# On the Security of Mobile Sensors


Maryam Mehrnezhad

School of Computing Science

Newcastle University, UK

A thesis submitted for the degree of

*Doctor of Philosophy*

April 2017

To science
In the hope of a better world
For generations to come

تقدیم به علم

به امید دنیایی بهتر

برای آیندگان

# Acknowledgements

I would like to thank my supervisors, Dr. Feng Hao and Dr. Siamak F. Shahandashti – I am truly grateful to them for their expertise, guidance, and support of my research, and their mentorship and friendship over the years. I would like to thank the School of Computing Science, Newcastle University and the Secure and Resilient Systems (SRS) group for their world-class and friendly academic environment, and the generous support that they provided to me during my studies.

I would like to acknowledge specific people with whom I had the pleasure to collaborate during this research project. In Chapter 3, Mohammed A. Ali and Ehsan Toreini helped with designing and performing some of the experiments. Prof. Aad van Moorsel, Dr. Michael Ward (from MasterCard), and Dr. Martin Emms contributed insights on the EMV contactless specifications. The initial JavaScript code used in Chapter 4 was developed by Ehsan Toreini. In Chapter 5, Dr. Kovila Coopamootoo provided feedback on designing the user studies; Ehsan Toreini, and Dr. Siamak F. Shahandashti helped with participant recruitment and interviews.

I wish to thank my examiners Dr. Charles Morisset form Newcastle University and Prof. Chris Mitchell from Royal Holloway, University of London for their constructive feedback on this thesis. Many people from W3C, Qualcomm, Intel, Apple, Google Chrome, Mozilla Firefox, Opera, ISO, EMV, and MasterCard provided valuable feedback on some of the works done in this project. I thank all the volunteer participants in the technical experiments and user studies of this research. All the experiments of this thesis gained approval through Newcastle University's research ethics processes.

I wish to thank the following people from Newcastle University for their support over the years: Prof. Brian Randell, Prof. Aad van Moorsel, Prof. Tom Anderson, my friends: Sami, Zoya, Ako, David, Paddy, Diego, Iryna, Cov, the school admins: Dee Carr, Jill Green, Catherine McAndrew, and

my mentor: Dr. Joy Dinsdale. My special thanks goes to my friends beyond the university: Mahshid, Marziyeh, Nasrin, Elham, Mehran, Leila, Nadia, Mahsa, Samin, and Fesenjoon. I would like to thank the Blackswan company (Tynemouth, UK) whom I had the pleasure to work with toward the end of my PhD.

Finally, I would like to thank my beloved parents, sisters, and grandparents for their faith, encouragement, support, and love over the years of my studies. Last but not least, I would like to truly thank my husband and my best friend, Ehsan, who believed in me more than I did myself. I feel fortunate to share my passion for science and this journey with him.

# Publications

### Journals

- M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "TouchSignatures: Identification of User Touch Actions and PINs based on Mobile Sensors via JavaScript", Journal of Information Security and Applications, Elsevier, Volume 26, February 2016, Pages 23-38.
- M. Mehrnezhad, E. Toreini, S. F. Shahandashti, F Hao, "Stealing PINs via Mobile Sensors: Actual Risk versus User Perception", International Journal of Information Security, Springer, April 2017, Pages 1-23.

### Conferences

- M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "TouchSignatures: Identification of User Touch Actions based on Mobile Sensors via JavaScript", In the Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS 2015, Singapore, April 14-17, 2015, ACM, Pages 673-673.
- M. Mehrnezhad, F. Hao, and S. F. Shahandashti, "Tap-Tap and Pay (TTP): Preventing Man-In-The-Middle Attacks in NFC Payment Using Mobile Sensors", In the Proceedings of the Second International Conference of Security Standardisation Research, SSR 2015, Tokyo, Japan, December 15-16, 2015, Springer International Publishing, Pages 21-39.
- M. Mehrnezhad, E. Toreini, S. F. Shahandashti, F Hao, "Stealing PINs via Mobile Sensors: Actual Risk versus User Perception", The 1st European Workshop on Usable Security, EuroUSEC 2016, Workshop at the Privacy Enhancing Technologies Symposium (PETS 2016), July 18, 2016, Germany.

- M. Mehrnezhad, M. A. Ali, F. Hao, A. V. Moorsel, "NFC Payment Spy: Privacy attacks on contactless payments using NFC-enabled mobile", In the Proceedings of the Third International Conference of Security Standardisation Research, SSR 2016, USA, December 5-6, 2016, Springer International Publishing, Pages 1-20.

# Abstract

The age of sensor technology is upon us. Sensor-rich mobile devices are ubiquitous. Smart-phones, tablets, and wearables are increasingly equipped with sensors such as GPS, accelerometer, Near Field Communication (NFC), and ambient sensors. Data provided by such sensors, combined with the fast-growing computational capabilities on mobile platforms, offer richer and more personalised apps. However, these sensors introduce new security challenges to the users, and make sensor management more complicated.

In this PhD thesis, we contribute to the field of mobile sensor security by investigating a wide spectrum of open problems in this field covering attacks and defences, standardisation and industrial approaches, and human dimensions. We study the problems in detail and propose solutions.

First, we propose "Tap-Tap and Pay" (TTP), a sensor-based protocol to prevent the *Mafia attack* in NFC payment. The Mafia attack is a special type of Man-In-The-Middle attack which charges the user for something more expensive than what she intends to pay by relaying transactions to a remote payment terminal. In TTP, a user initiates the payment by physically tapping her mobile phone against the reader. We observe that this tapping causes transient vibrations at both devices which are measurable by the embedded accelerometers. Our observations indicate that these sensor measurements are closely correlated within the same tapping, and different if obtained from different tapping events. By comparing the similarity between the two measurements, the bank can distinguish the Mafia fraud apart from a legitimate NFC transaction. The experimental results and the user feedback suggest the practical feasibility of TTP. As compared with previous sensor-based solutions, ours is the only one that works even when the attacker and the user are in nearby locations or share similar ambient environments.

Second, we demonstrate an in-app attack based on a real world problem in contactless payment known as the *card collision* or *card clash*. A card collision happens when more than one card (or NFC-enabled device) are presented to the payment terminal's field, and the terminal does not know which card to choose. By performing experiments, we observe that the implementation of contactless terminals in practice matches neither EMV nor ISO standards (the two primary standards for smart card payment) on card collision. Based on this inconsistency, we propose "NFC Payment Spy", a malicious app that tracks the user's contactless payment transactions. This app, running on a smart phone, simulates a card which requests the payment information (amount, time, etc.) from the terminal. When the phone and the card are both presented to a contactless terminal (given that many people use mobile case wallets to travel light and keep wallet essentials close to hand), our app can effectively win the race condition over the card. This attack is the first privacy attack on contactless payments based on the problem of card collision. By showing the feasibility of this attack, we raise awareness of privacy and security issues in contactless payment protocols and implementation, specifically in the presence of new technologies for payment such as mobile platforms.

