the two devices are placed 15 cm apart, the drop in the humidity readings coincides (Fig. 7.8).

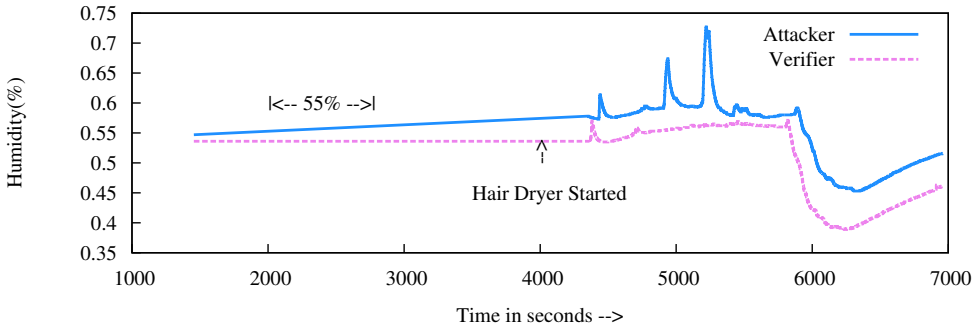


FIGURE 7.7: Decreasing humidity with hair dryer such that VS and FS are at same location; the attacker trying to decrease to 50%.

Gas Manipulation

Following prior work [39], we study Carbon Monoxide (CO) level as a modality for co-presence

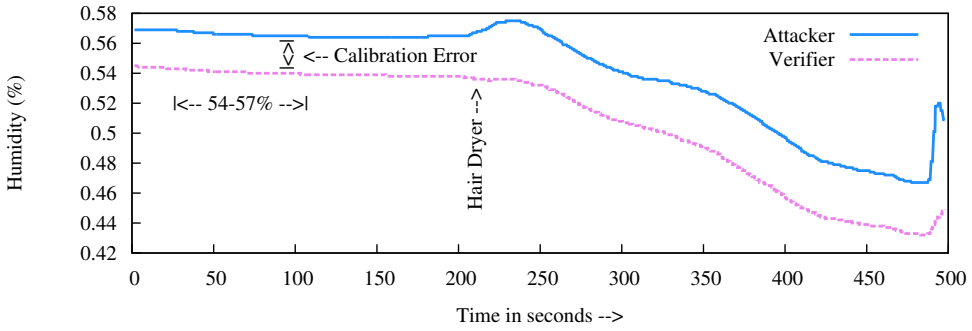


FIGURE 7.8: Decreasing humidity with hair dryer such that VS and FS are within a range of 15 cm; the attacker trying to decrease to 50%.

detection. While manipulating this modality, an attacker may not be detected even when he alters the gas content near either $\mathcal{V}$ or $\mathcal{P}$ (unlike the rudimentary model of Section 7.3.1.2), unless there is a significantly large change, or gas monitors are installed. This provides flexibility to the attacker to increase/decrease the CO level at both sides such that both readings match.

Increasing the Gas (CO): We performed several activities such as using a smoking cigarette to exhale a high amount of CO gas to the sensor, and using a car exhaust to increase the CO level. We also found out that room heaters emit gases which increase CO readings when we placed the sensor device on top the gas vent while the heater was turned on. The aerosol spray also increased the CO level when it was sprayed around on top of the sensor. The effect of different propane gas heaters as well as aerosols air fresheners on gas content has been mentioned in [175]. All these activities, though, increased the CO level abruptly, it takes a long time for sensor reading to descend back to normal, which provides the attacker with a sufficiently long attack window as shown in Fig. 7.9. The effects of cigarette and car exhaust on CO level are described in Appendix B.2 in detail. We observed these activities for more than five times, and noticed that it took more than thirty seconds to decrease by 1 ppm when gas level decreased below 10 ppm which is already above average of normal gas level.

Decreasing the Gas (CO): To reduce the gas level, an attacker needs to "purify" the air from the CO content around the sensors. We implemented this strategy using a kitchen exhaust fan which is used to remove pollutants. We found that when sensor was placed near the exhaust fan, it decreased the CO gas content.

The gas reading heavily depends upon the location of $\mathcal{P}$ and $\mathcal{V}$. In a heavy traffic or polluted area, this may be higher than 10 ppm while in a normal workplace, it may be around 0 ppm to 5

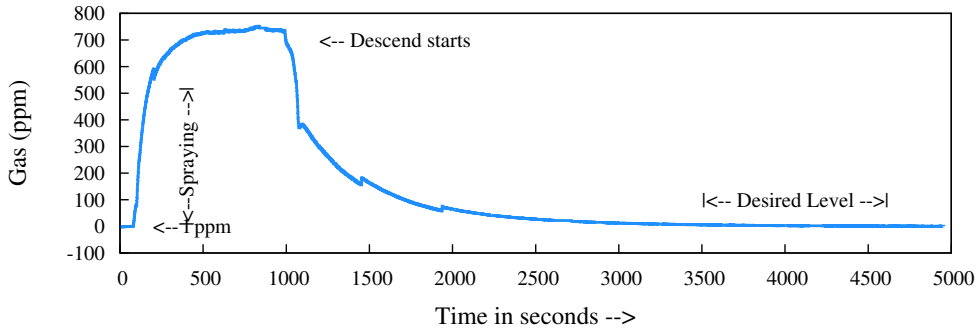


FIGURE 7.9: Effect of aerosol spray in CO level; increasing the CO gas level to arbitrary value and wait to decrease to desired level.

ppm. If $\mathcal{P}$ is located in low CO area while $\mathcal{V}$ is located in high CO area, the attacker may use the kitchen exhaust fan activity to decrease the CO level in $\mathcal{V}$'s location. However, if the attacker cannot reduce the CO level by significant amount, he can always collude with the attacker at $\mathcal{P}$'s side to increase the CO level using an aerosol spray. This can increase the CO level by significant amount and then it only takes a while to fall back to the normal gas level. This effect can be confirmed from Fig. 7.9.

Altitude Manipulation

The altitude of a location is inversely correlated to the pressure at that location. The Sensordrone device detects the pressure, and uses it to calculate the altitude based on a standard conversion method.

Manipulating sensors so as to increase or decrease altitude directly seems very difficult. In order to manipulate the altitude readings, one may physically carry the verifier device to a higher or lower altitude as needed. If the verifier device is portable (such as a stolen laptop), doing so is easy. However, there are many scenarios where directly changing the altitude is not feasible (e.g., when $\mathcal{V}$ is a car and $\mathcal{P}$ is a car key carried in victim's pocket). We show that it is still possible to manipulate altitude readings *indirectly* by manipulating the pressure readings.

Increasing the Altitude: To increase the altitude indirectly, an attacker must decrease the pressure near the sensors. To achieve this functionality, we created a low-cost air compressor. We placed the sensor inside a Ziploc bag and then used an electric air pump [176] to suck-up the air from the bag. When $\mathcal{V}$ is large in size or shape (such as a car), an attacker just needs to create an enclosure around its sensor, while if it is a portable/small device (e.g., a laptop), the device itself can be placed inside a bag. When the air pump sucks up the air around the sensors enclosed inside the

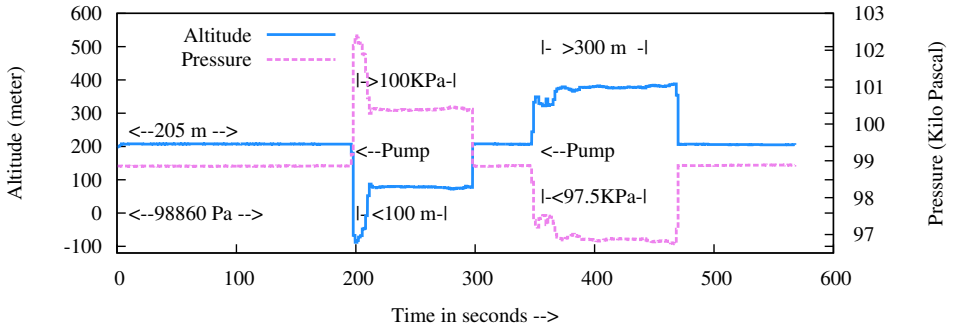


FIGURE 7.10: Using an air pump to change pressure to the sensors wrapped inside a Ziploc bag by pumping air in and out.

Ziploc bag, the weight of air exerted on the sensor is reduced. This reduces the pressure around the sensor and hence increases the altitude level. In our experiment, we effectively altered the altitude by more than 60 meters (Fig. 7.10). By using an air pump with a higher power, the attacker can further increase the altitude level. A vacuum cleaner may also be used in place of an air pump (as described in Appendix B.3).

Decreasing the Altitude: To decrease the altitude (i.e., increase the pressure), we placed the sensor inside a polythene bag and applied high pressure by squeezing the bag, blowing air into the bag, and finally using the air pump device to blow the air inside. First, we wrapped the sensor inside a polythene bag to see if there is any change in altitude when we blow air into the bag by mouth, or squeeze the air tight polythene bag. This increased the pressure by very high amount and decreased the altitude correspondingly. However, it was not doable in a controlled way, i.e., sometimes the altitude decreased by 5 meters while on other occasions, it decreased by 50 meters. Ideally, an attacker would want to have a relatively long time window where the desired altitude remains constant for him to perform the relay attack. To address this issue, we used the air pump mentioned above. Filling up the air into the bag increased the pressure and decreased the altitude such that it remained constant for almost 14 seconds. A video demo of this experiment has been uploaded to YouTube [177].

7.3.2.4 Manipulating Multiple Sensor Modalities Simultaneously

As demonstrated by prior work [38, 39, 98], a contextual co-presence detection system can use combinations of several sensor modalities. In such cases, the attacker needs to manipulate multiple modalities at the same time (multi-modality attacker). However, performing one activity

may be altering not only the target modality but also one or more other modalities that a system might be using for context detection, such as (T and H) or (Al and Au) even though they are not directly correlated.

For example, hair dryer increases temperature but also dries-up the air (i.e., potentially reduces the humidity) around the sensor where it is applied. It also changes the ambient noise. An attacker needs to manipulate in such a way that if the multiple modalities are involved in the system he should change the target modality without altering other modalities by effective amount. We also found that hair dryer activity results in a huge momentary change in gas level. However, the reading comes back to normal when hair dryer is applied for a long period of time. Altitude and pressure did not change with the hair dryer activity. Hair dryer activity also does not impact on RF signals. Hence, hair dryer activity can be used to manipulate the system which uses either temperature or humidity along with gas, altitude and RF signals.

Using aerosol spray to increase the gas content does not have effective change on any other modalities besides humidity. Similarly, updating RF signals does not seem to have any effect on physical modalities. Therefore, an attacker can simultaneously manipulate radio, temperature and gas while he hopes that audio, altitude and humidity either match the minimum criteria from both sides or is not used by the system.

Using an ice cube to decrease the temperature does not affect other modalities effectively. However, if the ice melts then it may affect the humidity of the space near the sensors. In our experiment, we saw that humidity fluctuates when we tried to decrease the temperature using an ice cube. Hence, using an ice cube to decrease temperature activity can be used with all other modalities except altitude and humidity.

Hot coffee cup changes the humidity along with the temperature, while other modalities remain unchanged. In this case, an attacker can manipulate humidity along with radio, audio and gas while he cannot control temperature and humidity together.

When an attacker has to use an air pump or vacuum cleaner to increase or decrease the altitude, it affects ambient noise. Also, an air pump was used in conjunction with a Ziploc bag where the sensors were wrapped to create an enclosed space. When the attacker performs such activity in an enclosed space, it will be difficult for him to change gas, temperature or humidity. Thus, we only claim that the attacker can manipulate altitude along with radio modalities.

To summarize, our attacks support the following combinations of multi-modality manipulations: (1) Al, B, W; (2) Au, B, G, (increase for H), W; (3) Au, B, G, (decrease for T), W; (4) Au, B, G, W; (5) B, G, H, W; (6) B, G, T, W. However, a more sophisticated attacker (than the one we considered) may use different techniques to possibly attack other combinations too.

7.3.3 Environment Manipulation: Analysis

In light of the attacks presented in Section 7.3.2, we present our attack analysis on multi-modality attackers who can simultaneously control multiple sensor modalities, in addition to the single-modality attacker. We assume that a contextual attacker can manipulate radio contexts in *both directions*. The same assumption applies to Gas sensors in light of our aerosol spray attack.

7.3.3.1 Analysis Methodology

To fairly evaluate the resilience of co-presence detection systems in the presence of our contextual attacker, we used the same datasets and the same set of features originally used to evaluate the systems in question. We use datasets in Chapter 6 to evaluate the resistance of the respective systems against multi-modality attackers. In addition, we conducted new audio relaying experiments to collect data and evaluate audio-based co-presence detection performance. Furthermore, we collected a new dataset corresponding to the audio-radio-physical system.

We used the same classification techniques as in Chapter 6 (Decision Tree and Random Forest), implemented in Scikit-learn [178]. The results are reported after running ten-fold cross validation. We use *False Positive Rate (FPR) as a metric to represent the attacker's success probability*. FPR corresponds to "non co-presence" samples which are mislabeled as "co-presence", reflecting the security of the system (higher the FPR, lower the security). We use False Negative Rate (FNR) as a metric to represent the usability of the system. FNR represents "co-presence" samples that are mislabeled as "non co-presence" (lower the FNR, better the usability). F1 score is reported only for the overall performance of the classification model under zero-modality attack. Please check Equations 4.1 and 4.2 for the definitions of FPR, FNR and F1 score.

Whenever multiple sensor modalities are used, we fuse the data from these modalities before feeding it to the classifier. We considered the following fusion approaches.

- **Features-fusion:**

  The features of all sensor modalities are together fed to the classifier. The decision of co-presence or non co-presence is made one-time only based on the output of the prediction model. This is the approach followed our work in Chapter 6 [38, 39].

- **Decisions-fusion:**

  Each of the n sensors (with all its features) is used separately by the classifier. As result there are n decisions made. All decisions are then combined to produce a final decision. This is an approach that has not been used for co-presence detection in previous works. Decisions-fusion can aggregate decisions from single sensor modalities or from subsets of sensor modalities, for example, three subsets can be built on top of seven sensors: acoustic = {Au }, radio = {B, W }, physical = {Al, G, H, T }. In the latter fusion approach, classifiers of subsets are built using features-fusion.

  We consider two methods to aggregate decisions. The first method is based on a simple majority voting (hereafter referred to as *equal voting*) which takes binary decisions from all sensors with equal weights. The second method is a novel variant, we call *weighted voting*, which fuses decisions with different weights assigned to each sensor. We start with a list of sensors, $S_1, S_2, ..., S_n$, sorted by the order of attack resilience according to the single modality attack, the weight assigned to S_i is computed as: $1/N * i$, where $N = \frac{n*(n+1)}{2}$, i is the position of S_i in the sorted list. As an example, the decisions-fusion weighted voting for Audio-Radio is done as follows. Sensors Au, B, W have performance: 3%, 2.7%, 99.8% (Table 7.2), respectively. The sorted list is therefore [W, Au, B]. When these sensors are fused, their weights will be assigned as: 1/6 for W, 2/6 for Au and 3/6 for B. Note that in equal voting, the weight of each sensor is 1/3. When more sensors are fused, weights will be adapted with similar scheme.

7.3.3.2 Audio-Only System

Halevi et al. [98] proposed the use of (only) audio for co-presence detection. Their work showed that audio is a good ambient context resulting in 100% accuracy and 0% False Positive Rate (FPR). To assess how an attacker can manipulate ambient audio via the streaming attack (Section 7.3.2.1), we conducted a set of experiments to collect about 100 audio samples for the non

TABLE 7.1: Relay attack success rate (FPR) for audio streaming via Wi-Fi and Cellular networks

Acoustic relaying environments ($\mathcal{P}$ freq $\rightarrow$ $\mathcal{V}$ freq)	**Wi-Fi**	**Cellular**
High $\rightarrow$ *Medium*	100%	40%
High $\rightarrow$ *Low*	100%	20%
Medium $\rightarrow$ *Medium*	100%	0%
Medium $\rightarrow$ *Low*	100%	60%
Low $\rightarrow$ *Low*	20%	0%

co-presence case. The audio streaming was done over two different channels: Wi-Fi and cellular data. $\mathcal{P}$ was a Galaxy Nexus device while $\mathcal{V}$ was a Galaxy S3 device. Unidirectional streaming of the audio from $\mathcal{P}$'s side to $\mathcal{V}$'s side was done between a pair of devices (from a Galaxy S4 to an iPhone 5 in the case of the cellular data channel, and from a MacBook Air to a ThinkPad Carbon X1 in the Wi-Fi channel). The attacker devices used a Skype connection as the audio relay channel.

The audio features used in [98] are based on audio frequency. Therefore, to evaluate the impact of frequency on the attack feasibility, we tested three different ranges of ambient audio frequencies collected by controlled experiments where we set up the ambient noise surrounding recording devices falling into different categories. *Low ambient audio* (frequency less than 100 Hz); *Medium ambient audio* (frequency in the human audible range, at around 500 Hz); *High ambient audio* (frequency 5000 Hz or more).

We used the dataset for ambient audio of previous work [38] which collected ambient acoustic data to build the classification model (F1 of 0.86 and FPR of 9.3%). The 100 samples we collected via audio streaming channels are fed to the classifier for prediction. Table 7.1 presents the FPR of non co-presence detection under the streaming attacks over Wi-Fi and cellular data channels. The results indicate that the attacker (1) has a higher chance of success using the Wi-Fi channel and (2) could be thwarted when either the ambient audio at $\mathcal{P}$ is low frequency or if the ambient audio at $\mathcal{V}$ is high frequency.

This simple streaming attack with commodity devices shows that the audio-only system is highly vulnerable to relay attacks, especially via the Wi-Fi channel. The attack has very high success rate regardless of hardware variations and network delays inherent to streaming. However, an attacker can succeed only when relaying ambient audio from a higher frequency acoustic environment to a similar or lower frequency acoustic environment, such that, the higher

frequency dominates the lower frequency, and makes $\mathcal{V}$ falsely record $\mathcal{P}$'s ambient noise instead of the real "localized" ambient noise.

The audio features we used, i.e., the ones proposed in [98], are not sensitive to time synchronization. This is effective in terms of co-presence detection (i.e., results in very low FNR). However, as we can see from our experiments, these features also enable the attacker to succeed in the relay attack with a very high chance.

7.3.3.3 Audio-Radio System

To analyze the attack, in each run, the non co-presence samples in the test data were transformed as below.

Audio: Because raw audio data is additive, and one-side context manipulation for audio is tested, an adversary can be modelled by replacing $\mathcal{V}$ side audio (X_a) to be the sum of its own ambient audio and $\mathcal{P}$ side audio ($X_a + X_b$).

Radio (B and W): In Section 6.3.2.1, the set of radio records from two devices A and B are defined as: $S_a = \{(m_i^{(a)}, s_i^{(a)}) \mid i \in \mathbb{Z}_{n_a-1}\}$, and $S_b = \{(m_i^{(b)}, s_i^{(b)}) \mid i \in \mathbb{Z}_{n_b-1}\}$, where (m, s) with m is an identifier and s is associated signal strength of a beacon; n_a and n_b denote the number of different beacons (i.e., Wi-Fi access points or Bluetooth devices). The both-sides contextual adversary can be modeled by replacing S_a with $S_a \cup \{(m, s)\ \forall (m, s) \in S_b, m \notin S_a^{(m)}\}$, and S_b with $S_b \cup \{(m, s)\ \forall (m, s) \in S_a, m \notin S_b^{(m)}\}$.

We considered two approaches of fusing sensor data against bi-directional relay attacks and showed which of them is more suitable for resisting against the presence of contextual attackers.

Table 7.2 (columns 1, 2 and 3) presents the analysis results of training model combining all three audio-radio modalities (Au, B and W) and testing with different attacks. Zero-modality attack shows the very low FPR with both fusion methods. The FNR for decisions-fusion is higher compared to that for features-fusion. For features-fusion, the results are aligned with the ones reported in Section 6.3.2.2.

In single-modality attack, manipulating Wi-Fi, the dominant feature, results in a very high success rate with features-fusion. The results change when decisions-fusion was applied (equal voting and weighted voting). In such case, manipulating any single sensor, even the most powerful one, does not significantly degrade the overall security. The FPR in case W was manipulated

TABLE 7.2: **FPRs with/without different contextual attacks in various audio/radio/physical systems**. **Notations:** Sets of manipulated sensors are put inside curly braces $\{\}$. $\{\widetilde{X}\}$ denotes an arbitrary set of sensor modalities. **Fusion**: F denotes features-fusion, D-S denotes decisions-fusion, equal voting from single modalities, D-S (w) denotes decision-fusion, weighted voting from single modalities, D-M denotes decisions-fusion, equal voting from subsets of modalities. **Result highlights**: Manipulation of sensor modalities, especially multiple of them, can significantly reduce security (increase FPR) in most cases. Decisions-fusion can help improve security when dominant sensors are manipulated, but it may reduce usability (increase FNR). Grey highlights the improvement of decisions-fusion.

		Audio-Radio				**Physical**				**Audio-Radio-Physical**			
	Fusion	*F*	*D-S*	*D-S (w)*		*F*	*D-S*	*D-S (w)*		*F*	*D-S*	*D-M*	*D-S (w)*
		(1)	(2)	(3)		(4)	(5)	(6)		(7)	(8)	(9)	(10)
Zero-modality		2.0%	2.0%	2.5%		7.5%	13.0%	58.6%		3.0%	27.1%	6.9%	34.5%
	FNR	1.4%	12.0%	2.7%		3.9%	14.5%	7.8%		0.0%	0.3%	0.0%	1.5%
	F1	0.977	0.925	0.979		0.928	0.861	0.810		0.990	0.923	0.980	0.898
Single-modality	{Au}	3.0%	3.0%	2.5%	{T}	8.3%	17.0%	89.7%	{Au}	87.7%	45.3%	36.9%	39.4%
	{B}	2.7%	9.0%	14.8%	{G}	11.9%	20.0%	69.0%	{B}	100%	45.8%	36.9%	35.0%
	{W}	99.8%	8.0%	2.5%	{H}	15.3%	24.4%	68.0%	{W}	12.3%	44.8%	35.0%	39.9%
					{Al}	55.1%	33.1%	56.0%	{Al}	5.4%	37.9%	6.9%	28.1%
									{G}	5.9%	29.6%	6.9%	40.9%
									{H}	3.4%	29.1%	6.9%	54.7%
									{T}	3.4%	31.5%	6.9%	38.9%
Multi-modality	{Au,B}	3.6%	96.0%	99.0%	{G,T}	13.9%	40.1%			{B}∪{$\widetilde{X}$}:	{2 sensors}:	{Au, B}∪{$\widetilde{X}$}:	{2 sensors}:
	{Au,W}	100.0%	96.0%	2.5%	{G,H}	15.7%	57.5%	69.5%		100.0%	32.0-75.4%	>97.5%	28.1-68.5%
	{B,W}	99.8%	100.0%	100.0%	{H,T}	29.6%	41.9%			{Au}∪{$\widetilde{X}$}:	{3 sensors}:	{Au, W}∪{$\widetilde{X}$}:	{3 sensors}:
	{Au,B,W}	100.0%	100.0%	100.0%	{Al,T}		50.6%	89.7%		>74.9%	37.4-97.5%	>88.2%	36.5-100%
					{Al,H}		61.2%	65.0%		{$\widetilde{X}$}\{Au, B}:		{Al,G,H,T}:	{4 sensors}:
					{Al,G}		65.5%	65.0%		12.3%	97.5-100%	9.90%	58.6-100%
					{G,H,T}	31.1%					rest: 100%	{B,W}: 36.9%	rest: 100%
					{Al,G,H}			65.0%				rest:	
					{Al}∪{$\widetilde{X}$}	64.7-100%						6.9-87.7%	
					rest		100%	100%					

decreases from 99.8% (features-fusion) down to 8% (decisions-fusion, equal voting and 2.5% (decisions-fusion, weighted voting). The performance of audio and radio sensors is comparable to that reported in Section 6.3.2.2 with F1 ranges from 0.857 for Au to 0.989 for W). This explains why decisions-fusion reduces the overall performance slightly (F1 reduces from 0.977 to 0.925) in case of zero-modality attack but significantly improves the security under a single-modality attack. The security is very low in multi-modality attack, and neither of the fusion approaches could restore the security level when majority of the sensors are under attacker's control.

7.3.3.4 Physical System

In Section 6.4, four physical modalities (Al, H, G, and T) were introduced for co-presence detection. The performance of the features-fusion based classifier trained with their dataset is good (F1 of 0.957, FPR of 5.81%) against a zero-modality adversary.

Based on our attacks against physical modalities (Section 6.4), we consider an adversarial model where an attacker can manipulate the physical context on one side (unattended verifier) to match the sensor readings at the other side (prover). To model this attack, all non co-presence samples in the test set were transformed to the "attack" value (distance 0). The distance is set to 0 as data collection in [39] was done by a single device at a given point of time, hence, no hardware effect or calibration error was taken into account. The non co-presence class in the dataset is about 18 times larger than co-presence class. To correct this imbalance, we applied the same under-sampling as in [39]: we divided the non co-presence samples into 19 subsets, ran several rounds of cross validation taking 10 subsets in each round and aggregated the results in the end. In addition to the features-fusion employed in [39], we tested the decisions-fusion similar to our audio-radio system analysis in the previous section.

Table 7.2 (columns 4, 5 and 6) shows our analysis results. The system performance in zero-modality attack is well-aligned with the one reported in [39]. As in [39], among four physical modalities, AI performs the best. Consequently, manipulating only AI degrades the security vastly with features-fusion (FPR increases to over 50%). Decisions-fusion in general brings lower security and lower performance/usability in zero-modality attack and single-modality attack. However, it avoids the dominance of sole sensor in case the attacker can control such sensor (AI in this case). Decisions-fusion can also help improve security against a multi-modality attacker who manipulates AI along with other sensors. Compared to audio-radio system, in physical system, attacking each single modality results in higher success rate.

7.3.3.5 Audio-Radio-Physical System

With an app designed and implemented in Chapter 6, Sections 6.3 and 6.4, we recorded sensor data from different devices. Each device, in a pair of devices, was connected to its own Sensordrone device. Two users were involved in the data collection. Data was collected at different locations in two countries for ten days. The resulting dataset has 203 non co-presence samples and 335 co-presence samples.

We collected data from pairs of devices, and therefore hardware variance and calibration errors between co-presence device sensors need to be taken into account. When we try to model the contextual attack on given sensor(s), distance 0 does not ensure that the attack will succeed. As the classifier is trained with data which may contains noise, we compute the mode of the

histogram for distance values for the co-presence samples. As the data aggregated is from two participants, histograms of distance values are not uninomial but multinomial. Multinomial distribution implies several modes. For each physical sensor, we choose a mode value and assign it as the distance value. The mode values for Al, G, H and T are 13.54, 0.3, 6.61 and 0.153, respectively. As the manipulation by replacing the radio data at both sides has to be identical, the distance features for radio sensors are set to 0.

Table 7.2 (columns 7, 8, 9, and 10) reports our analysis results with different fusion methods. Under zero-modality attack, features-fusion performs the best while decisions-fusion from single modalities performs the worst. Features-fusion uses all possible features for training so that the classifier can be built based on the best features or best combination of features (B and Au with our current dataset). Thus, it returns the best results (in the absence of context manipulation) compared to any other ways of fusing sensor data. Decisions-fusion based on single modalities lets the worst sensors being able to contribute to the voting scheme, thus bringing down the overall performance. This is the case in our dataset where radio sensors and audio sensor perform better than physical sensors. Note that if all sensors perform equally well, features-fusion and decisions-fusion would not differ much. Decisions-fusion from subsets of sensors has a moderate performance, worse than features-fusion but better than decisions-fusion from single modalities. This hybrid approach avoids mis-learning as in the case of using a single modality only.

Let us now assess the security of this co-presence detection system when any single modality is controlled by the attacker. Depending on how sensors are fused, the impact of manipulated sensor varies. In features-fusion, as the classifier decision relies on the best features of dominant sensors, the FPR increases drastically when such sensors are manipulated (i.e., Au or B in our dataset). In contrast, when weaker sensors (physical or W) are manipulated, it has a relatively small impact on the security as the resulting FPR increases a bit compared to a zero-modal attack (especially for W). Decisions-fusion reduces attacker success rate when single sensor is manipulated, for example, FPR of manipulating B decreases from 100% to 35% (decisions-fusion, weighted voting). Recall that manipulating single sensor is not difficult as we demonstrated in Section 7.3.2.

An attacker has the highest chance to succeed if he can control the dominant sensors or a subset of sensors that contain the dominant sensors. In such case, the success rate could reach 100% with only one single dominant sensor (i.e., B in our dataset) if the system uses

features-fusion or with majority dominant sensors (i.e., Au and B). In most cases, attacking the set of weak sensors (e.g., {Al, G, H, T}) does not impact the security much, except when system uses decisions-fusion from single modalities.

7.3.4 Summary

Contextual co-presence detection has been shown to be a very promising relay attack defense in many mobile authentication settings suitable for off-the-shelf, sensor-equipped devices. We presented a systematic assessment of co-presence detection in the presence of a context-manipulating attacker. Our work suggests that tampering with the context can be achieved with simple yet effective strategies, and the security offered by co-presence detection is therefore weaker than previously believed. We also suggested potential countermeasures (e.g., decisions-fusion based machine learning, especially involving weighted voting) that may be used to strengthen the security of co-presence detection against a multi-modality attacker.

7.4 Attack on Two Factor Authentication: *Sound-Danger*

7.4.1 Introduction

In Chapter 6, we showed that we can use environmental context such as RF and audio (Section 6.3) or physical ambient parameters (Section 6.4) to detect proximity of two devices and prevent relay attacks. Detecting proximity of the two devices can be used as an authentication based on token ("something you have"). This can be used in conjunction with the traditional password based authentication ("something you know") forming a two factor authentication (2FA). Currently, different web applications such as Google, Facebook, etc., have deployed 2FA where users enter one-time PIN (OTP) from the token over to the authentication terminal after entering their passwords. This improves security because the attacker now needs to not only guess the user's password but also the current OTP value to hack into the user's account. The use of a general-purpose smartphone as a token [43–45], as opposed to a dedicated device [46, 47], helps improve usability and deployability of 2FA.

As mentioned in Section 2.1.6, *Sound-Proof* [30], elicits *ambient sounds* to detect the proximity between the phone and the login terminal (browser) to provide 2FA with no degradation in usability. However, in Sections 7.2 and 7.3, we showed that such systems can be thwarted by a

context manipulating attacker with a little effort. In this section, we set out to closely inspect the security of *Sound-Proof* [30], motivated by its very appealing usability and practicality features. We identify a fundamental weakness of the *Sound-Proof* system, namely, the remote attacker against *Sound-Proof* does not have to predict the ambient sounds near the phone, but rather can make the phone create predictable or previously known sounds, or wait for the phone to produce such sounds (e.g., ringer, notification or alarm sounds). Since the phone itself creates such noise, the recordings would be dominated by these sounds rather than the ambient noises.

Exploiting this weakness, we introduce and build *Sound-Danger*, a full attack system that can successfully compromise the security of *Sound-Proof*. The attack involves remotely buzzing the victim user's phone, or waiting for the phone to buzz on its own, and feeding the corresponding sounds at the browser to login on behalf of the user. The attack works precisely under the limits of *Sound-Proof*'s threat model, only uses the information available in hacked password databases (e.g., passwords, phone numbers or other account information [179–185]), is fully remote and can be launched against multiple user accounts. We note that phone numbers, in particular, are readily available in password databases as they are commonly used to facilitate account recovery in case of forgotten username/password and are essential for 2FA-supported web services which often need to send OTPs to users' phones via SMS (*Sound-Proof* also supports fallback to traditional 2FA [30]).

7.4.2 Background

Sound-Proof [30] is claimed to be a usable and deployable zero-effort 2FA mechanism, which does not require interaction between a user and the 2FA application on the device during the authentication process. In *Sound-Proof*, the second authentication factor is the proximity of the user's phone and the client terminal (browser), which is verified by the application on the phone by comparing the ambient noise recorded by the phone and the browser.

7.4.2.1 Threat Model

The primary goal of *Sound-Proof* is to defeat a remote attacker, who may be attempting to login into a victim user's account from a remote machine, which is in full control of the attacker. *Sound-Proof*'s threat model assumes that this remote attacker has the knowledge of the victim user's username and password. This information can be learned, for example, via leaked password

databases of the web service that may be using *Sound-Proof* or other web services for the purpose of authenticating its users. The attacker's goal is to authenticate to the web service on behalf of the user and possibly compromise multiple user accounts. *Sound-Proof* assumes that the attacker has not compromised the user's phone and/or the user's terminal. If the attacker gains control of one of the victim's devices, the security of any 2FA scheme reduces to the security of password-only authentication. Also, *Sound-Proof* does not consider targeted attacks such as those involving co-located malicious entities that are in close physical proximity of the victim.

This threat model may be weaker than that considered by traditional 2FA schemes involving OTPs. However, as argued in [30], given the prominence of remote attacks, this is still a very legitimate model. If more and more web services and users adopt *Sound-Proof* given its unique zero-effort feature, and remote attackers could still be thwarted, this will be a major improvement to the state of web authentication in practice.

As such, our proposed *Sound-Danger* system follows a threat model very similar to that of *Sound-Proof*. We consider that the attacker gets other user information from the leaked password database besides user credentials. That is, we assume that the password databases store phone numbers for password-only or 2FA implementations in order to send account recovery information or verification codes [183, 185], IP address information from which the users log in [182–184], or even users' physical address information [184]. The *Sound-Danger* attacker uses the phone numbers to perform active attacks while it utilizes IP addresses or physical address information to locate the users and their timezones. By identifying the timezone of the users, the attacker can estimate when a particular noise may occur at the users' side, such as morning alarms. Since many users often use the same usernames across multiple web applications, the attacker can utilize the username information to perform attacks based on notifications triggered by apps that use the same username. For example, if a user has the same username in the leaked database server and Skype, the attacker can send a notification (e.g., a friend's request) to user's Skype account. Like in *Sound-Proof*'s threat model, the *Sound-Danger* system does not attempt targeted attacks. Rather, it assumes that the attacker can collect general population statistics through online user surveys in order to devise specific attack strategies against a population of users for compromising multiple user accounts.

7.4.2.2 Implementing *Sound-Proof* Framework

As a prerequisite to evaluating the *Sound-Danger* attack system, we first re-implemented *Sound-Proof*, as described in [30]. We implemented phone-side, server-side and browser-side applications as described below (the flow diagram of our implementation is shown in Figure 7.11):

- *Phone Application:* We created an Android app that stays idle in the background and is automatically activated when a push message arrives. Google Cloud Messaging (GCM) is used to send a push message from the browser to the Android phone. When GCM push message arrives from the browser for recording, the Android app automatically gets activated and starts recording the ambient noise. The app stops recording as soon as another GCM push message arrives.

- *Web Server and Browser Application:* The server component is implemented using PHP while the browser component is implemented in HTML and JavaScript. Browser application has a simple button to control the recordings on the browser and on the phone. When the button is pressed to "start recording", the browser application sends GCM push message to the Android phone. If the button is pressed to stop recording, a "stop recording" GCM push message is sent to the Android phone. In the meantime, the browser application also starts recording ambient noise. Thus, the browser application has two main functions: (1) sending start/stop recording commands, i.e., GCM push messages, to the Android phone, and (2) recording ambient noise. In order to record ambient noise through the browser, we use HTML5 WebRTC API [186]. In particular, we use navigator.getUserMedia() API to access the local microphone from within the browser.

Time Synchronization: As the two devices (phone and terminal running browser application) may have two different local time clocks, our implementation, like *Sound-Proof*, requires the recordings from these devices to be synchronized. For this reason, both the phone and the browser applications run a simple time synchronization protocol with the web-server. Similar to *Sound-Proof*, the protocol is implemented over HTTP that allows each device to compute the time difference between the local time and the server time. Each device runs the time synchronization protocol while it is recording the ambient audio. Both devices compute their round-trip time

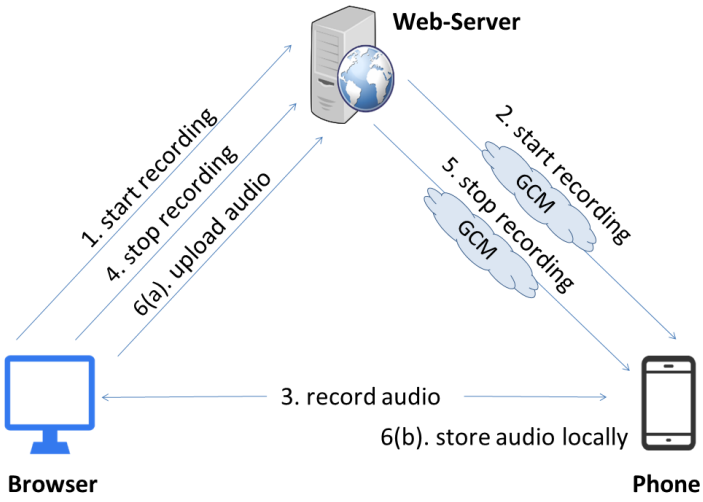


FIGURE 7.11: System flow diagram for our implementation of *Sound-Proof*. The phone and the computer (browser) record ambient audio using their microphones. The audio from the browser is uploaded to the server while that from the phone is stored locally. *Sound-Proof* sends the browser recordings to the phone using secured channel (not implemented in our design).

delay (θ) and then clock difference (δ) with the web-server as shown in Equation 7.1.

$$\theta = t_2 - t_0; \qquad \delta = t_2 - \frac{\theta}{2} - t_1; \tag{7.1}$$

Here:

- t_0 is the device's timestamp of the request transmission,
- t_1 is the server's timestamp of request reception and response transmission, and
- t_2 is the device's timestamp of response reception.