Third, we show that, apart from attacking mobile devices by having access to the sensors through native apps, we can also perform sensor-based attacks via mobile browsers. We examine multiple browsers on Android and iOS platforms and study their policies in granting permissions to JavaScript code with respect to access to *motion and orientation* sensor data. Based on our observations, we identify multiple vulnerabilities, and propose "TouchSignatures" and "PINLogger.js", two novel attacks in which malicious JavaScript code listens to such sensor data measurements. We demonstrate that, despite the much lower sampling rate (comparing to a native app), a remote attacker is able to learn sensitive user information such as physical activities, phone call timing, touch actions (tap, scroll, hold, zoom), and PINs based on these sensor data. This is the first report of such a JavaScript-based attack. We disclosed the above vulnerability to the community and major mobile browser vendors classified the problem as high-risk and fixed it accordingly.

Finally, we investigate human dimensions in the problem of sensor management. Although different types of attacks via sensors have been known

for many years, the problem of data leakage caused by sensors has remained unsolved. While working with W3C and browser vendors to fix the identified problem, we came to appreciate the complexity of this problem in practice and the challenge of balancing security, usability, and functionality. We believe a major reason for this is that users are not fully aware of these sensors and the associated risks to their privacy and security. Therefore, we study user understanding of mobile sensors, specifically their risk perceptions. This is the only research to date that studies risk perceptions for a comprehensive list of mobile sensors (25 in total). We interview multiple participants from a range of backgrounds by providing them with multiple self-declared questionnaires. The results indicate that people in general do not have a good understanding of the complexities of these sensors; hence making security judgements about these sensors is not easy for them. We discuss how this observation, along with other factors, renders many academic and industry solutions ineffective. This makes the security and privacy issues of mobile sensors and other sensor-enabled technologies an important topic to be investigated further.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Today sensors are everywhere: from your personal devices such as mobiles, tablets, watches, fitness trackers, and other wearables, to TVs, cars, kitchens and homes, as well as roads, parking lots, and smart cities. These new technologies are equipped with many different sensors, such as NFC, accelerometer, orientation and motion, and are connected to each other through the Internet of Things (IoT). These sensors are providing ever more features to end users enabling to interact with their real world surroundings. As a result, users are benefiting from richer and more personalised apps which use these sensors for different purposes such as fitness, gaming, and even security applications such as authentication. However, the growing number of sensors introduces new security and privacy risks to end users, and makes the task of sensor management more complex.

## 1.1 Mobile device sensors

According to the Economist [38], smartphones have become the fastest-selling gadgets in history, outselling personal computers four to one. Today about half the adult population owns a smartphone; by 2020, 80% will. In this thesis, the focus will be on the sensors of mobile devices, particularly smart phones and tablets. However, the problems and the solutions we investigate are generally applicable to other sensor-enabled technologies.

Mobile device vendors are increasingly augmenting their products with different types of sensors. Here we present a list of different sensors by inspecting the official websites of the latest iOS (iPhone 6[1]) and Android (Nexus 6P[2]) products, and the specifications that W3C[3] and Android [52] provide for developers. We also add some extra sensors (wireless technologies, camera, microphone, touch screen, and GPS) as common sensing mobile hardware. We categorise these sensors into four main groups: communicational sensors, identity-related (biometric) sensors, ambient (environmental) sensors, and movement sensors, as presented in Table 1.1. Note that this list can be even longer if all mobile brands are included. For example, the world's first thermal imaging sensor on mobile phones is offered by Cat S60 smartphone[4].

In the following, we present a brief description of each sensor:

- GPS: identifies the real-world geographic location.

[1] apple.com/uk/iphone-6/specs/
[2] store.google.com/product/nexus_6p
[3] w3.org/2009/dap/
[4] catphones.com/en-gb/phones/s60-smartphone

| Category | Sensors |
|---|---|
| Identity-related (Biometric) | GPS, Camera, Microphone, Fingerprint (TouchID), Touch Screen |
| Communicational | WiFi, Bluetooth, NFC |
| Ambient (Environmental) | Temperature (ambient, device), Humidity, Pressure (Barometer), Light, Proximity, Gravity, Magnetic Field, Hall Sensor |
| Movement | Gyroscope, Accelerometer, Rotation, Orientation, Motion, Sensor Hub |

Table 1.1: Categorisation of current mobile sensors

- Camera, Microphone: capture pictures/videos and voice, respectively.
- Fingerprint, TouchID: scans the fingerprint.
- Touch Screen: enables the user to interact directly with the display by physically touching it.
- WiFi: is a wireless technology that allows the device to connect to a network.
- Bluetooth: is a wireless technology for exchanging data over short distances.
- NFC (Near Filed Communication): is a wireless technology for exchanging data over shorter distances (less than 10 cm) for purposes such as contacless payment.
- Proximity: measures the distance of objects from the touch screen.
- Ambient Light: measures the light level in the environment of the device.
- Ambient Pressure (Barometer), Ambient Humidity, and Ambient Temperature: measure the air pressure, humidity, and temperature in the environment of the device, respectively.
- Device Temperature: measures the temperature of the device.
- Gravity: measures the force of gravity.
- Magnetic Field: reports the ambient magnetic field intensity around the device.
- Hall Sensor: produces voltage based on the magnetic field.
- Accelerometer: measures the acceleration of the device movement or vibration.
- Rotation: reports how much and in what direction the device is rotated.

- Gyroscope: estimates the rotation rate of the device.
- Motion: measures the acceleration and the rotation of the device.
- Orientation: reports the physical angle that the device is held in.
- Sensor Hub: is an activity recognition sensor and its purpose is to monitor the device's movement.

## 1.2 Smart apps

From an artificial intelligence point of view, sensors are added to mobile devices to make them "smart": to sense the surrounding environment and infer aspects of the context of use from the sensor data, and thus to facilitate more meaningful interactions with the user. Many of these sensors are used in popular mobile apps such as fitness training and games. Mobile sensors have also been proposed to use for security purposes, e.g. for authentication [20, 34], authorization [69], and device pairing [76]. In this vein, in Chapter 2, we propose to utilize sensors to build a novel "Tap-Tap-and-Pay" (TTP) mechanism [79] to prevent the *Mafia attack*.

**The problem.** The Mafia attack presents a realistic threat to payment systems including mobile NFC payment. In this attack, a user consciously initiates an NFC payment against a legitimate-looking NFC reader (controlled by the Mafia), not knowing that the reader actually relays the data to a remote legitimate NFC reader to pay for something more expensive. Two main categories of solutions have been commonly suggested for the Mafia attack: distance bounding protocols [13, 23, 37], and (ambient) sensor-based solutions [53, 73, 98]. Most distance bounding protocols defined in the literature [23, 37] are based on symmetric key encryption which requires the card and the reader to have a pre-shared symmetric key. In current practice, the card only has a pre-shared key with the issuing bank. On the other hand, the underlying assumption in the previous sensor-based solutions is that malicious and legitimate readers are in two different locations with distinct ambient environments. However, the validity of this assumption may not always hold in practice.