During our offline analysis of audio samples, the recordings from each of the devices are adjusted taking into account the clock difference (δ) with the web-server.

7.4.2.3 Implementing and Testing *Sound-Proof*'s Correlation Engine

Correlation Analysis: Correlation analysis between an audio pair is implemented in a similar fashion as *Sound-Proof*. That is, we used one-third octave band filtering and cross-correlation to get a similarity score of an audio pair, as described below:

- *One-third Octave Bands:* We divide the audio samples into different bands based on frequency. Each band covers a specific range of frequencies. A frequency is said to be an octave in width when the upper band frequency is twice the lower band frequency. A one-third octave band is defined as a frequency band whose upper band-edge frequency is equal to the lower band frequency multiplied by the cube root of two [187]. The audio spectrum from 20Hz to 20kHz can be divided into 32 one-third octave bands with the center frequency of 19th one-third octave band set to 1000Hz. The center frequency of the lowest band is 16Hz covering from 14.1Hz to 17.8Hz, while the center frequency of the highest band is 20kHz covering from 17.78kHz to 22.39kHz [188].

  Since we are targeting *Sound-Proof*, we divide the audio into the bands ranging from 50Hz to 4kHz. *Sound-Proof* utilizes only these set of bands, as these bands provided the best Equal Error Rate (EER) in the analysis reported in [30]. Hence, we only use the sixth band with the center frequency 50Hz to the twenty sixth band with the center frequency 4kHz, i.e., we consider only twenty bands out of the thirty two available bands. We use twentieth order Butterworth bandpass filter [189] in MATLAB to split the audio samples into these bands.

- *Cross Correlation:* We use the same system that was implemented in [98] to correlate ambient noise. *Sound-Proof* also closely follows this system for calculating cross-correlation. We use standard cross-correlation function to measure the similarity between the time-based signals X_i and X_j. To calculate the similarity, we first normalize the signals according to their energy. Then, we calculate the correlation between each signal at different lags and use maximum correlation value. The correlation between two time-based signals X_i and X_j is measured as:

$$Corr(i,j) = max(CrossCorr(X_i, X_j)) \tag{7.2}$$

  Sound-Proof also considers the lag to get the cross-correlation. This plays a major role to prevent attacks on the system when an attacker submits a similar audio sample as that in victim's environment, which may be separated by a certain lag. *Sound-Proof* has bound the lag l between 0 and l_{max}, where it sets l_{max} to 150ms. Hence, in our attack analysis,

we also check the maximum cross-correlation of audio pairs with the time lag bound to 150ms. This lag value yielded a low EER in *Sound-Proof*'s analysis reported in [30].

Data Collection and Experiments: We collected audio samples using the framework described in Section 7.4.2.2 at different locations such as lab/office, home, cafe, and library. We used Google Chrome on MacBook Air and Samsung Galaxy S V to record the audio samples using our implementation of *Sound-Proof*. We collected total of 525 audio pair samples and mix-matched them to get the correlation between each audio pair. The audio recordings were around 8 seconds long which were trimmed to 3 seconds after time synchronization for the correlation analysis (similar to [30]).

Similar to *Sound-Proof*, our implementation uses one-third octave band filtering and cross-correlation to calculate the similarity score between an audio pair, as described above. We use octave band filtering to split 3 second long audio recordings from both devices into 20 one-third octave bands. Maximum correlation is computed with the time-lag bound to 150ms between these audio pairs in their respective bands. The average correlation value obtained from these bands is the correlation between the audio pair.

The audio pairs which are co-recorded (recorded at the same location and at almost the same time) are labeled as True Positive (TP) and the rest are labeled as True Negatives (TN). Once the correlation values for each of the co-recorded audio pairs as well as non co-recorded audio pairs are calculated, we compute the FPR (False Positive Rate) and FNR (False Negative Rate) as a function of the correlation threshold. FPR for a given threshold defines the fraction of audio pairs (out of all audio pairs) which are not co-recorded but are classified as valid (TP) at that threshold, while FNR for a given threshold defines the fraction of co-recorded audio pairs (out of all audio pairs) that are classified as invalid (TN) at that threshold. Using FPR and FNR at different threshold values, we calculate EER (Equal Error Rate) and determine the optimal value of correlation threshold (T_c) at which FNR and FPR are equal.

From the data collected in our experiments, we obtained the optimal threshold T_c of 0.1524 yielding an EER of 0.1607. The correlation threshold set in our experiment is in line with the correlation threshold of *Sound-Proof* (0.13) [30]. Moreover, our other parameter settings are exactly the same as in *Sound-Proof*'s implementation [30], i.e., using the audio samples each of length 3 seconds, filtering the audio samples into 20 different one-third octave bands,

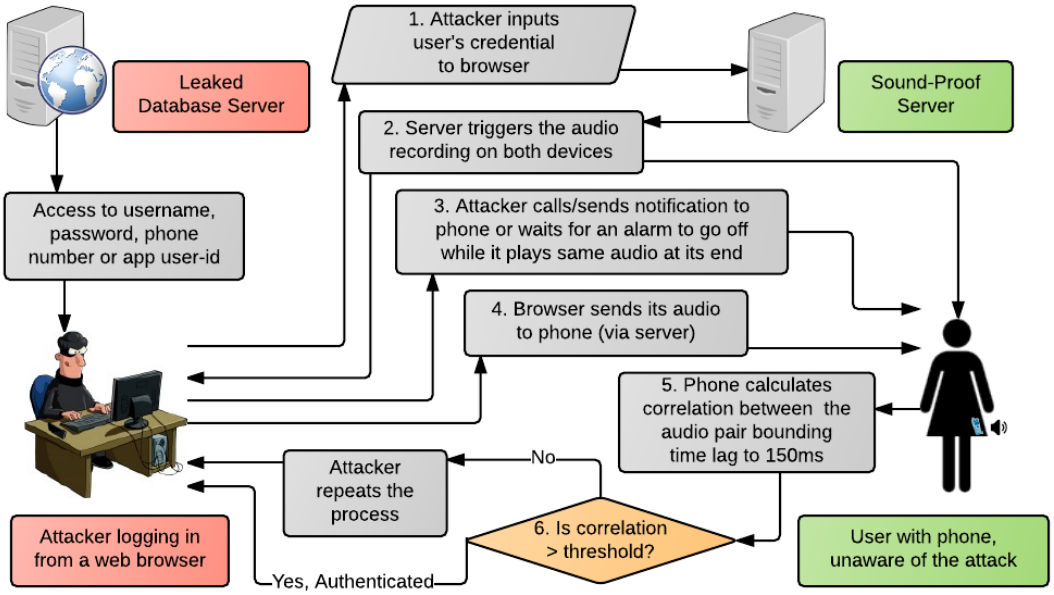


FIGURE 7.12: ***Sound-Danger* Attack Flowchart:** The attacker (human or bot) enforces the ambient audio to be highly similar at both (attacker's and victim's) ends by making calls or sending notifications to the victim's phone, or waiting for an alarm to go off at the victim's phone, and by simultaneously feeding the same sounds at its own end. The attacker would succeed in logging into the webservice, while the user may remain unaware of the attack or even when the attack is detected, the account may already have been compromised.

and cross-correlating the audio samples with time lag bound to 150ms. Since our correlation threshold is higher than that used in *Sound-Proof*'s implementation, it means that attacking our implementation will be harder than attacking *Sound-Proof*'s implementation reported in [30]. In other words, the *Sound-Danger* attack against our implementation (threshold 0.1524) would imply an attack against *Sound-Proof*'s implementation (threshold 0.13) [30]. Nevertheless, we analyze the performance of *Sound-Danger* for different (higher) correlation threshold values in the attack analysis in Section 7.4.4, and show that the attacks still work well even at such higher thresholdization.

7.4.3 Attacks

As the threat model of our *Sound-Danger* attack system suggests, we assume that the attacker is already in possession of the victim's username and password but is not co-located with the victim (i.e., victim's phone). The attacker's goal is to satisfy the second factor requirement, which is to fool the system into accepting the co-location of the *attacker's terminal* and the *victim's phone*.

We consider two types of attacks against *Sound-Proof*, both of which exploit the sounds generated by the phone itself. The first type of *Sound-Danger* attack is the active attack, where the

attacker performs an activity by which a sound (a phone ringing tone or an app-based notification) would dominate the ambient audio around the victim's device. Since the attacker is aware of the audio produced at the phone's end, it can generate the same sound at its own surroundings (or feed the same sound programmatically to the browser) and succeed in proving the co-location with the phone. The second type of *Sound-Danger* attack is the passive attack, in which the attacker waits for the phone to create a previously known sound, specifically a morning alarm, at an opportune moment and then tries to generate the same noise at its local terminal. The steps taken by the attacker to target *Sound-Proof* are shown in Figure 7.12.

7.4.3.1 Active Attacks

In the *Sound-Danger* active attack scenarios, we assume that the attacker has already compromised the web service that uses *Sound-Proof*. Since account databases on such servers typically store other forms of user's data (e.g., phone number for the password recovery purposes), hacking into the server reveals other information that can be a used in the active attacks. Such data if not stored on the main server is assumed to be obtained with data aggregation attack through other services that user has an account with (which probably does not use 2FA).

Based on the type of information that the attacker possesses, we define (and later implement and evaluate) the following attacks:

Ringtone Attack: In this attack, the attacker predicts the ringtone (or vibration sound) that the victim user has set for the received phone calls. The attacker makes a call to the victim using the knowledge it has obtained from compromised account database. At the same time, the attacker attempts to login by entering the previously leaked victim's credentials to the login page via its own login terminal. To mimic the victim's ambient noise (now the sound of the ringtone), attacker plays the same ringtone audio at its location next to the login terminal.

Information known to the attacker: Victim's username and password, victim's phone number and victim's ringtone.

Task of the attacker: Ring the victim's phone and create same sound around the local login terminal.

App Notification Attack: In this attack, the attacker predicts the voice/messaging application running on the victim's phone and tries to activate the notification tone or ring tone of the

application by communicating to the victim through the application. Since the user typically registers to many of the web services using phone number or user id, the attacker can contact the user on these applications either by their phone number (obtained by hacking the primary account database) or user id (possibly similar to the one registered with the primary service.

Examples of such applications are Google Voice, FaceTime, Skype, Facebook, WhatsApp and Viber. Calling or texting the user on these applications generates a default ringtone or notification tone that is known to the attacker and is usually not changed by the users. Hence, the attacker starts a login attempt to the primary service using the known credentials, and then contacts the user on any of the mentioned applications. At the same time, it plays the same ringtone or notification tone locally near the login terminal. The attacker would succeed since it regenerates the same ambient noise as the victim's phone locally (around the attacker's login terminal).

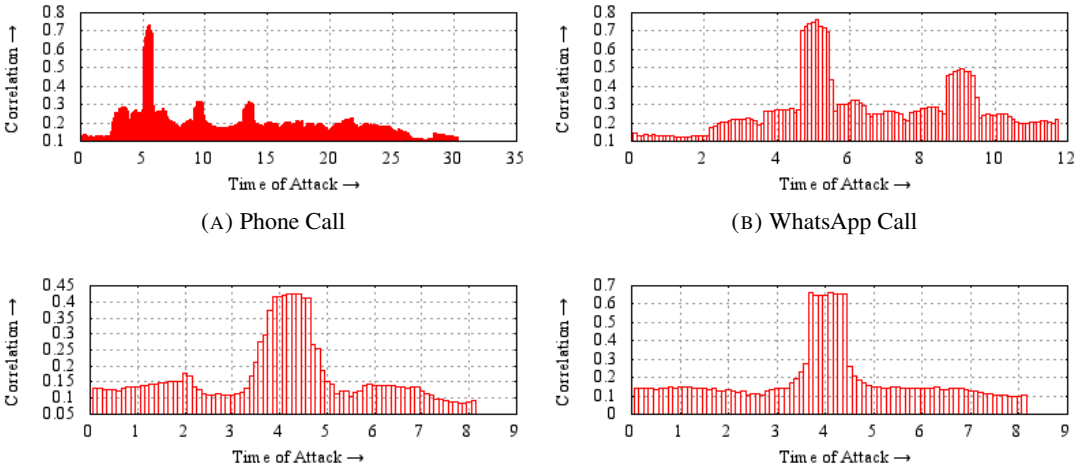
Information known to the attacker: Victim's username and password, victim's phone number, victim's installed application on the phone, victim's id with the application (same as phone number or primary username) and application ringtone.

Task of the attacker: Ring the victim's messaging application and create the same sound around the local login terminal.

Feasibility of Attacks: In all of our attacks above, the attacker would have to predict some information necessary to execute the attacks (e.g., the type of ringtone used by the victim). However, given predictable patterns and phone usage habits across users, this is not much of a problem for the attacker. In Section 7.4.7, we support these assumptions and claims made here, based on the data gathered from the participants in an online survey.

7.4.3.2 Passive Attacks

In the passive attack, the attacker predicts, or knows users' activity and launches the attack based on the knowledge it has from the users' profile gathered from the leaked database. Unlike the active attack, the attacker does not attempt to generate a sound at the user's side but only regenerates the same ambient sound that is supposed to be available at the user's side. Unlike the active attack, the passive attack does not alert the user by creating a sound, and hence can be repeatedly attempted without triggering suspicion.



(A) Phone Call (B) WhatsApp Call

(C) SMS Notification (D) Facebook Notification

FIGURE 7.13: Change in correlations when an attacker makes call or sends notification via different apps at different point of time. In Figures a and b, the ringtone at the victim's side starts playing at the 5th second while in Figures c and d the notification audio goes off at 4th second as depicted by highest correlation. There is no audio at the victim's side from the ringtone when the attacker plays the respective audio at 0th second. The correlation values are higher when the ringer is ringing compared to that when there is no ringer, i.e., before 2 sec in call.

Although there are different scenarios where an attacker can create a similar ambient noise to that at victim's side such as similar media attack (both attacker and victim are watching same media/TV channel, as also briefly considered in [30]), same event (both attacker and victim are attending a popular event), or similar vehicles sound (attacker knows when victim commutes and uses similar in-vehicle sound), we chose to exploit the ambient noise that is created by victim's phone itself such as alarms or morning reports. We believe that this attack has a higher chance of succeeding compared to other ambience-based passive attacks since the sound of the alarm of the phone will dominate the ambient sounds.

Alarm Attack: In this attack, the attacker knows the specific app which generates an audio at particular time of day such as the morning alarm. Attacker attempts to log in at a specific moment when such alarm is supposed to go off using the victim's credential. At the same moment, the attacker plays the alarm tone at its local login terminal to mimic the ambient noise around the victim's phone. In this attack apart from the victim's username and password, the victim's alarm time, alarm tone is also known to the attacker.

Information known to the attacker: Victim's username and password, victim's alarm ringtone and victim's timezone.

Task of the attacker: Create the same alarm sound around its local login terminal.

Feasibility of the Attack: Similar to the active attacks, the attacker would have to predict some information necessary to execute the attacks (e.g., the type of ringtone used by the victim and victim's timezone). However, given predictable patterns and phone usage habits across users, this is not challenging, as we demonstrate in Section 7.4.7 based on the results of an online survey.

7.4.3.3 Active vs. Passive Attacks

We introduced passive and active attacks, each of which has their own merits. With the passive alarm attack, it is very likely that the victim user would not notice the ongoing malicious login attempt. Therefore, the attacker might be able to repeat the attack repeatedly until it is successful. In case of active attacks, the sounds generated by the attacker on the user's phone (e.g., a phone ringing tone) could notify the user and seek her attention. However, only a few seconds of audio is enough for the *Sound-Proof* system to verify the co-presence of the phone and the terminal. Therefore, by the time the user attends to the phone (e.g., to pick up the call) and even when the user notices the malicious login attempt, the attack would have already succeeded and the user's account might have already been compromised.

Although *Sound-Proof* logs the login attempts on the device (as suggested in [30]), the users may leave their phones unattended, in purses or bags, might not be concerned about security or be diligent enough to the extent that they review the logs carefully and frequently. Extensive research literature in user-centered security shows that users may not pay attention to security notifications or heed security warnings and messages (e.g., [19, 190]). Moreover, relying upon the users to detect such attacks will break the "zero-effort" property of *Sound-Proof*. Furthermore, even if the logs were read and understood by the users, the attack may have already succeeded by the time suspicious activity is noticed.

7.4.4 Analysis

In this section, we show the correlation analysis of different attacks using *Sound-Danger* introduced in Section 7.4.3. In other words, we test the rate at which the attack samples (corresponding to attacker's browser and victim's phone) will be accepted as valid login attempts by our implementation of the *Sound-Proof* app. In our analysis, we used Samsung Galaxy S5 from Verizon as the victim's smartphone along with Google Chrome browser in MacBook Air (mid 2012) as

attacker's terminal to perform the attack. The attacker makes calls or sends notification from LG G3 from Verizon or a computer to create an audio it desires at the victim's side.

To perform the attacks described in Section 7.4.3, the attacker follows the steps as illustrated in Figure 7.12. The attacker who performs such attacks tries to generate or predict a similar audio at the victim's side while it logs into the victim's account with the victim's credentials. The attacker has full control over the computer/browser that it is using. However, the attacker does not have any direct control over the victim's smartphone/app.

7.4.5 Ringtone and App Notification Attacks

To test our ringtone and app notification based active attacks, as described in Section 7.4.3, we use phone ringtone (call/SMS) as well as various other ringtones from some of the popular apps in Google Play Store, such as Facebook, WhatsApp Messenger, Viber, and Skype. We use default ringer of the app/phone call. Although some of the victims may have customized the ringtone for phone call or any other app, some of these apps do not allow users to customize the ringtone for calls or notifications. The primary difference between call and notification attack is that the ringtone audio is played longer for the call than it is for the notification.

Since the attacker does not have any direct control over the victim's phone, the attacker faces challenges due to two types of delays: (1) *Sound-Proof* recording delay, and (2) call/notification delay. Due to *Sound-Proof* recording delay, the attacker cannot perfectly guess at what time instance *Sound-Proof* starts recording audio from the victim's phone for the purpose of login. And, due to the call/notification delay, when the attacker makes call or sends notification to the victim's phone, the attacker also cannot make a perfect estimate to when the ringer sound will be played at the victim's side. To estimate these two delays, the attacker can run an experiment by making calls or sending notifications to itself and monitor the delays. Based on this, the attacker tries to synchronize the ringer being played at both sides as much as possible. We run and analyze the attacks assuming that the attacker knows when *Sound-Proof* starts recording at its end. This is a valid assumption since the attacker fully controls its terminal. Because of the delays mentioned above, during the actual attack, *Sound-Proof* may start recording either before or after the ringer goes off at the victim's side. We set forth to analyze how the correlation values change when the victim's ringer goes off at different points of time.