**Our solution.** In TTP, we address the above shortcomings by leveraging the characteristics of vibrations when an NFC card is physically tapped on an NFC reader. We observe that the accelerometer measurements produced by both devices are closely correlated within the same tapping event, while they are different if obtained from different tapping events. Hence by comparing the two measurements, the issuing bank in the back-end of the payment network can distinguish legitimate NFC payments

from Mafia frauds. This work serves to highlight the potential to make use of sensors to build useful security mechanisms.

## 1.3 In-app attacks

There is also the other side of sensors: access to the sensor streams provides an app running in the background with an exploit path. Researchers have shown that the user's PIN/password can be disclosed through sensors such as camera and microphone [99], ambient light [100], and gyroscope [115].

Sensors can also be misused to attack financial payments. Since the introduction of Google Wallet[5], almost all new smart phones have added NFC sensors. A malicious app (with permission to access the NFC sensor) can easily turn the phone into an NFC reader. This presents a realistic threat to users, as in practice many people place the smart phone in close proximity to their contactless bank cards (e.g. in purses). Once the card is within the NFC field of the phone, using techniques we present in Chapter 3, we demonstrate a privacy attack, "NFC Payment Spy" [78], on contactless payments by taking advantage of a situation called *card collision* [9, 15] or *card clash* [105].

**The vulnerability.** In a contactless transaction, a card collision happens when more than one card (or an NFC-enabled device) are presented to the payment terminal's field, and the terminal does not know which card to choose to proceed with the transaction. According to EMV [15] (the primary standard for smart card payment), once a collision is detected, the terminal should not proceed with the interaction; instead it should reset the field. On the other hand, ISO/IEC 14443 [9] specifies no termination in the case of a collision. Instead, a race condition is created in which depending on the implementation of the terminal and the UIDs of the cards available in the field, only one card would be selected.

**Our attack.** We observe that the implementation of contactless NFC readers in practice does not follow EMV's card collision algorithm, or ISO's. Our attack works based on this inconsistency; when the user aims to pay contactlessly while placing her card close to her phone, the attack app engages with the terminal before the card does. Accordingly, the attack app can retrieve from the terminal the Processing Options/ Data Object List (PDOL) data, which include information about the payment such as the amount and date. This attack is the first privacy attack on contactless payments based on the problem of card collision. By suggesting this attack we raise awareness

[5] wallet.google.com/

of privacy and security issues in the standard specifications and implementations of contactless cards and readers, especially with regard to the sensors available in modern mobile devices.

## 1.4 Sensor management challenges

**Over-privileged apps.** Today users spend much of their time consuming digital media within mobile applications [68]. Android is the most popular mobile OS with 84.7% market share as of 2015 Q3 [50], and the Google Play Store (as the largest and only official Android marketplace) is boasting in excess of 1.6 million apps [101]. According to [90], the average consumer uses over 26 different apps per month.

As shown in [103], the average number of permissions used by Android apps (installed from Google Play) increases over time, especially for popular apps as well as free apps. These permissions are requested for having access to the OS resources as well as sensors such as GPS, camera, and microphone. This has the potential to make apps over-privileged and unnecessarily increase the attack surface.

**Unmanaged sensors.** As pointed out by researchers in [115], the fundamental problem here is that "sensing is unmanaged on existing smartphone platforms". The in-app access to certain sensors including GPS, camera, and microphone requires user permission when installing and running the app. However, as discussed in [99], an attacker can easily trick a user into granting permission through social engineering (e.g. presenting it as a free game app). Once the app is installed, usage of the sensor data is not restricted. Even worse, access to many other sensors including accelerometer, gyroscope, and light is unrestricted; any app can have free access to the sensor data without needing any user permission, as these sensors are left unmanaged in operating systems. As it can be seen in Table 1.2, permission policies for having access to different sensors vary across sensors and platforms [6, 108].

**Relying on users and app stores.** Although the information leakage caused by sensors has been known for years [99,100,115], the problem has remained unsolved in practice. One main reason is the complexity of the problem (as we illustrate in Chapter 5). Another reason, from the practical perspective, is that all the reported attacks depend on one condition: the user must initiate the downloading and installing of the app. Therefore, users are relied upon to be vigilant and not to install untrusted apps. Furthermore, it is expected that app stores such as the Apple app store and Google Play will screen the apps, and impose severe penalties if the app is found to

| Sensor | Android | iOS | W3C |
|---|---|---|---|
| GPS | ✓ | ✓ | ✓ |
| Camera | ✓ | ✓ | ✓ |
| Microphone | ✓ | ✓ | ✓ |
| Fingerprint/ TouchID | ✓ | ✓ | NA |
| Touch Screen | ✗ | ✗ | ✗ |
| WiFi | ✓ | ✓ | ✗ |
| Bluetooth | ✓ | ✓ | ✓ |
| NFC | ✗* | Locked | ✗ |
| Proximity | ✗ | ✗ | ✗ |
| Ambient Light | ✗ | ✗ | ✗ |
| Ambient Pressure/ Barometer | ✗ | ✗ | NA |
| Ambient Humidity | ✗ | NA | NA |
| Ambient Temperature | ✗ | NA | NA |
| Device Temperature | ✗ | NA | NA |
| Gravity | ✗ | ✗ | NA |
| Magnetic Field | ✗ | ✗ | ✗ |
| Hall Sensor | ✗ | NA | NA |
| Accelerometer | ✗ | ✗ | ✗ |
| Rotation | ✗ | ✗ | ✗ |
| Gyroscope | ✗ | ✗ | ✗ |
| Motion | ✗ | ✗ | ✗ |
| Orientation | ✗ | ✗ | ✗ |
| Sensor Hub | Locked | Locked | NA |

Table 1.2: Current permission policies of sensors on different platforms, ✓: permission required, ✗: permission not required, NA: not supported, and Locked: not open to developers. *NFC should be turned on manually in Android for any program to be able to use it.

contain malicious content. However, in a new attack shown below we demonstrate that these measures are ineffective.

## 1.5 In-browser attacks

**The vulnerability.** In Chapter 4, we show that the sensor management problem is spreading from apps to browsers. W3C specifies standard APIs to allow in-browser access to certain sensors (e.g. GPS, light, motion and orientation) through JavaScript code. As shown in Table 1.2, mobile web browsers allow the JavaScript code in a web page to access motion and orientation sensor data without any user permission being required. The associated risks to user security and privacy are, however, not considered in the W3C specification. Nonetheless, browser vendors still took the

precaution to reduce the sensor rate available in-browser by a factor of 3 to 5 as compared with what is attainable in-app (e.g. in Chrome, the rate is reduced from 200 Hz to 60 Hz[6]).

**Our attacks.** We introduce two JavaScript-based attacks: "TouchSignatures" and "PINLogger.js" [80–83], and demonstrate that despite the much lower sampling rate, a remote attacker is still able to learn the user's touch actions (i.e./ tap, scroll, hold, and zoom) and PINs based on reading the motion and orientation sensor data. In contrast to previous attacks, our JavaScript-based attack does not require any app installation. Once the malicious web page is opened, it can covertly read the sensor streams without needing any user permission. The attack works when the JavaScript code is embedded in an iframe (this is where the third-party advertisement is often hosted) instead of the main page. Depending on the mobile platform and browsers, the attack continues to work when the attacker's web page is left open in an inactive tab, when the browser is minimized, or even when the screen is locked. These vulnerabilities present a serious threat to the end user's privacy and security.

**Reporting to industry.** Following responsible disclosure practices, we informed W3C and browser vendors in private of our findings. W3C acknowledged: "This would be an issue to address for any future iterations on this [W3C] document." All major mobile browser vendors, including Google Chrome, Mozilla Firefox, Apple Safari and Opera, have also acknowledged the problem we identified. For example, Mozilla replied, "Indeed, and it should be fixed consistently across all the browsers and also the spec [W3C specification] needs to be fixed."

## 1.6 Industrial vs. academic approaches

**Industrial approach.** As mentioned earlier, some mobile sensors such as accelerometer and gyroscope are unrestrictedly accessible through both native apps and browsers without any permissions. For other sensors such as GPS and camera, mobile platforms and browsers have adopted different approaches for granting permissions. For example, run-time permissions are granted by users in iOS, while it used to be install-time permission requests for Android before Android 6[7]. In addition, although the run-time approach offers more control, but many users continue to accept permission requests blindly possibly due to a lack of understanding [39, 47, 48].

[6] bugs.chromium.org/p/chromium/issues/detail?id=421691

[7] developer.android.com/training/permissions/requesting.html

**Academic approach.** A range of solutions to address the in-app access attacks have been suggested in the literature: e.g. restricting the sensor to one app, reducing the sampling rate, temporal pause of the sensor on sensitive entries such as keyboard, rearranging keyboard for password entry, asking for explicit permission from the user, ranking apps based on their similarities to malware, and obfuscating anomalies in sensor data [17,21,33,84,85,88,92,99,100,115]. However, after many years of research, none of the major mobile platforms have revised their access policy. While the number of sensors is increasing on mobile platforms, the risk of the newer sensors such as movement sensors (accelerometer, gyroscope, etc.) and ambient sensors (pressure, temperature, etc.) to users privacy and security seems to be still underestimated.

**Our countermeasures.** When reporting our in-browser attack [81] to the industry, we suggested several countermeasures, including: 1) to treat the motion and orientation sensors with the same sensitivity as the GPS sensor; 2) to disable the JavaScript access to sensors when the JavaScript code in the iframe has a different origin from the main page, or when the view of the web page is hidden. Our first suggestion is consistent with what has been recommended in many previous academic papers: namely, requiring user permission to access the sensor data. However, relying on user prompts is not considered a usable solution by the industrial community.

In the end, browser vendors have chosen to adopt the second countermeasure we suggested in [81]. For example, starting from version 46 (released in April 2016), Firefox restricts JavaScript access to motion and orientation sensors to only top-level documents and the same-origin iframe. In the latest Apple Security Updates for iOS 9.3 (released in March 2016), Safari took a similar countermeasure by "suspending the availability of this [motion and orientation] data when the web view is hidden." Moreover, W3C has drafted a new version of the motion and orientation specifications including a security section as suggested by us.

**Limitation.** However, we believe the second countermeasure should only serve as a temporary fix rather than the ultimate solution. In particular, it has the drawback of removing potentially useful web applications in the future. For example, a web page running a fitness program has a legitimate reason to access the motion sensors even when the web page view is hidden. However, this is no longer possible in the new versions of Firefox and Safari. Our concern is shared by the Google Chromium team. As stated by one member of the Chromium team on the online Bug Track of the issue, "We're particularly concerned about preventing interesting future use cases. We suspect we could copy Safari's behavior without breaking too much existing content,

but preventing new forms of content (like spherical video) is a serious risk"[8].

**Necessity of sensor management.** In future, we expect the boundary between apps and web programs on mobile devices will gradually diminish. Many native apps will move towards web-based programs so they do not require client installation. This is similar to the trend in the PC era that many installable programs migrate to be web based so no client software installation is needed. With the growing number of sensors, and more sensitive sensor hardware provisioned with new mobile devices, the problem of information leakage caused by sensors is expected to become more severe. In view of all this, we believe a systematic solution to securely manage the sensors in both apps and browsers is urgently needed.

## 1.7 Human dimensions

While working with W3C and browser vendors to fix the identified problem, we came to appreciate the complexity of the sensor management problem in practice and the challenge of balancing security, usability and functionality. We believe a major reason for this is that users are not aware of i) the data generated by the sensors, ii) how that data might be used to undermine their security and privacy, and iii) what precautionary measure they could and should take. As stated by the chair of the Geolocation Working Group in W3C (the working group which is in charge of motion and orientation standard), "the need for research into balancing usability of sensor data with privacy and security is acute, given explosion in mobile applications and websites that leverage access to such data".

In Chapter 5, we study users' perception of the risks associated with mobile phone sensors [80, 83]. We design a few user studies to measure the general familiarity with different sensors and their functionality, and to investigate how concerned users are about their PIN being stolen by an app that has access to all these sensors. Our results show that mobile users are not generally familiar with mobile sensors, specifically with the newer ones. Moreover, we observe that there is significant disparity between the actual and perceived levels of the threat with regard to the compromise of the user PIN. We discuss how this observation, along with other factors, renders many academic and industry solutions ineffective in managing mobile sensors. This result has significant implications on the appropriate interaction design strategy for a sensor management system. For example, it might indicate that relying on user prompts to manage sensors is inadequate.

[8] bugs.chromium.org/p/chromium/issues/detail?id=523320#c18

We believe that, in conjunction with more large scale user studies, a solution that adopts a human-centred approach to designing a sensor management system would address this issue. We leave this as future work and encourage the researchers in the field to pay more attention to security and privacy issues associated with sensors.

## 1.8 Methodology

The research in this thesis is driven by tackling real-world security problems. Throughout this research, we worked on practical problems concerning security and privacy issues of mobile users based on sensors. In order to demonstrate the feasibility of our ideas in practice, we have performed all of our experiments by using off-the-shelf devices and tools. In Chapters 2, 3, and 4 of this thesis, we develop application-level attacks and designs, and in Chapter 5, we conduct users studies to directly investigate some of the human dimensions of the sensor technology.