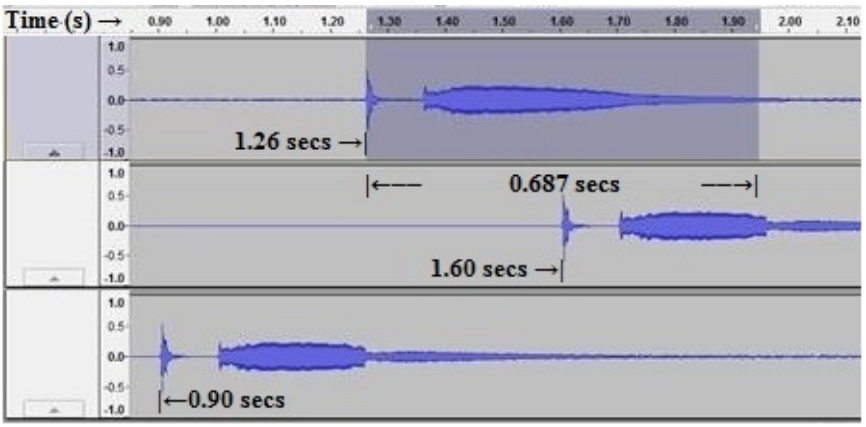


FIGURE 7.14: Analyzing Facebook notification audio sample (∼687ms long). The first audio sample (top) represents the audio played by the attacker. The second audio sample (middle) and the third audio sample (bottom) represent the audio recorded at victim's side when the notification audio rings after or before the attacker's side, respectively.

Attack Analysis: *Sound-Proof* compares 3-second long audio samples, as discussed in Section 7.4.2.3. Let us say the victim receives call/notification by an attacker at the n^{th} second. The attacker starts the authentication at the t^{th} second. This is when *Sound-Proof* starts recording at both sides. If *Sound-Proof* starts recording at t such that $t < n - 3$, *Sound-Proof* will not record any audio component due to the ringer, and hence, there will be low correlation between the audio pair. When $t = n$, the correlation will be the highest as the attacker has fully synchronized the audio at its side with that at the victim's side. This pattern exhibited by the correlation values can be visualized in Figure 7.13. Here, the ringer for call goes off at the 5th second (Figures 7.13a and 7.13b) while the sound of the notification goes off at the 4th second (Figures 7.13c and 7.13d). Hence, the correlation is very low prior to the first 2 seconds. Now, when $t > n$, the correlation should drop as the two audio samples are not synchronized. However, the correlation after the ringer has started ringing ($t > n$) is higher than that when there was no audio at all ($t < (n - 3)$). Therefore, we know that the correlation increases and is reasonably high as long as there is some matching audio even if the two audio samples are not at synchronized. The increase in the correlation values after $t > (n - 3)$ proves this pattern as depicted in Figures 7.13a and 7.13b.

To further analyze this property, we choose Facebook notification attack instead of call attack for simplicity. We use Audacity[1] to analyze why the correlation in Figure 7.13d increases at $t = 3.4$ second and drops only after $t = 4.6$ second while the exact match occurs at 4th second as illustrated in Figure 7.14. The first audio signal in Figure 7.14 represents the audio signal that the

[1] http://www.audacityteam.org/

attacker plays at its terminal which is 3 second long. The audio due to Facebook notification has audible signal of length 687ms which starts at 1.26th second. The second audio signal represents the audio recorded by the app at victim's terminal for $t = 4.6$ second where the notification ringtone goes off at 1.60th second. The third audio signal represents the audio recorded by the app at victim's terminal for $t = 3.4$ second where the notification ringtone goes off at 0.90th second in the 3 second long audio. From these three audio samples, we can see that whenever there is an overlap between the audible sounds, the correlation rises despite the time lag bound to 150ms. This is because when there is some audio, the correlation values from some of the 1/3-octave bands out of 20 bands increase, increasing the overall average correlation value.

Real Attacks and Success Rates: After analyzing the attack methodologies mentioned above, we set forth to perform the real attacks. To perform such attacks, the attacker needs to: (1) make a call/send a message via different apps to a victim's smartphone, (2) log in from a browser on a terminal which it fully controls, and (3) play the ringtone or a notification sound that the victim device may generate due to attacker's call/message. At the attacker's end, we used an LG G3 phone and a MacBook Air and, at the victim's end, we used a Samsung Galaxy S5 phone. The attacker first observed how long it takes for another device to ring in each different app when it makes a call to the corresponding apps. Then, the attacker made calls to the victim's device from those apps. The attacker tried to synchronize the ringtone played when it logs in from the Google Chrome browser on MacBook Air.

We tested different attacks (active calls and notifications) against our implementation of the *Sound-Proof* system, and collected the audio samples stored in the victim's smartphone and the audio uploaded to the server from the attacker's browser. The success rates for our attacks with the correlation threshold $T_c = 0.1524$ for different types of attacks are shown in Table 7.3. We can see that many of our attacks were highly successful, including WhatsApp, Facebook and Viber calling, Viber notification and alarm. We also used vibration based attack where the noise is produced by a vibration of the phone instead of the phone playing a ringtone during a call. We placed the phone in different location such as on desk, inside a pocket, inside a bag, and in hand to see if the audio detected by the phone for such placements of the phone affects the attack success rate. Table 7.3 also shows the attack success rate when the threshold was increased to $T_c = 0.18$ and $T_c = 0.2$. When the correlation threshold is increased, the attack success rate decreases slightly as expected (although many attacks are still highly successful).

TABLE 7.3: Success rate of different types of attacks with respect to different correlation thresholds. Highlighted cells represent attack with success rate at least 90%.

	Attack Type	Tc= 0.1524	Tc = 0.18	Tc = 0.2
Active Call	Phone Call	81.82%	72.73%	63.64%
	Viber	100.00%	100.00%	90.00%
	WhatsApp	100.00%	100.00%	100.00%
	Facebook	100.00%	100.00%	72.73%
	Skype	41.67%	25.00%	16.67%
	Facetime	92.86%	57.14%	42.86%
	Vibration	85.42%	81.25%	72.92%
Notification	SMS	64.71%	35.29%	17.65%
	Skype	85.71%	52.38%	19.05%
	WhatsApp	66.67%	33.33%	25.00%
	Viber	100.00%	92.86%	85.71%
Passive	Alarm	100.00%	90.00%	80.00%

We note that increasing the threshold would make the attacks a little harder but at the expense of usability since even legitimate user may be prevented from logging in more frequently. Further experiments revealed that the attack success rate did not change even when the victim device was placed in front of a television with high volume. This confirmed our hypothesis that the sounds of the phone will dominate the sounds of the ambient surroundings.

7.4.6 Passive Alarm Attack

As described in Section 7.4.3.2, the attacker could execute the alarm attack at a specific time of the day (assuming the attacker knows when the alarm will go off at the victim's phone). Here, the attacker may know the timezone of the victim (through leaked password databases). Since the attacker has control over the browser and the device it is using at its end, the attacker can change its own timezone to be synchronized with that of the victims. To simulate this setting, we played LG G5's default alarm in front of the browser while the phone was set to create the alarm at a fixed time instance. Since both phones played the same alarm simultaneously at different ends, we achieved high correlation for the alarm attack reflecting to 100.00% success rate with $T_c = 0.1524$. The success rate decreased when we increase the correlation threshold, but we could still achieve 80% success rate. The result for this attack for different threshold is

summarized in Table 7.3 (last row).

7.4.7 Population Statistics

To support the claims and assumptions made in Section 7.4.3, we conducted a survey by recruiting Amazon Mechanical Turk workers. The participation in the study was strictly voluntary and participants could opt out of the study at any time. The survey took only about 10 minutes for each participant, for which they were compensated $0.7. In this section, we discuss the design and results from this survey.

7.4.7.1 Study Design

To better inform the design and execution of our attacks in the real-world, we asked the participants to answer several questions about their smartphone usage, including their habits of using the smartphones, the smartphone applications they use, and the ringtone and the notification sound they set or prefer to use. Following is the summary of the results for the set of questions we asked during the survey.

Demographic Information: We asked the participants about their gender, age, education, industry or field they belong to, country of residence, and their general computer knowledge. The demographic information of the participants is shown in Appendix C Table 1.

Applications: We asked the participants about the applications installed and used on the phone, particularly those that generate a ringtone or a notification sound (e.g., Google Voice, FaceTime, Skype, Viber, Tango, ooVoo, LINE, WhatsApp, Telegram Messenger, Facebook, Phone, Text Message, Alarm Clock, and Calendar). Such applications are the primary target of the attacker in our *Sound-Danger* system.

Notifications and Sounds: We queried about the type of ringtone (e.g., default, vibrate, silent), and notification tones that the participants set for their applications in different situations and time of the day (e.g., while at work or asleep). If a particular popular ringtone is set often, the attacker can possibly attack many participants with our ringtone attack.

7.4.7.2 Study Results

General and Technical Background: We recruited 113 Amazon Mechanical Turk workers. Almost equal number of male (50.82%) and female (49.18%) users participated in the study. Although we did not set any geographical restriction, majority of the participants were from United States (73%) and India (21%). The participants were falling in the age group 18 to 65, precisely 18-24 (12.30%), 25-34 (54.92%), 35-44 (22.95%), 45-54 (8.20%), and 55-64 (1.64%). The participants had high school (6.56%), college degree (25.41%), Associate degree (7.38%), Bachelor's degree (40%), Master's degree (1.72%), and Doctorate degree (2.46%). The participants were from different industrial background including: education, technical services, marketing, information technology, health care, and financial services. The demographic information shows that the survey covers a representative sample of real-world users.

The participants seem to have a reasonable general computer background as they ranked their general computer skill mostly as good (40%) and excellent (45%). We asked the users about their choice of username and password. The result shows that many users reuse the same username and/or password over multiple services. We will discuss in Section 7.5.3 that reusing the username may help an attacker who has compromised the web-service and knows the username of the victim to more successfully perform the ringing attack on an application (e.g., Skype) with the same username.

Habits of Using Smartphone and Apps: All of our participants said they have a smartphone. 80% of the participants said they carry their phone all the time and they have their phone connected to Internet always or most of the time. Most of the participants had voice, text and data services activated in their plan. Apple iPhone with over 39% and Samsung with 27% were the two most popular phone brands (other popular brands were: LG, Motorola, and HTC). The information obtained from this part of the survey shows that launching the introduced attack would be feasible, since many of the participants have a smartphone with voice/data/text plans that can be used by the attacker in an active attack.

All the participants in the survey said they frequently use phone calling and text messaging applications. The most popular instant messaging applications installed on participants' phones were: Facebook, Skype, Google Voice, FaceTime, WhatsApp and Viber. Application with higher popularity (such as Skype) especially those for which people use default ringing tone are the most

TABLE 7.4: Popularity of instant messaging applications and the default ringtones for each app among participants.

Application	Popularity	Default Ringtone
Facebook	87%	67%
Skype	55%	63%
Google Voice	41%	66%
FaceTime	41%	83%
WhatsApp	36%	66%
Viber	22%	68%

attractive target applications for *Sound-Danger*. Table 7.4 summarizes the fraction of participants who use a given app and the default ringtone popularity among the participants who use the app.

We asked the participants about the kind of sounds they use for each application on their smartphone. The more predictable the ringtone is, the more successful the active ringtone attack would be in our *Sound-Danger* system. For the phone calls and text messaging, vibration and default ringtone are the most popular settings, silent and custom ringtone being less popular. It seems people tend to set the vibration at work, and set the default ringtone while at home. While still some participants tend to set custom ringtone for their phone calls and text messaging, custom ringtone is not that popular for instant messaging applications. More than half of the participants set the default ringtone for the instant messaging applications. Vibration is the second most popular setting for the instant messaging applications. The participants said that during a day they keep their phone on ringing mode or on vibrate mode about half the time, while they set it on silent mode only once in a while. These measures show that the *Sound-Danger* active attacks that target users by calling them on some popular instant messaging applications have a higher chance to succeed. Apart from the instant messaging application, launching a passive attack by playing the default alarm tone seems quite feasible. About half of the participants set the default alarm tone that the attacker can play locally at certain time of the day to mimic the sounds of the victim's phone. Table 7.5 summarizes the popular ringing setting of the two most popular phone brands, namely iPhone and Samsung among the participants, in three common situations: at home, at work and while asleep.

7.4.8 Strategy

The survey results in Section 7.4.7 help us devise real-world attack strategies and estimate the corresponding attack success rates. The effective success rate of the attack, for an adversary who

TABLE 7.5: Popular ringtone setting for Samsung and iPhone.

Phone Brand	Location	Ringing Setting			
		Silent	Vibrate	Default	Custom
Apple	At home	5%	43%	45%	30%
	At work	20%	64%	16%	11%
	Asleep	25%	41%	30%	18%
Samsung	At home	13%	13%	40%	37%
	At work	20%	50%	13%	20%
	Asleep	27%	20%	37%	20%

does not know (but can guess) the user's behavior and their habits of using the phone, can be calculated by multiplying the usage probabilities we obtained from the survey by the success rates reported in our correlation analysis of the attacks (Section 7.4.4).

Preliminaries: Let us call the success rate of the correlation-based attack, as we presented in Table 7.3, as x (e.g., success rate of a ringing attack for an attacker who *knows* the victim's ringtone). Then, such attack would succeed with the probability $p = device \times state \times x$, where $device$ denotes the probability of owning a specific type of device (e.g., Apple's iPhone) and $state$ denotes the probability of the phone being in a particular state (e.g., default ringtone at home). Note that the attacker can make multiple login attempts at a given point of time to increase the chances of success. In this case, for k iterations of login, the "Iterative success rate" (termed $Itt(k)$), for a particular attack (with success rate x) can be calculated as:

$$Itt(k) = 1 - (1 - x)^k \tag{7.3}$$

The attacker repeats the above attack, with different attack variations, in multiple rounds. During each attack round i, the attacker performs the attack with k iterations targeting the remaining uncompromised users from round $i - 1$ (i.e., UN_{i-1}). Initially, no users have been compromised (i.e., $UN_0 = 100\%$). Thus, the "Effective attack success rate" $Eff_i(k)$, in round i with k iterations, which represents the fraction of users the attacker has compromised in round i, is given by:

$$Eff_i(k) = device \times state \times Itt(k) \times UN_{i-1} \tag{7.4}$$

Note that $device$, $state$ and $Itt(k)$ do not depend on the attack round per se, but rather they depend on the type of attack performed, i.e., these values remain the same even when the order

in which the attack rounds are executed is changed. In contrast, Eff and UN depend upon the attack round.

A Concrete Sample Strategy: The attacker can devise a strategy to compromise the maximum number of victims by choosing any subset of the attack variations discussed in Section 7.4.3 and launching them in a particular order. In the rest of this section, we show a sample attack strategy based on our online survey results (Section 7.4.7) and a subset of our attacks (Table 7.3) in a specific order to compromise about 83% of user accounts in a total period of less than a day. Our attack strategy is summarized in Table 7.6. This is only a sample strategy for demonstrating the overall effectiveness of our attack. In practice, a real-world attacker can devise other strategies to maximize the impact of the attack based on the target user population under question.

As in our *Sound-Danger* attack model, we start with the assumption that the attacker has already obtained username, password, phone number and timezone of each of the target users by compromising a server and is trying to login to the victim user's account by: (1) entering the first authentication factor (username and the password), and (2) attacking the *Sound-Proof* application to prove the possession of the second factor (the phone). We also assume that the server throttles login attempt after three login trials to prevent a login brute force attack (a common practice employed by many web services). Therefore, we limit the attacker's login attempts to three, setting $k = 3$ in our attack strategy throughout. We test our attack strategy against our implementation of *Sound-Proof* with the threshold $T_c = 0.1524$.

We start with UN_0 equal to 100.00% of the user accounts (no account has yet been compromised). Since many of the users in our survey indicated that they keep their phones in vibration mode at work and this attack works irrespective of the device type, we attack such users in the first round. We know that 57.00% of the users have their device in vibration mode at work (Table 7.5), and the attack success rate x for the vibration attack is 85.42% (Table 7.3). This yields the iterative attack success rate (Itt) in this round to be 99.69%. Hence, the effective attack success rate (Eff_1) for this attack is $device \times state \times Itt \times UN_0$ = $100.00\% \times 57.00\% \times 99.69\% \times 100.00\%$ = 56.82%. This means that we can compromise 56.82% of the users with the vibration attack by just making three phone calls during work hours. The calculations and success rates for this round of the attack are summarized in Table 7.6, row 1. Compromising about 57% accounts in just one round, for example, right after the password database was leaked, would be a significant

TABLE 7.6: ***Sound-Danger* Attack Strategy ($T_c = 0.1524$):** The percentage of compromised users at the beginning of each attack round i is denoted as CN_i. Effective attack success rate (Eff_i) of the attack at each round i depends upon the particular type of device victim is using ($device$), the particular state of the device the attack is targeting ($state$), the iterative success rate for a number of login attempts (k) the attack will be repeated for (Itt), and the percentage of currently uncompromised users the attack is targeted towards (UN_i), as shown in Equation 7.4. In our calculations, $k = 3$ all throughout. The last column ($CN_i = 1 - UN_i$) shows the percentage of compromised users (CN) after each attack round. Before the start of the attack, i.e., at round 0, $CN_0 = 0\%$ ($UN_0 = 100\%$). When the attack applies to all devices (e.g., vibrational attacks), the device probability ($device$) is 100%, and when it applies to specific device types, iPhone and Samsung, the device probabilities are 39% and 27%, respectively. The highlighted cell represents the percentage of user accounts successfully compromised in eight rounds, which may finish in less than a day.

Attack Round (i)	**Attack Description**	**Probabilities**				$Eff_i(k = 3)$	CN_i ($1 - UN_i$)
		$device$	$state$	x	$Itt(k = 3)$		
1	Vibration at work	100.00%	57.00%	85.40%	99.69%	56.82%	56.82%
2	iPhone call at work	39.00%	16.00%	81.80%	99.40%	2.68%	59.50%
3	Samsung call at work	27.00%	13.00%	81.80%	99.40%	1.41%	60.19%
4	Vibrate at night	100.00%	30.00%	85.40%	99.69%	11.69%	72.60%
5	iPhone call at night	39.00%	30.00%	81.80%	99.40%	3.19%	75.79%
6	Samsung call at night	27.00%	37.00%	81.80%	99.40%	2.40%	78.19%
7	iPhone alarm	39.00%	50.00%	100.00%	100.00%	3.19%	81.38%
8	Samsung alarm	27.00%	50.00%	100.00%	100.00%	1.88%	83.27%

threat. The attacker may stop here, or continue to the next round in order to compromise more user accounts.