**Data collection from mobile users.** All of our proposed attacks and designs follow an experimental method for collecting data from real mobile users. In Chapter 2, when designing TTP as an NFC payment protocol, we simulate a contactless payment activity and augment our proposed Tap-Tap gesture in a payment app in the lab. This experiment is very similar to a real NFC payment in terms of the procedure. The only difference is that we use a mobile phone as a reader which is not connected to the bank and does not charge the simulated card on the phone. We collect the sensor data (accelerometer measurements) during multiple Tap-Tap and Pay activities being performed by volunteer users. Furthermore, when conducting user studies to compare TTP with a conventional contactless payment, we again simulate both activities as close as possible to the implementation of contactless cards and readers in a real-world payment system.

In Chapter 4, we perform a series of attacks on sensitive user information such as touch actions and PINs. We set our data collection configuration in a way that allows our users to work with a mobile phone in an everyday manner. While our users are completing the required task (working with the mobile screen by using different touch actions such as click, scroll, and type, and entering PINs), our JavaScript code sends the sensor measurements (motion and orientation) to the server. In different phases of our attacks in Chapter 4, multiple volunteer users have been working with iPhone and Android devices using Chrome.

Before running any experiment in this research, we gain approval through Newcastle University's research ethics processes. During different data collections, we

present a description of our studies to the users. These description presentations can be either oral, printed papers, video guides, or combined methods as we explain in each chapter in detail. The reported results in different chapters of this thesis are based on the above data collection approach originated from real mobile users.

**Experiments in the wild.** When possible, we extend our lab experiments to the wild. In Chapter 3, when investigating the card collision problem, we perform all of our experiments in different metro stations. We use pairs of real bank cards and test each pair in a race condition when both are presented to the terminals (ticket machines) in each metro station. Moreover, when performing our attack by attaching the bank cards on the back of an NFC-enabled phone, we again test the experiments in the wild. We put each card and the phone (simulated card app) in a race condition and observe the behaviour of the terminal. The results of these real-world experiments are presented in Chapter 3.

**Communication with the industry.** In each phase of this thesis, not only do we study the literature, but we also investigate the state-of-the-art practical solutions offered by the industry. By following this approach, we carefully go through the standards and specifications available for each technology. For example in Chapters 2 and 3, we read over a thousand pages of EMV and ISO documents on contactless payment technology. Similarly in Chapters 4 and 5, we studied sensor-related specifications presented by W3C.

Furthermore due to the evolving nature of mobile sensors, while conducting some parts of this research, we followed many different forum discussions and mailing lists accommodated by the mobile browser vendors and standardisation bodies. We continuously were in contact with the industry (namely EMV, MasterCard, ISO, W3C, Qualcomm, Intel, Mozilla Firefox, Google Chrome, Apple Safari and Opera) via mailing-lists, forums, and in some cases private emails. W3C is interested in github discussions (and sometimes via the mailing-lists) when the team members are drafting new (versions of the) specifications. We actively followed the up-to-date changes and engaged in the process. Similarly, our communication with Firefox and Chrome has been mainly through Bugzilla (the Mozilla's bugtracker), and Chromium bugs website, respectively. We find that forums and mailing-lists are very effective channels to interact with technical and non-technical group members from different teams. These forum discussions and emails are normally publicly available, and are considered as track records of (un)solved problems. Researchers actively follow these forum discussions to gain better insight on the issues which have been discussed by experts

previously. Due to our continuous and active connection with the industry, we managed to contribute toward fixing some of the problems of the sensor technology as we present in Section 1.10.

Being in contact with the industry has other benefits for academic researchers too. For example, after Google Chrome team expressed their concern on the usability issues of some of the suggested solutions (limiting sensor access on mobile platforms and browsers), we were inspired to specifically study some of the human dimensions of the sensor technology. We dedicate Chapter 5 of this thesis to such study. The results of our user studies in this chapter has been shared with the community to be considered as expert opinion on their ongoing sensor-related projects [9].

## 1.9 Contributions

The work presented in this thesis makes a number of contributions:

- We propose "Tap-Tap and Pay" (TTP) as a new countermeasure to prevent Mafia attacks in contactless payment. We present a proof-of-concept implementation of TTP by using a pair of NFC-enabled smartphones. We also conduct user studies to evaluate the usability of our TTP prototype. The results show that our solution can effectively prevent the Mafia attack in contactless payments. Details of this work are presented in Chapter 2.

- We perform experiments to discover the behaviour of contactless terminals when a card collision occurs. We show that, due to the inconsistency between the implementation of contactless terminals and EMV protocols, it is possible to track the user's contactless payment activities, for instance through a malicious app. We propose an attack named "NFC Payment Spy" following EMV contactless specifications. We develop an Android app and perform experiments to support our claim. The results show that our attack can effectively break users' privacy and discover the pattern of their contactless payment activities. This work is demonstrated in Chapter 3.

- We examine multiple popular browsers on mobile platforms for their policies in granting permissions to JavaScript code with respect to access to orientation and motion sensor data. Based on our findings, we propose "TouchSignatures" and "PINLogger.js" which include attacks that compromise user security through

[9] w3.org/2000/09/dbwg/details?group=43696&public=1

malicious JavaScript code by listening to these sensor data streams. Our attacks are designed in three phases: 1) identifying the touch actions (e.g. tap, scroll, hold, and zoom), 2) identifying the PIN digits, and 3) identifying full 4-digit PINs. We demonstrate the practicality of the above attacks by collecting data from real users and reporting high success rates using our proof-of-concept implementations. These attacks are illustrated in Chapter 4.

- We conduct user studies to investigate users' understanding about mobile sensors and also their perception of the security risks associated with them. We show that users in fact have fewer security concerns about mobile movement sensors (motion and orientation) as compared to better known examples such as camera and microphone. We also study and challenge current suggested solutions by the industry and academia, and discuss why our studies show they cannot be effective. We argue that a usable and secure solution is not straightforward and requires further research. Details of this work are presented in Chapter 5.

## 1.10 Industrial impact

During this PhD research, we have been in close contact with the mobile industry via bug forums, github, mailing lists, and private emails. As mentioned before, we have disclosed the vulnerability that we discovered in Chapter 4 to W3C and all major mobile browser vendors, and contributed toward fixing this problem. As a direct result of this work:

- Firefox deployed the fix based on our proposed solution and released it in Firefox 46, with the unique Vulnerabilities and Exposures ID of CVE-2016-2813[10].

- Apple included a fix based on our work in iOS 9.3, with the unique Vulnerabilities and Exposures ID of CVE-2016-1780[11].

- The World Wide Web Consortium (W3C), the main international standards organisation for the World Wide Web, has drafted a new version of the motion and orientation specifications which includes a security section as suggested by us and citing our work[12].