To attack the rest of the uncompromised users, $UN_1 = 43.18\%$, after the first round, we choose the next popular $device$ and $state$ combination based on Table 7.5. Through our survey, we observed that 39.00% of the users have iPhone. From Table 7.5, we know that 16.00% of the users keep their device under default ringtone at work. Therefore, in our strategy, the second attack round would involve the default ringtone call for the "iPhone at work" users, since we can have the maximum impact with this approach. From Table 7.3, the attack success rate for default ringtone x is 81.80%, which amounts to Itt equal to 99.40%. The effective attack success rate (Eff_2) for this round of the attack is $device \times state \times Itt \times UN_1$ = $39.00\% \times 50.00\% \times 16.00\% \times 43.18\%$ = 2.68%. Hence, after the second round of the attack, we have successfully compromised 59.50% of the user population (56.82% in the first round plus 2.68% in the second round). This attack round is summarized in Table 7.6, row 2. The attacker may continue for the next few rounds in a similar fashion, as shown in Table 7.6. As we can see, after the eighth round, our *Sound-Danger* system will have compromised a total of over 83% of the user accounts.

Since we started the attack "at work", and end it in the morning time (with the alarm attack),

it is fair to say that all the rounds of the attacks will have finished over a period of less than a day. A persistent attacker may continue further the next day, perhaps trying other attacks at different points of time, and may gradually compromise almost all user accounts in few days.

Finally, we re-calculated the success rate for the above attack strategy against our implementation of *Sound-Proof* with threshold values higher than $T_c = 0.1524$. We found that even when we increase T_c, our attack strategy is still successful, by compromising 82.60% of the users with $T_c = 0.18$, and 81.52% with $T_c = 0.2$. This shows that our attack strategy remains robust to increased thresholdization, highlighting the overall vulnerability of *Sound-Proof*.

7.4.9 *Sound-Proof* Demo Analysis

Karapanos et al. [30] have deployed Sound-Proof demo app[2] and released apps for Android[3] and Apple[4]. We set forth to analyze how the demo app (version 1.6 on Android) performs against our attacks compared to our implementation of *Sound-Proof*. We observed that the demo app uses higher value of correlation threshold ($T_c = 0.2$) than the one reported in the paper [30] ($T_c = 0.13$). As we only had access to the app binary (and not source code), we could not directly figure out the values for other parameters deployed in the demo app.

Our evaluation showed that FNR of the demo app (benign setting) when the phone was kept beside the computer was quite high, at 27.91%. When the phone was kept inside a bag/purse, FNR increased to 50%. Compared to the results reported in [30], the higher FNR might have been due to the use of higher value of correlation threshold (and possibly other tighter parameters) than that reported in the paper.

We then attacked the demo app with one active attack and one passive attack. In the active attack trials, we made calls to the victim using WhatsApp. In the passive attack trials, we tested the demo app against alarm audio using two different devices (victim uses Samsung Galaxy S5 while attacker uses LG G3). FPR for the active attacks was found to be 38.46% while that for the passive attacks was 64.71%. The attack success rate on the demo app is less than that on our implementation of *Sound-Proof*. This may again be due to the fact that the demo app uses higher value of T_c (and possibly other stricter parameters). As shown in Table 7.3, the success rate for different attacks against our implementation of *Sound-Proof* decreased when Tc was

[2] http://sound-proof.ch/
[3] https://play.google.com/store/apps/details?id=ch.soundproof
[4] https://itunes.apple.com/us/app/*Sound-Proof*/id1069858990

increased. Moreover, we noticed that the average correlation provided by the demo app was relatively high (e.g., the average correlation for the alarm clock is 0.29 with 0.16 as minimum correlation). Hence, if the demo app had used the T_c reported in the paper [30] ($T_c = 0.13$), the alarm attacks would have been 100.00% successful.

Furthermore, we analyzed for which parameters in our implementation of *Sound-Proof* will produce similar results to that by *Sound-Proof* demo app. To this end, we recorded audio from two devices simultaneously using both apps. We collected 30 audio instances and logged the correlation score from the demo app. We calculated the correlation results for different length of audio recorded (3s, 4s, 5s, and 6s) and for different threshold values (0.1524, 0.18 and 0.2). We found that when we compared 6 second long audio with $T_c = 0.2$, the scores from our app and the demo app had the maximum correlation (we used the alternative computation formula for Pearson's r [191] to calculate the correlation level). This suggests that the demo app is using 6-second long audio snippets rather than 3-second.

Overall, this analysis suggests that the *Sound-Proof* demo app uses much stricter parameters, which resulted in very high FNRs. Notably, even with this parameterization resulting in very low usability (high FNR), the *Sound-Danger* attack can still be successful against the demo app, further validating its feasibility as a viable real-work attack against *Sound-Proof*.

7.4.10 Summary

Zero-effort two-factor authentication is a compelling notion that may push two-factor authentication towards wide-scale adoption on the web. The idea of using ambient sounds to verify the user's possession of (or proximity to) the authentication token (phone) is intriguing, which aims to remove the human user from the loop of authentication (except of password entry). In this section, we demonstrated that the ambient audio approach to zero-effort two-factor authentication is highly susceptible to a remote attack that makes the phone record its own predictable sounds in the form of a ringer, app-based notifications and alarms. Since these sounds are predictable, the attacker can generate the same (highly correlated) sounds at the browser's end under its full control and succeed in logging into the user's account. Further, by proactively collecting statistics about a given population (not a specific user), the attacker can devise a strategy that can allow the attacker to compromise a large fraction of users' accounts in a short span of time.

Since the attack exploits the sounds of the phones, one obvious defense would be to deliberately mute the phone at the time the login takes place. However, this may reduce user experience since important calls or notifications may be missed or delayed. Further work is necessary to assess this and other potential mitigation strategies we presented in this work. Overall, our work serves to call the security of a prominent, deployment-ready zero-effort two-factor authentication scheme into question, highlight the tension between the security and usability of two-factor authentication and raise a demand for utmost care when attempting to design authentication approaches transparent to the human user.

7.5 Potential Mitigation

7.5.1 Defense against *SMASheD*

We suggest the following potential mitigation strategies to defend against the adversarial applications of *SMASheD* (Section 7.2.1).

First, we believe that it is important to raise people's awareness of the possible security risks associated with installing services through the ADB shell. Second, we suggest following the permission models of Android for native services that are executed through the ADB shell. In the current model, any native service that starts through the ADB shell is granted all the permissions that the shell has without notifying the user. These permissions include accessing logs, frame buffer, motion, position, environmental, and user input sensors. An attacker may not reveal all the resources that the service is accessing. For example, the attacker could publish a service as a snapshot service while injecting code that accesses sensor files as well. This may be prevented if the service is only granted permissions after informing the user. Third, we suggest enforcing security policies for the communication between processes running on the device through sockets. We recommend that Android monitors the open sockets on the device and the apps that are accessing those sockets. Whenever an unusual communication is detected, Android should at least inform the user. Whether or not users would pay attention to such notifications is an independent concern. However, we believe that the potential risks should be conveyed to the users.

Although these strategies may not fully prevent the attacks, they may help reduce the impact of the underlying vulnerability.

7.5.2 Defense against Environment Manipulating Adversary

7.5.2.1 Using Decisions-Fusion

In Section 7.3.3.5, on analysis of an audio-radio-physical system, we showed that decisions-fusion reduces attack success rates in cases where the minority of the sensors are manipulated. However, this may come at the cost of higher FNR which represents the usability of co-presence systems.

Decisions-fusion, equal voting from single sensors improves security when individual sensors perform well. However, it increases the attack success rate for weak sensors as they equally contribute to the voting. For example, in the context of the audio-radio-physical system, attacking weak sensors such as H or G brings relatively high success rate compared to features-fusion. To avoid this issue, we introduced decisions-fusion, weighted voting which are derived from the strength of attack resilience. Decisions-fusion, weighted voting in general further reduces FPR in case strong sensors are attacked. However, at the same time, FPR increases for weak sensors in such cases. For example, in (Audio-Radio system, single modality attack), Table 7.2, decisions-fusion, weighted voting reduces FPR of Au, W, but increases FPR of B. Decisions-fusion, equal voting from subsets of sensors reduces the FPR in general especially when dominant sensors are controlled by the attacker.

As can be seen from Table 7.2, weighted voting results in a significant improvement in some cases. For example, FNR decreases in Audio- Radio, zero modality and Physical, zero modality system from 12.0% to 2.7% and from 14.5% to 7.8%, respectively, while FPR decreases in case of single modality in Audio-Radio, when an attacker manipulates Au or W, or in Audio-Radio-Physical system, when an attacker manipulates B only. There is a significant improvement in FPR, when an attacker manipulates two modalities {Au, W } in (Audio-Radio system, multi modality). Therefore, we recommend using the weighted voting variant to defeat such context manipulating adversary.

7.5.2.2 Other Potential Countermeasures

Typically, during the authentication/deauthentication process, the prover moves nearer to/farther away from the verifier. In this case, the radio signals changes gradually, i.e., if prover and verifier move towards APs, then new APs will be shown, or their signal strengths will continuously grow, while if they move further away from APs, their strengths will decrease or the APs will not be

visible at all. If the verifier or prover device detects much more APs (or Bluetooth devices) nearby all of a sudden, it probably indicates a radio manipulation attack. The system can be made aware of such situations.

We noticed that when the verifier is in an environment which has high frequency noise, an attacker tends to fail with audio streaming. This can be used to design an active defense mechanism such that whenever audio contextual information is requested, the verifier can emit a high frequency audio. This audio signal can be for a short duration, and does not need to be loud (not high amplitude). As a result, the chances of attacker succeeding in a relay attack could be reduced.

When an authentication request has been initiated or finished, the user can be passively notified at both devices. Passive notification can be a flashing of LED light or beep on the prover device (key or phone). Hence, even if the verifier device is left unattended, user may notice on the prover device that someone is trying to authenticate the verifier, or has authenticated on user's behalf. Whether or not users would actually pay attention to such notifications should be subject to scrutiny. It may help reduce the risk of context-manipulation relay attacks.

7.5.3 Defense against *Sound-Danger*

A natural defense against our attacks would be to disable the 2FA system in the scenario when a call or a notification is received (and the corresponding sounds are played by the phone), or when an alarm is triggered. Alternatively, the calls, notifications or alarms could be disabled when the 2FA login takes place. However, such mitigation will prevent the user from receiving calls/notifications or setting alarms while logging into *Sound-Proof* enabled accounts, and could possibly degrade the usability of the phone system.

Another possible defense is to reduce the probability of guessing the phone sounds. This defense relies on the user to prevent the attack by picking ringtones that are difficult for the attacker to predict and possibly changing them frequently in order to stop the attacker from attempting an exhaustive search. The analysis of the user survey in Section 7.4.7 shows that many users set the default ringtone for the instant messaging applications (e.g., Skype or Facebook) that makes it easier for an active attacker to predict the sound.

Similar to the custom ringtone, combination of sounds and/or vibration is a possible defense mechanism. During our attack analysis, we noticed that the correlation reduces to below the

threshold value when the notification sound is mixed with vibration and the attacker plays only the notification ringtone at its side. Simply combining the ringtone and the vibration at the attacker's side to mimic the audio at the victim's side does not work for the attacker as we noticed that the vibration started at different point when the ringtone starts playing at the victim's phone. Hence, for a successful attack, the combination should be precisely synced with the one at the victim's side (with the occurrence of vibration at the exact position in the audio), which seems unlikely.

Any of the above defenses possibly introduce certain usability issues. For example, users might prefer default notification tone over custom ringtone. Or reusing the username requires the user to remember multiple usernames associated with each account. Further study is required to understand how these possible usability issues may impact the user experience of the phone system while strengthening the security of *Sound-Proof* in the face of *Sound-Danger*.

CHAPTER 8

CLOSING REMARKS

In this chapter, we provide the conclusions from our dissertation work. We further discuss some of the key insights that we gained during our work, and point to a few research directions that we have identified to be pursued in future work.

8.1 Summary

From this dissertation work, we conclude that it is possible to effectively use different kinds of context to provide security for authorization and authentication for mobile devices. We presented our work to enhance the security of mobile devices against insider and outsider attacks based on context detection. Our work shows that context can be effectively detected explicitly or implicitly from the users, or from the environment, using different sensors embedded within the smart devices and off-the-shelf companion devices. Moreover, we showed that these contexts can be used for user-transparent biometric authentication using the inertial sensors. We successfully authenticated users with their tapping-biometrics and walk-biometrics. We also presented the vulnerabilities associated with the context detection system and exposed our analysis against the systems using context in the face of context manipulating adversary. First, we showed different application failures when the context can be manipulated from inside the device in the form of context manipulating malware (*SMASheD*), and then we showed co-presence detection system can fail when the context is manipulated from outside the device in the form of context manipulating adversary. In a similar vein, we presented an attack on the state-of-the-art 2FA defense *Sound-Proof*. We further provided mitigation strategies that could be used to undermine even such sophisticated attackers, thereby strengthening the security of the proposed systems.

8.2 Discussion

8.2.1 Adherence to Design Criteria

Lightweightness: One of the design goals of our systems is to be light-weight as high power consumption may reduce the effectiveness of the system. Since context detection process in our approaches lasts for no more than a few seconds, our approach is quite power efficient and light-weight as shown in Sections 5.2.4.4. The sensors are turned off while our process is running in the background as a service. The sensors are activated only when context detection process is required. Each context detection process requires sensor data for short duration and stops sensor recording after that. The detection approaches themselves are lightweight and require negligible amount of power.

Efficiency: Our context detection approaches take no more than a few seconds to run to provide improved security and increased efficiency. Our approaches rely upon machine learning techniques where the training of the data may take some time to build a classifier while the testing of a particular instance of data is quick and efficient. For the NFC tap authentication, one second long sensor data is enough to authenticate users while for *WUZIA*, users get authenticated as they walk towards the device requesting the authentication. Even for the co-presence detection between two devices, Karapanos et al. [30] have already shown that 3 second audio data is enough to verify the co-presence with very high accuracy, which is in line with our work.

Robustness: From our results, all of our context detection systems yield very high F-measure with very low FNR in benign setting and very low FPR against passive attackers. We also presented the results for active attacks for the context enhanced authentication systems where we analyzed active attackers who try to impersonate or replicate the context. Our results show that for such attackers, our systems can be resistant in most cases with very low FPR. Our *WUZIA* system is even tolerant to the sophisticated treadmill-based attacks as described in Section 5.3.5.2 since our system uses multiple uncorrelated sensor features.

We showed that environmental attacks can be manipulated by unsophisticated attackers in Section 7.3, which would increase FPR. However, we showed that such attacks can be better thwarted by implementing the context detection systems with multiple sensors and utilizing

weighted decision fusion technique. We also provided other mitigation strategies that can further lower the impact of even such sophisticated attacks.

Transparency and Consistency of Usage Model: During the context detection using implicit and environmental context, the user is not aware of an additional layer of security that is authenticating the users/transactions. User performs the transactions (NFC or $\mathcal{Z}IA$) without the change in usage model or authorizes an app requesting permission for sensitive resources/services. This entire process of authentication is transparent to the legitimate user. Even for the explicit context, our system requires the user to perform easy hand wave gestures. Hence, our systems satisfy the design goal of being transparent and having a consistent usage model to existing systems.

8.2.2 Fallback

Our detection approach has very low FNR with high recall. When we used sensors embedded on the mobile devices to extract implicit and explicit gestures using motion and inertial sensors, our system could identify the context with very low errors. However, there might be certain situations where, because of the user or device orientation, the activity performed by the user is so distinctive that the detection mechanism may mis-classify the gesture. For instance, the user might be on a moving vehicle or lying in specific posture where the sensor data may be completely different compared to when the user is in a normal position. A user may be injured, stressed, sick, or carrying the phone in a different way other than that during the training phase, which may significantly alter the motion of the mobile devices during the testing phase. In these scenarios, when the context detection mechanism fails, there is a need to fall back to allow the user to access the desired resource/service. This can be solved either by prompting the user to press a "Yes/No" button, or by asking the user for the explicit gestures such as hand-waving or rubbing, as proposed in [35, 112]. In situations where users are not able to make the gestures, for example under extreme emergency, a voice command could be used. Or, we can fallback to traditional password/key based approach for authentication/authorization.

8.2.3 Classifier Limitation

In our experiments, we recruited participants from two universities and different locations (U.S.A. and Finland). Even though we covered diverse population and showed that our machine learning

classifier identifies the context with very high accuracy, we do not provide the confidence of the classifier to identify/distinguish new population. Our experiments also consist of a limited number of participants since our primary goal is to show feasibility of our proposed approaches. Further work is needed such that the data is collected extensively from different users and from different locations to analyze how the proposed approaches perform in large scale data.

8.2.4 Local vs. Remote Classification

Our approaches classify the test data into binary class either based on specific threshold set from the experiments or based on machine learning. Once the machine learning classifier is trained offline or the thresholds are set for gesture detection, we use it to identify different gestures in real-time. This actual test can either be performed on user's device locally or outsourced to a remote server. The earlier approach allows the device to independently identify the gesture without relying on a third-party server and data connectivity. However, it may require more resources for the testing task. This is in line with many implementations that use machine learning for the purpose of malware detection. There already exist some apps, such as MyWeka in the Android Play store [192], which provides Weka implementation with limited machine learning classifiers. Similarly, Figura [193] provides a Weka library to implement machine learning approach for Android.

In the remote classification approach, whenever there is a need to identify a gesture, the sensor data would be sent to the server, which will perform the classification and provide the result back to the device (all communication between the device and server takes place over a secure channel). A similar approach of using remote server/cloud has already been proposed by Oberheide et al. for cloud based antivirus [194, 195]. The advantage to this approach is that the device does not require extra resources for testing the gesture and running the classifier. However, it needs to have a data connection. In case there is no data connection, the system may fall back to asking user for explicit gesture such as hand-waving/rubbing [35, 112]. A drawback of this approach is the need to trust the remote service – if this service is malicious and colludes with the device malware app, it will completely undermine the security of the system. The delay introduced in transmitting the sensor data to the server may reduce system's usability.

8.3 Future Work

8.3.1 Multiple Devices and Sensors

In our work, we have shown that using multiple sensors enhances the context detection capabilities of the system. Using more/uncorrelated features and implementing them in different ways not only reduced the FNR but also helped decrease FPR. Further, we showed that using multiple devices, such as using smartphone and smartwatch together, can even provide more information gain to the classifier being used. Moreover, using these devices provided additional layer of security to the system in case of loss of one or more devices. Future work may explore other types of wearable devices (such as glasses, which may capture head movements, or shoes, which may capture feet movements) or an additional environmental sensors (such as smell sensor) to further extend our approach, study the implementation of similar techniques on context detection systems, and conduct broader data collection campaigns with larger and diverse population samples.

8.3.2 Defeating Vehicular Criminals

We showed that our approach can be used to authenticate a legitimate user in $\mathcal{Z}IA$ systems such as Blueproximity system [138] and vehicles equipped with (Passive Keyless Entry and Start System (PKES)) [196] to prevent relay attacks. This approach in vehicular authentication can be further elicited to provide full-fledged car authentication system.

We can further design a vehicular system that uses different implicit and environmental context to authenticate the driver comprehensively preventing relay attacks and other vehicular crimes. The system can use implicit context (Section 5.3 – *WUZIA*) to authenticate the driver as he walks towards the vehicle based on his unique gait pattern, thereby transparently unlocking the vehicle door (just like current PKES systems, but with user authentication). The system can use environmental context (Chapter 6) to detect the proximity of the two devices (vehicle's key and vehicle) such that the radio signals are not being relayed. Further, the system can use implicit context to unlock the vehicle's engine as the driver opens the door, sits on the vehicle seat, pushes the brake pedal, and hits a button to start the engine. These implicit gestures can be detected using multiple devices worn by a driver (phone with watch like in *WUZIA*) and internal

vehicular sensors such as pressure sensor on brake pedals. Similarly, while driving, the driver can be authenticated by extracting implicit context of the driver's driving behavior using multiple devices worn by the driver as well as additional sensors embedded inside the vehicle.

We believe that our work can lead to future research on building such contextual systems extracting to authenticate the users in different settings and providing additional layer of security transparently to the users.

LIST OF REFERENCES

[1] ISO. Near field communication interface and protocol (nfcip-1)——iso/iec 18092:2004. Available online at http://www.iso.org/iso/catalogue_detail.htm?csnumber=38578, 2004.

[2] Google.com. Google wallet – an easy way to pay, purchase, and save. Available online at https://www.google.com/wallet/.

[3] Apple.com. Apple pay - apple. Available online at http://www.apple.com/apple-pay/, .

[4] Majd Alwan, Prabhu Jude Rajendran, Steve Kell, David Mack, Siddharth Dalal, Matt Wolfe, and Robin Felder. A smart and passive floor-vibration based fall detector for elderly. In *Information and Communication Technologies, 2006. ICTTA'06. 2nd*, volume 1, pages 1003–1007. IEEE, 2006.

[5] Koray Ozcan, Anvith Katte Mahabalagiri, Mauricio Casares, and Senem Velipasalar. Automatic fall detection and activity classification by a wearable embedded smart camera. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 3(2):125–136, 2013.

[6] Jules White, Chris Thompson, Hamilton Turner, Brian Dougherty, and Douglas C Schmidt. Wreckwatch: Automatic traffic accident detection and notification with smartphones. *Mobile Networks and Applications*, 16(3):285–303, 2011.

[7] Chris Thompson, Jules White, Brian Dougherty, Adam Albright, and Douglas C Schmidt. Using smartphones to detect car accidents and provide situational awareness to emergency responders. In *Mobile Wireless Middleware, Operating Systems, and Applications*, pages 29–42. Springer, 2010.

[8] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 3–14. ACM, 2011.

[9] Jun Han, Emmanuel Owusu, Le T Nguyen, Adrian Perrig, and Joy Zhang. Accomplice: Location inference using accelerometers on smartphones. In *Communication Systems and Networks (COMSNETS)*, pages 1–9. IEEE, 2012.

[10] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *Proc. 23rd USENIX Security Symposium (SEC'14), USENIX Association*, 2014.

[11] Robert Templeman, Zahid Rahman, David Crandall, and Apu Kapadia. Placeraider: Virtual theft in physical spaces with smartphones. In *Network and Distributed System Security Symposium (NDSS)*, volume 20, page 12, 2012.

[12] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. (sp)iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 551–562. ACM, 2011.

[13] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, page 9. ACM, 2012.

[14] Charlie Miller. Exploring the nfc attack surface. *Proceedings of Blackhat*, 2012.

[15] Walt Augustinowicz. Trojan horse electronic pickpocket demo by identity stronghold. Available online at http://www.youtube.com/watch?v=eEcz0XszEic, June 2011.

[16] Mario Ballano. Android threats getting steamy. Available online at http://www.symantec.com/connect/blogs/android-threats-getting-steamy.

[17] Chris Hoffman. ios has app permissions, too: And they're arguably better than android's. Available online at http://www.howtogeek.com/177711/ios-has-app-permissions-too-and-theyre-arguably-better-than-androids/, 2013.

[18] Apple.com. Understanding privacy and location services on iphone, ipad, and ipod touch with ios 8 - apple support. Available online at http://support.apple.com/en-us/HT203033, .

[19] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. In *ACM conference on Computer and communications security*, 2011.

[20] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 3. ACM, 2012.

[21] Nicole Eling, Siegfried Rasthofer, Max Kolhagen, Eric Bodden, and Peter Buxmann. Investigating users' reaction to fine-grained data requests: A market experiment. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 3666–3675. IEEE, 2016.

[22] Erika Chin, Adrienne Porter Felt, Vyas Sekar, and David Wagner. Measuring user confidence in smartphone security and privacy. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 1. ACM, 2012.

[23] Sonia Kolesnikov-Jessop. Hackers go after the smartphone, 2011. www.nytimes.com/2011/02/14/technology/14iht-srprivacy14.html.

[24] Mark Ward. Smartphone security put on test, 2010. Available online at http://www.bbc.com/news/technology-10912376.

[25] A. Juels. Rfid security and privacy: a research survey. *Selected Areas in Communications, IEEE Journal on*, 24(2):381 – 394, feb. 2006. ISSN 0733-8716. doi: 10.1109/JSAC.2005.861395.

[26] Ziv Kfir and Avishai Wool. Picking virtual pockets using relay attacks on contactless smartcard. In *Proc. First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, SECURECOMM '05, pages 47–58, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2369-2. doi: 10.1109/SECURECOMM.2005.32. URL http://dx.doi.org/10.1109/SECURECOMM.2005.32.

[27] G.P. Hancke and M.G. Kuhn. An RFID distance bounding protocol. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005.*, pages 67–73, 2005. doi: 10.1109/SECURECOMM.2005.56.

[28] Aurélien Francillon, Boris Danev, and Srdjan Čapkun. Relay attacks on passive keyless entry and start systems in modern cars. In *Proc. Network and Distributed System Security Symposium (NDSS)*, 2011.

[29] Mark D. Corner and Brian D. Noble. Zero-interaction authentication. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, MobiCom '02, pages 1–11, New York, NY, USA, 2002. ACM. ISBN 1-58113-486-X. doi: 10.1145/570645.570647. URL http://doi.acm.org/10.1145/570645.570647.

[30] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. Sound-proof: Usable two-factor authentication based on ambient sound. In *USENIX Security Symposium*, 2015.

[31] Babins Shrestha, Maliheh Shirvanian, Prakash Shrestha, and Nitesh Saxena. The sounds of the phones: Dangers of zero-effort second factor login based on ambient audio. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

[32] Babins Shrestha, Manar Mohamed, and Nitesh Saxena. Walk-unlock: Zero-interaction authentication protected with multi-modal gait biometrics. *arXiv preprint arXiv:1605.00766*, 2016.

[33] Babins Shrestha, Nitesh Saxena, Hien Thi Thu Truong, and N Asokan. Contextual proximity detection in the face of context-manipulating adversaries. *arXiv preprint arXiv:1511.00905*, 2015.

[34] Manar Mohamed, Babins Shrestha, and Nitesh Saxena. Smashed: Sniffing and manipulating android sensor data. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 152–159. ACM, 2016.

[35] Babins Shrestha, Di Ma, Yan Zhu, Haoyu Li, and Nitesh Saxena. Tap-wave-rub: Lightweight human interaction approach to curb emerging smartphone malware. *IEEE Transactions on Information Forensics and Security*, 10(11):2270–2283, Nov 2015. ISSN 1556-6013. doi: 10.1109/TIFS.2015.2436364.

[36] Babins Shrestha, Manar Mohamed, Anders Borg, Nitesh Saxena, and Sandeep Tamrakar. Curbing mobile malware based on user-transparent hand movements. In *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*, pages 221–229. IEEE, March 2015. doi: 10.1109/PERCOM.2015.7146532.

[37] Hien Thi Thu Truong, Xiang Gao, Babins Shrestha, Nitesh Saxena, N. Asokan, and Petteri Nurmi. Using contextual co-presence to strengthen zero-interaction authentication: Design, integration and usability. *Pervasive and Mobile Computing*, 16, Part B(0):187 – 204, 2015. ISSN 1574-1192. doi: http://dx.doi.org/10.1016/j.pmcj.2014.10.005. URL http://www.sciencedirect.com/science/article/pii/S1574119214001771. Selected Papers from the Twelfth Annual {IEEE} International Conference on Pervasive Computing and Communications (PerCom 2014).

[38] Hien Thi Thu Truong, Xiang Gao, Babins Shrestha, Nitesh Saxena, N. Asokan, and Petteri Nurmi. Comparing and fusing different sensor modalities for relay attack resistance in Zero-Interaction Authentication. In *IEEE International Conference on Pervasive Computing and Communications, PerCom 2014*, pages 163–171, March 2014. doi: 10.1109/PerCom.2014.6813957.

[39] Babins Shrestha, Nitesh Saxena, Hien Thi Thu Truong, and N. Asokan. Drone to the rescue: Relay-resilient authentication using ambient multi-sensing. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *Financial Cryptography and Data Security*, volume 8437 of *Lecture Notes in Computer Science*, pages 349–364. Springer Berlin Heidelberg, 2014. ISBN 978-3-662-45471-8. doi: 10.1007/978-3-662-45472-5_23. URL http://dx.doi.org/10.1007/978-3-662-45472-5_23.

[40] Haoyu Li, Di Ma, Nitesh Saxena, Babins Shrestha, and Yan Zhu. Tap-wave-rub: Lightweight malware prevention for smartphones using intuitive human gestures. In *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '13, pages 25–30, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1998-0. doi: 10.1145/2462096.2462101. URL http://doi.acm.org/10.1145/2462096.2462101.

[41] Yvo Desmedt, Claude Goutier, and Samy Bengio. Special uses and abuses of the fiat-shamir passport protocol. In *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology (CRYPTO)*, 1988.

[42] S. Drimer and S. J. Murdoch. Keep your enemies close: Distance bounding against smartcard relay attacks. In *16th USENIX Security Symposium*, August 2007.

[43] Google Inc. Google 2-step verification. https://www.google.com/landing/2step/.

[44] Duo Security Inc. Easy, mobile two-factor authentication. https://duo.com/solutions/features/user-experience/easy-authentication.

[45] Celestix hotpin two factor authentication. http://www.celestixworks.com/HOTPin.asp.

[46] RSA. Securid — rsa security token based authentication. https://www.rsa.com/en-us/products-services/identity-access-management/securid.

[47] Yubico AB. Trust the net with yubikey strong two-factor authentication. https://www.yubico.com/.

[48] Nick L. Petroni, Jr. and Michael Hicks. Automated detection of persistent kernel control-flow attacks. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 103–115, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-703-2. doi: 10.1145/1315245.1315260. URL http://doi.acm.org/10.1145/1315245.1315260.

[49] Arvind Seshadri, Mark Luk, Ning Qu, and Adrian Perrig. Secvisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity oses. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 335–350, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-591-5. doi: 10.1145/1294261.1294294. URL http://doi.acm.org/10.1145/1294261.1294294.

[50] Simon Eberz, Kasper B Rasmussen, Vincent Lenders, and Ivan Martinovic. Preventing lunchtime attacks: Fighting insider threats with eye movement biometrics. 2015.

[51] Siguna Müller. On the security of an rsa based encryption scheme. In *Information Security and Privacy*, pages 135–148. Springer, 1999.

[52] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[53] Daniel Arp, Michael Spreitzenbarth, Malte Hübner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Proc. of NDSS*, 2014.

[54] Sebastian Poeplau, Yanick Fratantonio, Antonio Bianchi, Christopher Kruegel, and Giovanni Vigna. Execute this! analyzing unsafe and malicious dynamic code loading in android applications. In *Proceedings of the 20th Annual Network & Distributed System Security Symposium (NDSS)*, 2014.

[55] B. Shebaro, O. Oluwatimi, and E. Bertino. Context-based access control systems for mobile devices. *Dependable and Secure Computing, IEEE Transactions on*, PP(99):1–1, 2014. ISSN 1545-5971. doi: 10.1109/TDSC.2014.2320731.

[56] Asaf Shabtai, Robert Moskovitch, Yuval Elovici, and Chanan Glezer. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. *Information Security Technical Report*, 14:16–29, Feb. 2009.

[57] Deepak Venugopal, Guoning Hu, and Nicoleta Roman. Intelligent virus detection on mobile devices. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, PST '06, pages 65:1–65:4, New York, NY, USA, 2006. ACM. ISBN 1-59593-604-1. doi: 10.1145/1501434.1501511. URL http://doi.acm.org/10.1145/1501434.1501511.

[58] Liang Xie, Xinwen Zhang, Jean-Pierre Seifert, and Sencun Zhu. pbmds: A behavior-based malware detection system for cellphone devices. In *Proceedings of the Third ACM Conference on Wireless Network Security*, WiSec '10, pages 37–48, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-923-7. doi: 10.1145/1741866.1741874. URL http://doi.acm.org/10.1145/1741866.1741874.

[59] Abhijit Bose, Xin Hu, Kang G. Shin, and Taejoon Park. Behavioral detection of malware on mobile handsets. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys '08, pages 225–238, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-139-2. doi: 10.1145/1378600.1378626. URL http://doi.acm.org/10.1145/1378600.1378626.

[60] Daniel R. Ellis, John G. Aiken, Kira S. Attwood, and Scott D. Tenaglia. A behavioral approach to worm detection. In *Proceedings of the 2004 ACM Workshop on Rapid Malcode*, WORM '04, pages 43–53, New York, NY, USA, 2004. ACM. ISBN 1-58113-970-5. doi: 10.1145/1029618.1029625. URL http://doi.acm.org/10.1145/1029618.1029625.

[61] Deepak Venugopal. An efficient signature representation and matching method for mobile devices. In *Proceedings of the 2Nd Annual International Workshop on Wireless Internet*, WICON '06, New York, NY, USA, 2006. ACM. ISBN 1-59593-510-X. doi: 10.1145/1234161.1234177. URL http://doi.acm.org/10.1145/1234161.1234177.

[62] A.S. Shamili, C. Bauckhage, and Tansu Alpcan. Malware detection on mobile devices using distributed machine learning. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 4348–4351, Aug 2010. doi: 10.1109/ICPR.2010.1057.

[63] A.-D. Schmidt, R. Bye, H.-G. Schmidt, J. Clausen, O. Kiraz, K.A. Yuksel, S.A. Camtepe, and S. Albayrak. Static analysis of executables for collaborative malware detection on android. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–5, June 2009. doi: 10.1109/ICC.2009.5199486.

[64] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. Crowdroid: Behavior-based malware detection system for android. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '11, pages 15–26, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1000-0. doi: 10.1145/2046614.2046619. URL http://doi.acm.org/10.1145/2046614.2046619.

[65] Jerry Cheng, Starsky H.Y. Wong, Hao Yang, and Songwu Lu. Smartsiren: Virus detection and alert for smartphones. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, MobiSys '07, pages 258–271, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-614-1. doi: 10.1145/1247660.1247690. URL http://doi.acm.org/10.1145/1247660.1247690.

[66] Ashwin Chaugule, Zhi Xu, and Sencun Zhu. A specification based intrusion detection framework for mobile phones. In Javier Lopez and Gene Tsudik, editors, *Applied Cryptography and Network Security*, volume 6715 of *Lecture Notes in Computer Science*, pages 19–37. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-21553-7. doi: 10.1007/978-3-642-21554-4_2. URL http://dx.doi.org/10.1007/978-3-642-21554-4_2.

[67] F. Roesner, T. Kohno, A. Moshchuk, B. Parno, H.J. Wang, and C. Cowan. User-driven access control: Rethinking permission granting in modern operating systems. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 224–238, May 2012. doi: 10.1109/SP.2012.24.

[68] Mauro Conti, Irina Zachia-Zlatea, and Bruno Crispo. Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call. In Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong, editors, *ASIACCS*, pages 249–259. ACM, 2011. ISBN 978-1-4503-0564-8.

[69] Feng Hong, Meiyu Wei, Shujuan You, Yuan Feng, and Zhongwen Guo. Waving authentication: Your smartphone authenticate you on motion gesture. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 263–266. ACM, 2015.

[70] Hugo Gascon, Sebastian Uellenbeck, Christopher Wolf, and Konrad Rieck. Continuous authentication on mobile devices by analysis of typing motion behavior. In *Sicherheit*, 2014.

[71] Muhammad Shahzad, Alex X. Liu, and Arjmand Samuel. Secure unlocking of mobile touch screen devices by simple gestures: You can see it but you can not do it. In *Mobile Computing & Networking*, pages 39–50. ACM, 2013.

[72] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE Transactions on Information Forensics and Security*, Jan 2013.

[73] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and i know it's you!: Implicit authentication based on touch screen patterns. In *SIGCHI Conference on Human Factors in Computing Systems*, CHI, 2012.

[74] Napa Sae-Bae, Kowsar Ahmed, Katherine Isbister, and Nasir Memon. Biometric-rich gestures: A novel approach to authentication on multi-touch devices. In *Conference on Human Factors in Computing Systems*, 2012.

[75] Bendik B Mjaaland, Patrick Bours, and Danilo Gligoroski. Walk the walk: attacking gait biometrics by imitation. In *Information Security*, pages 361–380. Springer, 2010.

[76] Øyvind Stang. Gait analysis: Is it easy to learn to walk like someone else? Master's thesis, Gjøvik University College, 2007.

[77] Jani Mäntyjärvi, Mikko Lindholm, Elena Vildjiounaite, Satu-Marja Mäkelä, and HA Ailisto. Identifying users of portable devices from gait pattern with accelerometers. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 2, pages ii–973. IEEE, 2005.

[78] Liu Rong, Zhou Jianzhong, Liu Ming, and Hou Xiangfeng. A wearable acceleration sensor system for gait recognition. In *Industrial Electronics and Applications*, pages 2654–2659, 2007.

[79] Heikki J Ailisto, Mikko Lindholm, Jani Mantyjarvi, Elena Vildjiounaite, and Satu-Marja Makela. Identifying people from gait pattern with accelerometers. In *Defense and Security*, pages 7–14. International Society for Optics and Photonics, 2005.

[80] Davrondzhon Gafurov, Kirsi Helkala, and Torkjel Søndrol. Biometric gait authentication using accelerometer sensor. *Journal of computers*, 1(7):51–59, 2006.

[81] Torkjel Søndrol. Using the human gait for authentication. Master's thesis, Gjøvik University College, 2005.

[82] Meng Chen, Bufu Huang, and Yangsheng Xu. Intelligent shoes for abnormal gait detection. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2019–2024. IEEE, 2008.

[83] Bufu Huang, Meng Chen, Panfeng Huang, and Yangsheng Xu. Gait modeling for human identification. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4833–4838. IEEE, 2007.