[10] mozilla.org/en-US/security/advisories/mfsa2016-43/
[11] prod.lists.apple.com/archives/security-announce/2016/Mar/msg00000.html
[12] w3c.github.io/deviceorientation/spec-source-orientation.html#security-and-privacy

## 1.11 Media coverage

The work described in Chapters 4 and 5 attracted considerable national and international media coverage (newspaper, radio, tv, online news, social media, etc.) in many different languages. Our journal paper [83] associated with this news, covering the PIN attacks and the user studies, was downloaded 10,000 times within a week of its publication. Examples of the media articles and interviews include:

- The Guardian[13]: "Tilted device could pinpoint PIN number for hackers, study claims", by Alex Hern, 11 April 2017.
- BBC[14]: "The way people tilt their smartphone can give away passwords and PINs", BBC Newsbeat, 11 April 2017.
- The Telegraph[15]: "How the way you hold your smartphone could allow hackers to steal your bank details", by Henry Bodkin, 11 April 2017.
- The Independent[16], "PINs and passwords can be stolen just by watching the way a phone tilts, scientists find", by Andrew Griffin, 10 April 2017.
- Mail Online[17]: "How the way you tap a phone can give hackers your PIN: Criminals can guess four-digit code by using software to monitor movements", by Sean Poulter, 11 April 2017.
- New York Post [18] via The Sun[19]: "The way you hold your phone could get you hacked" and "CYBER-SPY WARNING Hackers could discover your phone password by analysing the way the device tilts while you hold it", by Jasper Hamill, 11 April 2017.
- The Australian[20] via The Times: "How tilting your phone could let in hackers", 11 April 2017.

[13] theguardian.com/technology/2017/apr/11/tilted-device-could-pinpoint-pin-number-for-hackers-study-claims

[14] bbc.co.uk/newsbeat/article/39565372/the-way-people-tilt-their-smartphone-can-give-away-passwords-and-pins

[15] telegraph.co.uk/science/2017/04/10/smartphone-motion-sensors-provide-backdoor-hackers-steal-bank/

[16] independent.co.uk/life-style/gadgets-and-tech/news/pin-password-iphone-apple-android-security-safety-privacy-study-newcastle-university-a7677321.html

[17] dailymail.co.uk/news/article-4399932/How-way-tap-phone-hackers-PIN.html

[18] nypost.com/2017/04/11/the-way-you-hold-your-phone-could-get-you-hacked/

[19] thesun.co.uk/tech/3306078/hackers-could-guess-your-phone-password-by-analysing-the-way-the-device-tilts-whilst-you-hold-it/

[20] theaustralian.com.au/news/world/the-times/how-tilting-your-phone-could-let-in-hackers/news-story/83846b772b23f474a137a9f3e08a970c

- The Economic Times (the world's second-most widely read business newspaper, after the Wall Street Journal)[21]: "The way you type on your smartphone can help hackers steal your bank details", 11 April 2017.

- TechCrunch (a leading online publisher of technology industry news)[22]: "Researchers demonstrate how PINs and other info can be gathered through phone movement", by Brian Heater, 10 April 2017.

- Engadget (a multilingual technology blog network)[23]: "Your phone's motion sensors can give away PINs and passwords", by Mariella Moon, 12 April 2017.

- XDA Developers (a mobile software development community of over 6.6 million members worldwide)[24]: "Malicious JavaScript Code Can Steal PIN Codes via Motion Sensors", by Doug Lynch, 19 April 2017.

- Popular Science (an American bi-monthly magazine)[25]: "A neural network helped researchers crack smartphone PINs using built-in motion sensors", by Rob Verger, 11 April 2017.

- Science Friday (weekly radio talk show, recording in New York City)[26]: "Sensing Steps, And Perhaps Your PIN", 14 April 2017.

- CBC News (Canadian Broadcasting Corporation)[27]: "Mobile phone motion sensors can be used to crack your PIN", by Brandie Weikle, 12 April 2017.

- German Public Radio Deutschlandfunk[28]: "Wenn die Sensoren zum Sicherheitsrisiko werden (When the sensors become a safety risk)", interviwed by Manfred Kloiber, 15 April 2017.

- Die Welt (a German national daily newspaper)[29]: "Wie du dein Handy haltst, verrat PIN und Passworter (How you hold your mobile phone reveals PIN and passwords)", by Philipp Nagels, 11 April 2017.

[21] economictimes.indiatimes.com/magazines/panache/the-way-you-type-on-your-smartphone-can-help-hackers-steal-your-bank-details/articleshow/58125226.cms

[22] techcrunch.com/2017/04/10/pin-gathering-mobile/

[23] engadget.com/2017/04/12/phone-motion-sensor-pin-vulnerability/

[24] www.xda-developers.com/malicious-javascript-code-can-steal-pin-codes-via-motion-sensors/

[25] popsci.com/sensors-in-your-smartphone-could-be-its-weakest-link

[26] sciencefriday.com/segments/sensing-steps-and-perhaps-your-pin/

[27] cbc.ca/news/technology/mobile-phone-sensor-spy-1.4067407

[28] deutschlandfunk.de/smartphones-wenn-die-sensoren-zum-sicherheitsrisiko-werden.684.de.html?dram:article_id=383916

[29] welt.de/kmpkt/article163617553/Wie-du-dein-Handy-haeltst-verraet-PIN-und-Passwoerter.html

- Sina[30] and Sohu[31] (the two largest Chinese-language web portals): "Phone gyroscope or spoiler information! Look at the tilt angle to guess the password" and "The phone sensor may reveal your password", 13 April 2017.

- Mashable (one of the top a digital media website in French and English)[32]: "Des scientifiques peuvent deviner un code PIN en analysant les mouvements de votre smartphone (Scientists can guess a PIN code by analyzing the movements of your smartphone)", by Par Charlotte Viguie, 11 April 2017.

- Lavoz (a leading daily Spanish-language newspaper published in Argentina)[33]: "Dificil pero no imposible: el modo en que se inclina el celular al usarlo puede revelar contrasenas (Difficult but not impossible: the way the cellphone is tilted when using it can reveal passwords)", by Agencia Telam, 12 April 2017.

- Khorasan (a widely read Persian national newspaper published in Iran)[34]: "The possibility of hacking mobile PINs using mobile movements", 19 April 2017.

- Newcastle University Press Office[35]: "How criminals can steal your PIN by tracking the motion of your phone", 11 April 2017.

As a result, the work in this thesis has helped raise awareness amongst everyday users of the security and privacy risks involved in using sensor-rich devices. To contribute more to the public knowledge about sensor security, we also organised a workshop entitled: "What your sensors say about you?" on managing sensors on mobile devices. This workshop was hosted by Thinking Digital Women 2016[36]. In addition, we have provided two articles entitled: "Is your mobile phone spying on you?" and "Auditing your mobile app permissions" in the *Cyber Security: Safety at Home, Online, in Life* online course[37], part of Newcastle University's series of massive open online courses (MOOCs). This free online course, aimed at providing people with the knowledge to make informed security and privacy decisions in the modern world, has already had 20,000 subscribers in two rounds in 2016 and 2017.