[84] Stacy J Morris. *A shoe-integrated sensor system for wireless gait analysis and real-time therapeutic feedback*. PhD thesis, Massachusetts Institute of Technology, 2004.

[85] Tetsuya Yamamoto, Masahiko Tsukamoto, and Tomoki Yoshihisa. Foot-step input method for operating information devices while jogging. In *Applications and the Internet, 2008. SAINT 2008. International Symposium on*, pages 173–176. IEEE, 2008.

[86] Davrondzhon Gafurov and Einar Snekkenes. Gait recognition using wearable motion recording sensors. *EURASIP Journal on Advances in Signal Processing*, 2009:7, 2009.

[87] Elena Vildjiounaite, Satu-Marja Mäkelä, Mikko Lindholm, Reima Riihimäki, Vesa Kyllönen, Jani Mäntyjärvi, and Heikki Ailisto. Unobtrusive multimodal biometrics for

ensuring privacy and information security with personal devices. In *Pervasive Computing*, pages 187–201. Springer, 2006.

[88] Yoon Sang Kim, Byung Seok Soh, and Sang-Goog Lee. A new wearable input device: Scurry. *Industrial Electronics, IEEE Transactions on*, 52(6):1490–1499, 2005.

[89] John Kangchun Perng, Brian Fisher, Seth Hollar, and Kristofer SJ Pister. Acceleration sensing glove. In *iswc*, page 178. IEEE, 1999.

[90] Michele Sama, Vincenzo Pacella, Elisabetta Farella, Luca Benini, and Bruno Riccó. 3did: a low-power, low-cost hand motion capture device. In *Proceedings of the conference on Design, automation and test in Europe: Designers' forum*, pages 136–141. European Design and Automation Association, 2006.

[91] Davrondzhon Gafurov, Einar Snekkenes, and Patrick Bours. Spoof attacks on gait authentication system. *Information Forensics and Security, IEEE Transactions on*, 2(3):491–502, 2007.

[92] Rajesh Kumar, Vir V Phoha, and Anshumali Jain. Treadmill attack on gait-based authentication systems. In *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on*, pages 1–7. IEEE, 2015.

[93] Mark A Hall and Geoffrey Holmes. Benchmarking attribute selection techniques for discrete class data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 15 (6):1437–1447, 2003.

[94] Andrew H Johnston and Gary M Weiss. Smartwatch-based biometric gait recognition. In *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on*, pages 1–6. IEEE, 2015.

[95] Rajesh Kumar, Vir V Phoha, and Rahul Raina. Authenticating users through their arm movement patterns. *arXiv preprint arXiv:1603.02211*, 2016.

[96] Stefan Brands and David Chaum. Distance-bounding protocols. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '93, pages 344–359. Springer-Verlag New York, Inc., 1994. ISBN 3-540-57600-2. URL http://dl.acm.org/citation.cfm?id=188307.188361.

[97] Jason Reid, Juan M. Gonzalez Nieto, Tee Tang, and Bouchra Senadji. Detecting Relay Attacks with Timing-based Protocols. In *Proc. 2nd ACM symposium on Information,*

computer and communications security, ASIACCS '07, pages 204–213, New York, NY, USA, 2007. ACM. ISBN 1-59593-574-6. doi: 10.1145/1229285.1229314.

[98] Tzipora Halevi, Di Ma, Nitesh Saxena, and Tuo Xiang. Secure Proximity Detection for NFC Devices Based on Ambient Sensor Data. In *Proceedings of 17th European Symposium on Research in Computer Security, ESORICS 2012*, volume 7459 of *Lecture Notes in Computer Science*, pages 379–396. Springer, 2012. ISBN 978-3-642-33166-4.

[99] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. 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, pages 35–49, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-16821-3, 978-3-642-16821-5. URL http://dl.acm.org/citation.cfm?id=1926325.1926331.

[100] Gerhard P. Hancke and Markus G. Kuhn. Attacks on time-of-flight distance bounding channels. In *Proceedings of the first ACM conference on Wireless network security*, WiSec '08, pages 194–202, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-814-5. doi: 10.1145/1352533.1352566. URL http://doi.acm.org/10.1145/1352533.1352566.

[101] Ngu Nguyen, Stephan Sigg, An Huynh, and Yusheng Ji. Pattern-based alignment of audio data for ad hoc secure device pairing. In *16th International Symposium on Wearable Computers, ISWC*, pages 88–91. IEEE, 2012.

[102] Ngu Nguyen, Stephan Sigg, An Huynh, and Yusheng Ji. Using ambient audio in secure mobile phone communication. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pages 431–434. IEEE, 2012. doi: 10.1109/PerComW.2012.6197527.

[103] Dominik Schurmann and Stephan Sigg. Secure communication based on ambient audio. *IEEE Transactions on Mobile Computing*, 12(2):358–370, February 2013. ISSN 1536-1233. doi: 10.1109/TMC.2011.271. URL http://dx.doi.org/10.1109/TMC.2011.271.

[104] John Krumm and Ken Hinckley. The NearMe Wireless Proximity Server. In Nigel Davies, ElizabethD. Mynatt, and Itiro Siio, editors, *UbiComp 2004: Ubiquitous Computing*, volume 3205 of *Lecture Notes in Computer Science*, pages 283–300. Springer Berlin

Heidelberg, 2004. ISBN 978-3-540-22955-1. doi: 10.1007/978-3-540-30119-6_17. URL http://dx.doi.org/10.1007/978-3-540-30119-6_17.

[105] Alex Varshavsky, Adin Scannell, Anthony LaMarca, and Eyal De Lara. Amigo: Proximity-based authentication of mobile devices. In *Proc. 9th International Conference on Ubiquitous Computing (UbiComp)*, pages 253–270. Springer, 2007.

[106] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. Location Privacy via Private Proximity Testing. In *Proc. Network and Distributed System Security Symposium (NDSS)*, 2011.

[107] Alexei Czeskis, Karl Koscher, Joshua R. Smith, and Tadayoshi Kohno. RFIDs and Secret Handshakes: Defending Against Ghost-and-leech Attacks and Unauthorized Reads with Context-aware Communications. In *Proc. 15th ACM conference on Computer and communications security*, CCS '08, pages 479–490, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-810-7. doi: 10.1145/1455770.1455831.

[108] Alexei Czeskis, Michael Dietz, Tadayoshi Kohno, Dan Wallach, and Dirk Balfanz. Strengthening user authentication through opportunistic cryptographic identity assertions. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 404–414, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1651-4. doi: 10.1145/2382196.2382240. URL http://doi.acm.org/10.1145/2382196.2382240.

[109] Authy Inc. Two-factor authentication - Authy. https://www.authy.com/.

[110] Maliheh Shirvanian, Stanislaw Jarecki, Nitesh Saxena, and Naveen Nathan. Two-factor authentication resilient to server compromise using mix-bandwidth devices. In *Network and Distributed System Security Symposium*, 2014.

[111] Google acquires slicklogin, the sound-based password alternative. Available online at http://techcrunch.com/2014/02/16/google-acquires-slicklogin-the-sound-based-password-alternative/.

[112] Babins Shrestha, Nitesh Saxena, and Justin Harrison. Wave-to-access: Protecting sensitive mobile device services via a hand waving gesture. In Michel Abdalla, Cristina Nita-Rotaru, and Ricardo Dahab, editors, *Cryptology and Network Security*, volume 8257 of *Lecture Notes in Computer Science*, pages 199–217. Springer, 2013. ISBN 978-3-319-02936-8.

doi: 10.1007/978-3-319-02937-5_11. URL http://dx.doi.org/10.1007/978-3-319-02937-5_11.

[113] Charlie Sorrel. Lockitron: Unlock your home with your cellphone. Available online at http://www.wired.com/2011/05/lockitron-unlock-your-home-with-your-cellphone/.

[114] Rian Boden. Yale introduces residential deadbolt with nfc unlocking. Available online at http://www.nfcworld.com/2015/01/06/333372/yale-introduces-residential-deadbolt-nfc-unlocking/.

[115] Karl Dyer. Czech firm releases universal nfc id system. Available online at http://www.nfcworld.com/2013/06/19/324720/czech-firm-releases-universal-nfc-id-system/.

[116] Christoph Busold, Ahmed Taha, Christian Wachsmann, Alexandra Dmitrienko, Hervé Seudié, Majid Sobhani, and Ahmad-Reza Sadeghi. Smart keys for cyber-cars: secure smartphone-based nfc-enabled car immobilizer. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 233–242. ACM, 2013.

[117] Rainer Steffen, Jörg Preißinger, Tobias Schöllermann, Armin Müller, and Ingo Schnabel. Near field communication (nfc) in an automotive environment. In *2nd International Workshop on Near Field Communication*, pages 15–20. IEEE, 2010.

[118] Luka Finžgar and Mira Trebar. Use of nfc and qr code identification in an electronic ticket system for public transport. In *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*, pages 1–6. IEEE, 2011.

[119] Jimmy Gautam, Yogesh Kumar, and Arpan Gupta. Existing scenario of near field communication in transport sector. In *Signal Processing and Integrated Networks (SPIN), 2014 International Conference on*, pages 327–332. IEEE, 2014.

[120] Patrick Macgougan. Asia pacific to lead growth in global mobile payments. Available online at http://go-mashmobile.com/mcommerce-2/news-mcommerce-2/asia-pacific-to-lead-growth-in-global-mobile-payments-2828/.

[121] Android.com. Android – android pay. Available online at https://www.android.com/pay/.

[122] Samsung.com. Samsung pay - safe and simple mobile payments. Available online at http://www.samsung.com/us/samsung-pay/.

[123] Serge Egelman, Sakshi Jain, Rebecca S. Portnoff, Kerwell Liao, Sunny Consolvo, and David Wagner. Are you ready to lock? CCS '14, 2014.

[124] Jan Lauren Boyles, Aaron Smith, and Mary Madden. Privacy and data management on mobile devices. Available online at http://www.pewinternet.org/2012/09/05/privacy-and-data-management-on-mobile-devices/.

[125] Lookout.com. Mobile lost & found: Your phone's favorite hiding places. Available online at https://blog.lookout.com/blog/2012/03/22/.

[126] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Commun. ACM*, 42 (12):40–46, 1999.

[127] Robert Morris and Ken Thompson. Password security: a case history. *Commun. ACM*, 22 (11):594–597, 1979.

[128] Jeff Yan, Alan Blackwell, Ross Anderson, and Alasdair Grant. Password memorability and security: Empirical results. *IEEE Security and Privacy*, 2(5):25–31, 2004.

[129] Liam Tung. Google looks to ditch passwords for good with nfc-based replacement. Available online at http://www.zdnet.com/article/google-looks-to-ditch-passwords-for-good-with-nfc-based-replacement/.

[130] Robert McMillan. Google declares war on the password. Available online at http://www.wired.com/2013/01/google-password/all/.

[131] Anil Jain, Lin Hong, and Sharath Pankanti. Biometric identification. *Communications of the ACM*, 43(2):90–98, 2000.

[132] Roman V Yampolskiy and Venu Govindaraju. Behavioural biometrics: a survey and classification. *International Journal of Biometrics*, 1(1):81–113, 2008.

[133] ARM. ARM Security Technology Building a Secure System using TrustZone Technology. Technical report, April 2009. URL http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/index.html.

[134] Timo Kasper, Ingo von Maurich, David Oswald, and Christof Paar. Chameleon: A versatile emulator for contactless smartcards. In Kyung-Hyune Rhee and DaeHun Nyang, editors, *Information Security and Cryptology - ICISC 2010*, volume 6829 of *Lecture Notes in*

Computer Science, pages 189–206. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-24208-3. doi: 10.1007/978-3-642-24209-0_13. URL http://dx.doi.org/10.1007/978-3-642-24209-0_13.

[135] Roy Maxion, Kevin S Killourhy, et al. Keystroke biometrics with number-pad input. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 201–210. IEEE, 2010.

[136] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105–114. ACM, 2010.

[137] Abdul Serwadda and Vir V. Phoha. When kids' toys breach mobile phone security. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 599–610. ACM, 2013.

[138] BlueProximity. SourceForge Project. http://sourceforge.net/projects/blueproximity/.

[139] Mercedes-Benz. Mercedes-Benz TechCenter: KEYLESS GO, 2016. http://techcenter.mercedes-benz.com/_en/keylessgo/detail.html.

[140] Bruce Tognazzini. The apple iwatch. Blog posting in AskTOG: Interaction Design Solutions for the Real World, Feb 2013. http://asktog.com/atc/apple-iwatch/.

[141] Trevor Mogg. Study reveals americans lost $30 billion worth of mobile phones last year, 2012. http://www.digitaltrends.com/mobile/study-reveals-americans-lost-30-billion-of-mobile-phones-last-year.

[142] Lookout. Phone Theft In America, 2014. https://www.lookout.com/resources/reports/phone-theft-in-america.

[143] Donna Tapellini. Smart phone thefts rose to 3.1 million in 2013, 2014. http://www.consumerreports.org/cro/news/2014/04/smart-phone-thefts-rose-to-3-1-million-last-year/index.htm.

[144] LG USA. Lg nexus 5: Made for what matters — lg usa. Available online at http://www.lg.com/us/cell-phones/lg-D820-Black-nexus-5, .

[145] LG USA. Lg watch r (w110): Design comes full circle — lg usa. Available online at http://www.lg.com/us/smart-watches/lg-W110-lg-watch-r, .

[146] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[147] Miao Liu, Mingjun Wang, Jun Wang, and Duo Li. Comparison of random forest, support vector machine and back propagation neural network for electronic tongue data classification: Application to the recognition of orange beverage and chinese vinegar. *Sensors and Actuators B: Chemical*, 177:970–980, 2013.

[148] Keith G. Calkins. Correlation coefficients. Available online at https://www.andrews.edu/ calkins/math/edrm611/edrm05.htm.

[149] Explorable.com. Statistical correlation. Available online at https://explorable.com/statistical-correlation.

[150] Ka-Ping Yee. Aligning security and usability. *IEEE Security & Privacy*, 2(5):48–55, 2004.

[151] Albert Levi, Erhan Cetintas, Murat Aydos, Cetin Kaya Koc, and M.Ufuk Caglayan. Relay Attacks on Bluetooth Authentication and Solutions. In *Computer and Information Sciences - ISCIS*. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-23526-2. doi: 10.1007/978-3-540-30182-0_29. URL http://dx.doi.org/10.1007/978-3-540-30182-0_29.

[152] Di Ma, Nitesh Saxena, Tuo Xiang, and Yan Zhu. Location-Aware and Safer Cards: Enhancing RFID Security and Privacy via Location Sensing. *IEEE Transactions on Dependable and Secure Computing*, 10(2):57–69, March 2013. ISSN 1545-5971. doi: 10.1109/TDSC.2012.89. URL http://dx.doi.org/10.1109/TDSC.2012.89.

[153] Olivier Dousse, Julien Eberle, and Matthias Mertens. Place learning via direct WiFi fingerprint clustering. *2012 IEEE 13th International Conference on Mobile Data Management*, 0:282–287, 2012. doi: http://doi.ieeecomputersociety.org/10.1109/MDM.2012.46.

[154] W. Gellert, S. Gottwald, and M. Hellwich. *The VNR concise encyclopedia of mathematics*. Van Nostrand Reinhold New York, 2nd edition, 1989. ISBN 0442205902.

[155] Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. SoundSense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones.

In *Proc. 7th international conference on Mobile systems, applications, and services*, pages 165–178, 2009.

[156] Avery Wang. The shazam music recognition service. *Commun. ACM*, 49(8):44–48, August 2006. ISSN 0001-0782. doi: 10.1145/1145287.1145312. URL http://doi.acm.org/10.1145/1145287.1145312.

[157] Wai-Tian Tan, Mary Baker, Bowon Lee, and Ramin Samadani. The sound of silence. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 19:1–19:14, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2027-6. doi: 10.1145/2517351.2517362. URL http://doi.acm.org/10.1145/2517351.2517362.

[158] Geoffrey I. Webb. Multiboosting: A technique for combining boosting and wagging. *Mach. Learn.*, 40(2):159–196, August 2000. ISSN 0885-6125. doi: 10.1023/A:1007659514849.

[159] Peter Clarke. Sensirion preps multi-gas sensor 'nose' for smartphones. Available online at http://www.electronics-eetimes.com/en/sensirion-preps-multi-gas-sensor-nose-for-smartphones.html?cmp_id=7&news_id=222919117.

[160] Sergey Yurish. Smartphone sensing: What sensors would we like to have in the future smartphones? Available online at http://www.iaria.org/conferences2012/filesSENSORDEVICES12/Yurish_Smartphone_Sensing.pdf.

[161] Reto Meier. *Professional Android 4 application development*. John Wiley & Sons, 2012.

[162] R.J. Widlar. An exact expression for the thermal variation of the emitter base voltage of bi-polar transistors. *Proceedings of the IEEE*, 55(1):96–97, 1967. ISSN 0018-9219. doi: 10.1109/PROC.1967.5396.

[163] Denes K Roveti. Choosing a humidity sensor: A review of three technologies this discussion of the operating principles of capacitive, resisitive, and thermal conductivity humidity sensors also addresses their advantages, disadvantages, and applications. *Sensors-the Journal of Applied Sensing Technology*, 18(7):54–58, 2001.

[164] National Oceanic and Atmospheric Association. Pressure altitude. Available online at http://www.wrh.noaa.gov/slc/projects/wxcalc/formulas/pressureAltitude.pdf.

[165] Mark Rudolph. Sensordrone-control. Available online at https://github.com/Sensorcon/Sensordrone-Control, March 2013.

[166] Mark Hall et al. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11 (1):10–18, November 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656278.

[167] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL http://dx.doi.org/10.1023/A:1010933404324.

[168] Sensor Modalities. Online. http://web.eecs.utk.edu/~leparker/Courses/CS594-fall08/Lectures/Sept-18-Perception-2.pdf.

[169] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[170] Nils Ole Tippenhauer, Kasper Bonne Rasmussen, Christina Pöpper, and Srdjan Čapkun. Attacks on Public WLAN-based Positioning Systems. In *Proc. 7th International Conference on Mobile Systems, Applications, and Services*, MobiSys '09, pages 29–40, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-566-6. doi: 10.1145/1555816.1555820. URL http://doi.acm.org/10.1145/1555816.1555820.

[171] Scapy. Online. http://www.secdev.org/projects/scapy/.

[172] DD-WRT.com. www.dd-wrt.com — Unleash Your Router. Available online at http://www.dd-wrt.com/site/index.

[173] Hair Dryer. Conair. http://infiniti.conair.com/catalog.php?pcID=47&products_id=232.

[174] Hair Dryer Activity. Youtube. https://www.youtube.com/watch?v=3QG79NH0qjA.

[175] Michael Rubens Bloomberg and Salvatore Cassano. Fire safety education. http://www.nyc.gov/html/fdny/pdf/safety/fire_safety_education/2010_02/08_smoke_and_carbon_monoxide_alarms_english.pdf.

[176] Electric Air Pump. Walmart. http://www.walmart.com/ip/33563196.

[177] Ziploc and Air Pump Attack Video. Youtube. https://www.youtube.com/watch?v=Fy2F8rY6bzw.

[178] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2078195.

[179] Coalition of law enforcement hacked & agents information leaked, . http://thehackernews.com/2011/12/coalition-of-law-enforcement-hacked.html.

[180] Sony europe hacked by lebanese hacker... again - naked security. https://nakedsecurity.sophos.com/2011/06/04/sony-europe-hacked-by-lebanese-hacker-again/.

[181] Hacker leaks 250gb of nasa data, another group claims to hijack nasa drone. https://www.hackread.com/nasa-data-leaked-nasa-drone-hacked/.

[182] Bulgarian torrent tracker forum hacked and accused of collecting user ip, . http://thehackernews.com/2012/11/bulgarian-torrent-tracker-forum-hacked.html.

[183] Anonymous leaks database from israeli musical act magazine site #opisrael, . http://thehackernews.com/2012/12/anonymous-leaks-database-from-israeli.html.

[184] What's in the ashley madison database that hackers released online - quartz. http://qz.com/482875/whats-in-the-ashley-madison-database-that-hackers-released-online/.

[185] Snapchat hacked: 4.6 million user names, partial phone numbers leaked - ABC15 Arizona. http://www.abc15.com/news/local-news/water-cooler/snapchat-hacked-46-million-user-names-partial-phone-numbers-leaked.

[186] Google Inc. WebRTC. https://webrtc.org/.

[187] University of Illinois at Urbana-Champaign. Center frequencies and high-/low frequency limits for octave bands, 1/2- and 1/3-octave bands, 2011. https://courses.physics.illinois.edu/phys193/labs/octave_bands.pdf.

[188] The Engineering ToolBox. Octave bands - frequency limits. http://www.engineeringtoolbox.com/octave-bands-frequency-limits-d_1602.html.

[189] MathWorks. Butterworth filter design. http://www.mathworks.com/help/signal/ref/butter.html.

[190] Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, and Lorrie Faith Cranor. Crying wolf: An empirical study of ssl warning effectiveness. In *USENIX Security Symposium*, 2009.

[191] David Lane. Computing Pearson's r, 2003. http://cnx.org/contents/9zcGzDDu@4/Computing-Pearsons-r.

[192] Eduardo González. Myweka - android apps on google play. Available online at https://play.google.com/store/apps/details?id=weka.tfc.

[193] Juraj Figura. Machine learning for google android. Available online at http://www.cestina.cz/ obo/vyuka/projekty/figura-ml-for-android.pdf.

[194] Jon Oberheide, Kaushik Veeraraghavan, Evan Cooke, Jason Flinn, and Farnam Jahanian. Virtualized in-cloud security services for mobile devices. In *Proceedings of the First Workshop on Virtualization in Mobile Computing*, pages 31–35. ACM, 2008.

[195] Jon Oberheide, Evan Cooke, and Farnam Jahanian. Cloudav: N-version antivirus in the network cloud. In *USENIX Security Symposium*, pages 91–106, 2008.

[196] Aurelien Francillon, Boris Danev, and Srdjan Capkun. Relay attacks on passive keyless entry and start systems in modern cars. Cryptology ePrint Archive, Report 2010/332, 2010. http://eprint.iacr.org/.

[197] Car Vacuum Activity. Youtube. https://www.youtube.com/watch?v=vN_ZBi_rmJc.

Appendix

A *WUZIA*: Feature Correlation

A.1 Correlation Analysis

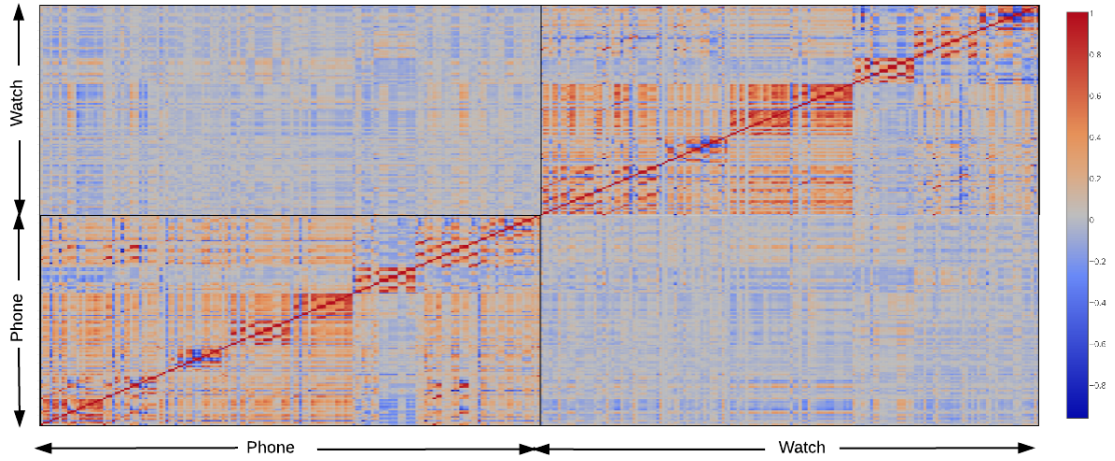


FIGURE 1: Heatmap showing pairwise correlation values between all the 336 features. Red depicts high positive correlation, Blue depicts high negative correlation, and Grey depicts low correlation. The first half (first half rows and the first half columns) represents the features associated with the phone while the second half (the second half rows and the second half columns) represents the features associated with the watch. We can see that the features of phones are more correlated with those of phone while features of watch are more correlated with those of watch.

Correlation is commonly used to find the relationship between two or more objects. To find the similarity between two different features, we calculated the correlation as follows. Let x and y be the values of two feature vectors and we have n data samples for each feature vector.

$$S_{xx} = \sum x^2 - \frac{(\sum x)^2}{n}; \quad S_{yy} = \sum y^2 - \frac{(\sum y)^2}{n} \tag{1}$$

$$S_{xy} = \sum xy - \frac{(\sum x)(\sum y)}{n} \tag{2}$$

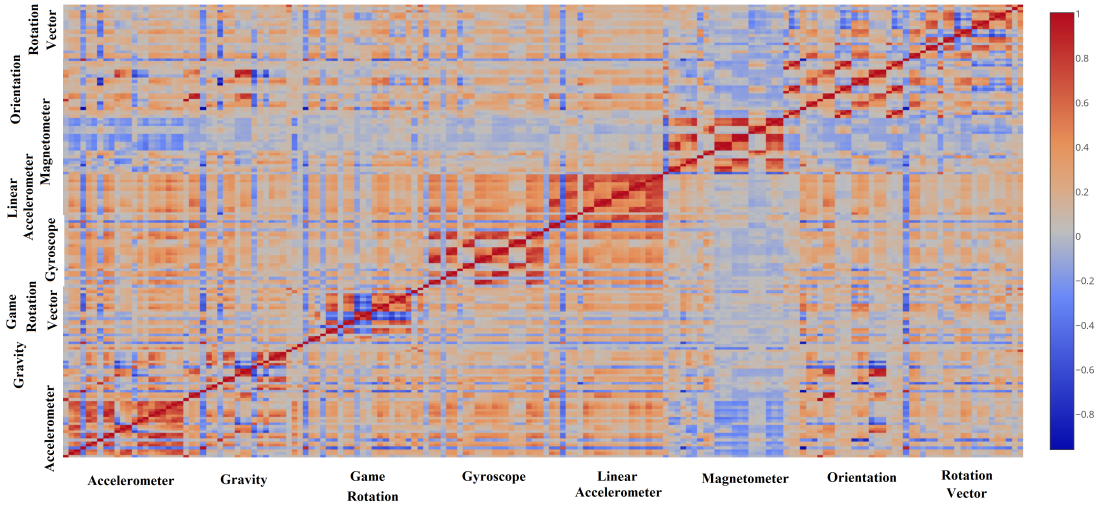


FIGURE 2: Heatmap showing pairwise correlation between the features from phone only. The color depiction is same as in previous figure. The features extracted using the same sensor have higher correlation compared to those extracted using different sensors.

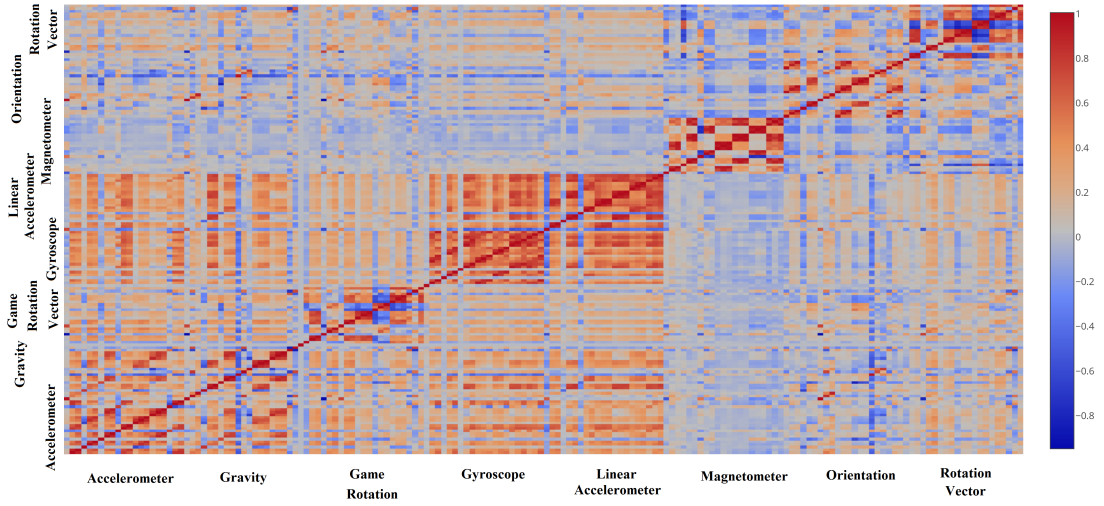


FIGURE 3: Heatmap showing pairwise correlation between the features from watch only. The color depiction is same as in previous figure. The features extracted using the same sensor have higher correlation compared to those extracted using different sensors.

The correlation (σ) between the features vectors x and y is

$$\sigma = \frac{S_{xy}}{\sqrt{S_{xx}S_{yy}}} \tag{3}$$

The calculation of the correlation (σ) is based on the alternative computation formula for Pearson's r [191]. The alternative computation for the Pearson's r avoids the step of computing deviation scores.

We compute the correlation between each features from different devices (smartphone and smartwatch) and plot the heat map as shown in Appendix A Figures 1, 2, and 3.

B Environment Manipulation

B.1 Increasing the temperature when the attacker does not know VS's location

An attacker who does not know the location of VS will try to keep the FS as close as possible and perform the attack activity. We placed the FS 10 cm apart from the VS and performed experiment in two settings. In the first setting, the hair dryer is closer to VS as shown in Appendix B Figures 4, and in the latter setting, the hair dryer is closer to FS as shown in Appendix B Figures 5.

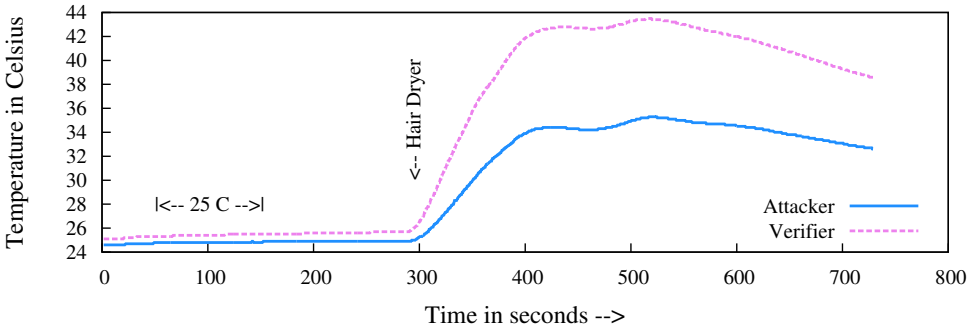


FIGURE 4: Increasing the temperature; location of VS unknown to the attacker; VS is 10 cm closer to hair dryer than FS; the attacker trying to increase temperature to 35 °C.

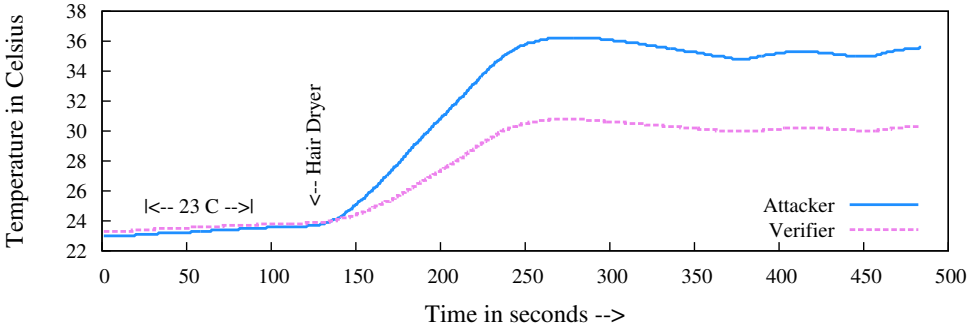


FIGURE 5: Increasing the temperature; location of VS unknown to the attacker; FS is 10 cm closer to hair dryer than VS; the attacker trying to increase temperature to 35 °C.

B.2 Increasing the CO gas level

We effectively manipulated the CO gas sensor using cigarette and car exhaust. The increase in the gas level due to the activity is abrupt when CO is blown onto the sensors, however, it takes a

while for the sensors to fall back to normal readings. This provides an enough time window for the attacker as depicted in Figs 6 and 7.

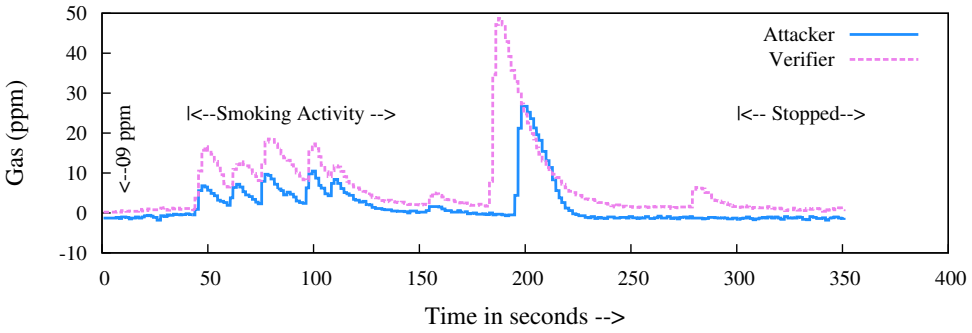


FIGURE 6: Effect of cigarette in CO level; increasing the gas content to an arbitrary value and waiting to decrease to desired level.

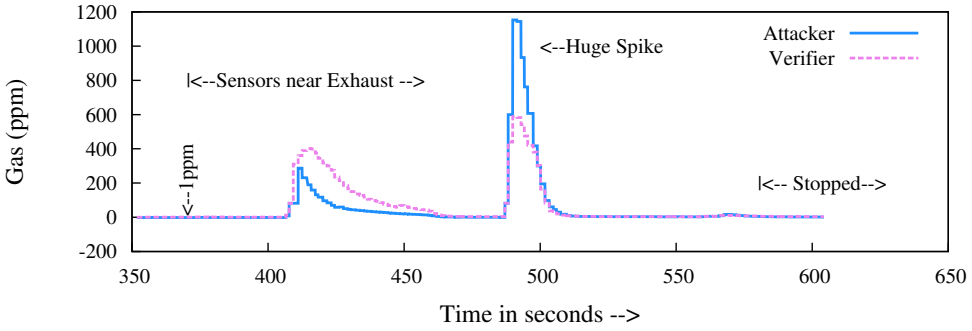


FIGURE 7: Effect of car exhaust in CO level; increasing the CO gas level to arbitrary value and wait to decrease to desired level.

B.3 Increasing the altitude using a car vacuum

As an alternative to air pump, we tried a portable car vacuum cleaner for inducing an altitude increase. When we hovered the vacuum cleaner pipe around the sensors, it did not have any effect. However, when we put the pipe just on top of the sensor, it increased the altitude by 10-11 meters as shown in Appendix B Figures 8. An attacker can adjust the altitude to a desired level by changing the power level of the vacuum cleaner, similar to the air pump manipulation. The earlier part of the Appendix B Figures 8 shows a little fluctuation in altitude when we hovered the pipe around the sensors while the later spikes clearly show that there was an increase of almost 10 meters when the pipe was touched to the sensors. A video demo of our attack has been uploaded to YouTube [197] to show the effect of portable car vacuum cleaner on the pressure/altitude sensors.

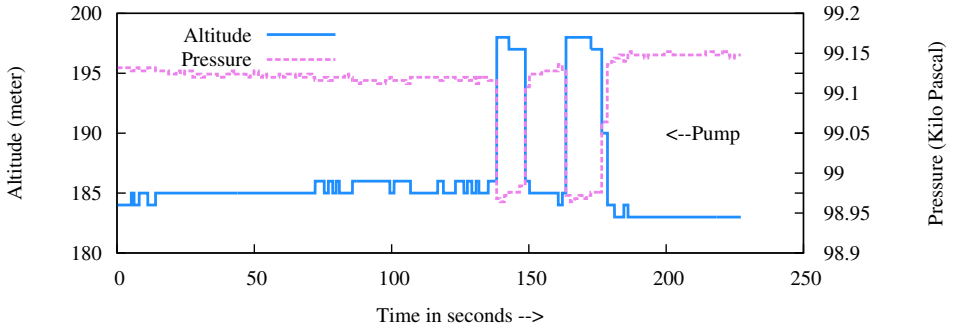


FIGURE 8: Using a car vacuum cleaner to reduce pressure around the sensor and increase the altitude.

C Demographic Information

In this section, we show the demographic information of the users who participated in our online survey for the phone usage for attacking *Sound-Proof*. Most participants are from U.S.A. with Bachelor's degree and within the age of 25-34%. The detailed demographic information is shown in Table 1.

TABLE 1: Demographic Info: Online Survey Study

	N = 100
Gender	
Male	50.82%
Female	49.18%
Age	
18-24 years	12.30%
25-34 years	54.92%
35-44 years	22.95%
45-54 years	8.20%
55-64 years	1.64%
Education	
High school graduate, diploma	6.56%
Some college credit, no degree	25.41%
Associate degree	7.38%
Bachelor's degree	40.16%
Master's degree	1.721%
Professional degree	0.82%
Doctorate degree	2.46%

	N = 100
Industry	
Information Technology	15.57%
Education	14.75%
Financial Services	10.66%
Marketing/Sales	7.38%
Healthcare	7.38%
Technical Services	6.56%
Entertainment	3.28%
Non-profit	3.28%
Others	24.58%
Country of Residence	
United States	77.05%
India	22.13%
Others	0.82%