[30] tech.sina.com.cn/it/2017-04-13/doc-ifyeimqc3202830.shtml
[31] sohu.com/a/133724154_114877
[32] mashable.france24.com/tech-business/20170411-smartphone-capteur-mouvement-code-pin-acces
[33] lavoz.com.ar/tecnologia/dificil-pero-no-imposible-el-modo-en-que-se-inclina-el-celular-al-usarlo-puede-revelar-co
[34] khorasannews.com/Newspaper/MobileBlock?NewspaperBlockID=571265
[35] ncl.ac.uk/press/news/2017/04/sensors/
[36] tdcwomen.com/workshops/
[37] futurelearn.com/courses/cyber-security

# Chapter 2

# Preventing the Mafia Attack in NFC Payment

## 2.1 Chapter overview

Mobile NFC payment is already very popular among mobile users. As has been estimated, nearly 150 million consumers will make NFC mobile payments in 2016 [1]. The Mafia attack presents a realistic threat to payment systems including mobile NFC payment. In this attack, a user consciously initiates an NFC payment against a legitimate-looking NFC reader (controlled by the Mafia), not knowing that the reader actually relays the data to a remote legitimate NFC reader to pay for something more expensive.

In this work, we present "Tap-Tap and Pay" (TTP), to effectively prevent the Mafia attack in mobile NFC payment. In TTP, a user initiates an NFC payment by physically tapping her mobile phone against the reader twice in succession. The physical tapping causes transient vibrations at both devices, which can be measured by the embedded accelerometers. Our experiments indicate that the two measurements are closely correlated if they are from the same tapping, and are different if obtained from different tapping events. By comparing the similarity between the two measurements, we can effectively tell apart the Mafia fraud from a legitimate NFC transaction. To evaluate the practical feasibility of this solution, we present a prototype of the TTP system based on a pair of NFC-enabled mobile phones and also conduct a user study. The results suggest that our solution is reliable, fast, easy-to-use and has good potential for practical deployment.

Most parts of the work in this chapter was published as follows under the supervision of Dr. Hao and Dr. Shahandashti. Some extra details about the TTP protocol are available in this chapter.

- M. Mehrnezhad, F. Hao, and S. F. Shahandashti, "Tap-Tap and Pay (TTP): Preventing Man-In-The-Middle Attacks in NFC Payment Using Mobile Sensors", In the Proceedings of the Second Conference of International Security Standardisation Research, SSR 2015, Tokyo, Japan, December 15-16, 2015, Springer International Publishing, Pages 21-39.

## 2.2 Introduction

In this section, first we present an overview on NFC payment. Next, we present the Mafia attack in detail. We then give an overview on NFC payment standards. Finally, after presenting the current suggested solutions for this attack, we describe our solution to this problem.

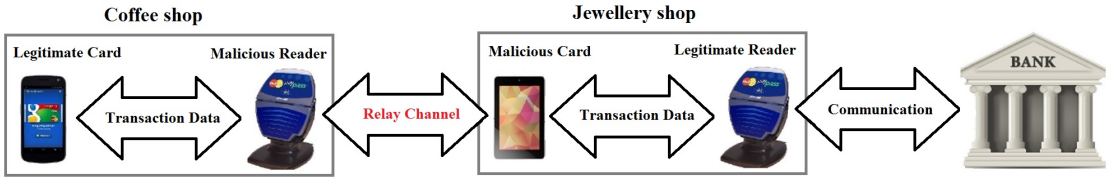


Figure 2.1: The Mafia attack: a malicious reader colludes with a malicious card and fools the honest card into paying for something more expensive to a legitimate reader.

### 2.2.1 NFC payment

Near Field Communication (NFC) payment is an upcoming technology that uses Radio Frequency Identification (RFID) to perform contactless payments. An RFID system has two parts: the RFID tag (card) that can be attached to any physical object to be identified and the RFID reader that can interrogate a tag within physical proximity, via radio frequency communication. An NFC-enabled payment card has an embedded RFID tag. To make an NFC payment, the user just needs to hold the card in front of an NFC reader for a short while and wait for confirmation. NFC payments are usually limited to rather small-value purchases[1].

A mobile phone can also be used as an NFC payment card. HSBC Hong Kong Mobile Payment[2], Google Wallet[3], Apple Pay[4], and Android Pay[5] are examples of NFC payment mobile apps. Using a mobile phone for NFC payment is considered convenient since people can save all of their cards in their phones. As estimated, nearly 150 million consumers will make NFC mobile payments in 2016 [1]. To support this trend, new generations of smart phones are commonly equipped with NFC sensors. In this work, we focus on mobile payment using NFC. Hence, unless stated otherwise, by "NFC card", we refer to an NFC-enabled mobile phone functioning as a payment card. By "NFC reader", we refer to a payment terminal that communicates with the card via NFC. A *legitimate* NFC reader is one that is authorised by the banking network and is connected to the back-end banking network for payment processing.

It is known that NFC payments are vulnerable to different types of Man-In-The-Middle (MITM) attacks [49], also known in the literature as relay or wormhole attacks [37]. In a simple form of relay attack known as a ghost-and-leech attack [53], the attacker places an NFC reader so as to secretly interrogate the user's NFC card

[1]For instance, the contactless limit increased from £20 to £30 in 2015 in the UK.
[2]hsbc.com.hk
[3]wallet.google.com
[4]apple.com/iphone-6/apple-pay
[5]android.com/intl/en_us/pay

without the user's awareness, and relays the card response to a remote NFC reader to obtain a payment from the victim's account. Such an attack is demonstrated in [41] and [49].

Relay attacks can be countered in a number of ways. A simple solution is to put the NFC card within an NFC protective shield such as Id Stronghold[6]. Equivalently, one can add an activation button so that the NFC function on the phone is only turned on with an explicit user action. More advanced countermeasures are proposed in the literature, including *Secret Handshakes* [32], *UWave* [72], *Still and Silent* [97], and *Rub* [69]. However, none of these solutions can prevent a more severe type of attack, as we explain below.

### 2.2.2 Mafia attack

Another type of MITM attack is called the Mafia attack, also known as Mafia fraud [37] or the reader-and-ghost attack [53, 98]. In this more severe attack, the user consciously initiates an NFC payment with a legitimate-looking reader controlled by the Mafia; but the reader actually relays the card response to a remote legitimate NFC reader — via a malicious card — to pay for something more expensive. Figure 2.1 shows an example of such an attack. This attack has been shown to be feasible [37].

Unlike simple relay attacks, the Mafia attack cannot be prevented by using a protective shield or an activation button since the user consciously initiates the payment. For the same reason, various user-movement-based unlocking mechanisms [32, 69, 72, 97] cannot stop the attack either. We will explain the current countermeasures to this attack by first reviewing the NFC payment standards and specifications.

### 2.2.3 NFC payment standards and specifications

EMV is the primary protocol standard for smart card payments in Europe. The EMV standards are managed by EMVCo[7], a consortium of multinational companies such as Visa, Mastercard, and American Express. These standards use smart-cards including contact and contactless cards and are based on ISO/IEC 7816 [16] and ISO/IEC 14443. Mobile NFC payment technologies, such as Android Host-based Card Emulation (HCE) [4], are also based on ISO/IEC 14443, which is an international standard in four parts, defining the technology-specific requirements for proximity cards used for identification [7–10].

[6] www.idstronghold.com
[7] emvco.com

The extensive EMV specifications — presented in 10 books: A [11], B [12], C1–C7 (e.g. [13, 14]), and D [15] — provide the details of EMV-compliant payment system design. Furthermore, EMVCo provides a book on security and key management [43] as part of the EMV 4.3 specifications, as well as additional security guidelines for acquirers [44] and issuers [45] of EMV payment cards.

The risk of MITM attacks in payment systems has generally been neglected in the above standards and specifications (except in a recent 2015 EMV Contactless payment specifications Book C-2 [13], as we will explain). As has been explained by Drimer et al. in [37], such attacks are commonly perceived to be too expensive to work. However, in the same paper, Drimer et al. show this is a misperception by demonstrating practical MITM attacks in a set of live experiments against the UK's EMV system. Given the practicality of deploying such attacks [37] and the projected rapid growth in the size of the contactless payment industry [2], we believe that it is important for the payment industry to seriously consider the security concerns posed by such attacks and the countermeasures that are needed.

### 2.2.4 Distance bounding protocols

Distance bounding protocols have been considered a potential solution to this problem. In the latest MasterCard EMV specifications (Book C-2 [13] released in March 2015), a distance bounding protocol (called the Relay Resistance Protocol in the specifications) is defined. This protocol starts with the reader sending the card a random challenge and the card replying with a digitally signed response. The reader verifies the digital signature and also checks the response time is within a specified range. This protocol requires an additional private key and a public key certificate to be installed on the card. Furthermore, the card needs to perform expensive public key operations, which may incur a notable processing delay. To minimize the processing delay on the card, most distance bounding protocols defined in the literature [23, 37] only use symmetric key operations, such as hash and symmetric-cipher encryptions. However, applying those solutions to NFC payment would require the card and the reader to have a pre-shared symmetric key. In current practice, the card only has a pre-shared key with the issuing bank. By contrast, our solution does not require any additional cryptographic keys. In fact, it is orthogonal to distance bounding protocols and can be used in conjunction with any of them.

### 2.2.5 Other countermeasures

Other countermeasures to the MITM attack have been actively explored by a number of researchers. One straightforward solution is to require user vigilance at the time of making the NFC payment. However, it has been generally agreed that user vigilance alone is not sufficient [53, 73, 98]. It is desirable to design a countermeasure that can effectively prevent Mafia attacks without having to rely on user vigilance. Current solutions generally involve using ambient sensors to measure the characteristics of the surrounding environment, such as light [53], sound [53], location via GPS [73] and a combination of temperature, humidity, precision gas, and altitude [98]. The underlying assumption is that the malicious and legitimate readers will be in two different locations with distinct ambient environments. However, the validity of this assumption may be challenged in some situations where the two readers are in similar environments (e.g. nearby stalls in the same mall).

### 2.2.6 Contributions

Our idea is based on the following observation: as a result of the physical tapping between a pair of devices (a card and a reader quipped with accelerometers), the tapping creates transient vibrations, which can be measured using embedded accelerometer sensors. By comparing the similarity of the two measurements, we are able to determine if the two devices were involved in the same tapping event. This effectively distinguishes the Mafia attack from a normal NFC transaction.

In contrast to the solutions described above, we do not assume that the attacker's reader is in an environment different from that of the legitimate reader. Thus our threat model considers a more severe attack.

Our main contributions are summarised below:

1. We propose "Tap-Tap and Pay" (TTP) as a new countermeasure to prevent Mafia attacks. Our solution is the first that works even if the malicious and legitimate readers are in similar environments.

2. We present a proof-of-concept implementation of TTP by using a pair of NFC-enabled smart phones. Experiments confirm that vibrations induced from the same tapping event are closely correlated between the card and the reader, while they are not if originating from different tapping events.

3. We conduct user studies to evaluate the usability of our TTP prototype. Based on the feedback, users generally find the suggested solution fast and easy to use.

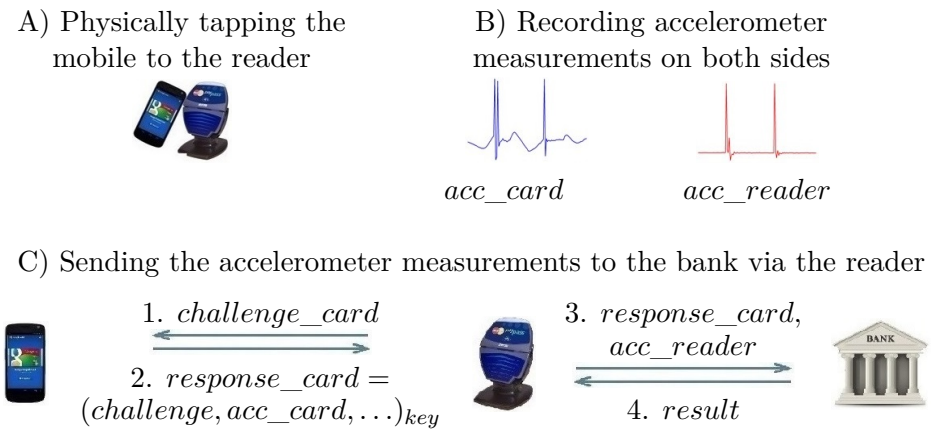


Figure 2.2: Overview of the proposed solution: Tap-Tap and Pay

## 2.3 Tap-Tap and Pay (TTP)

In this section, first we present the threat model and the overview of our system. Then, we demonstrate the sensor data processing steps. Finally, we explain our suggested comparison method.

### 2.3.1 Threat model

We assume a user consciously initiates an NFC payment against a legitimate-looking but malicious NFC reader without realizing that it is controlled by the Mafia. The difference between a malicious reader and the legitimate reader is that the former is not connected to the back-end banking network while the latter is. We assume the Mafia does not want to directly connect to the banking network, as that will run the risk of being caught by the bank. The malicious reader relays the victim's card to a remote legitimate reader to pay for something more expensive, through the help of an accomplice who holds a legitimate-looking NFC card (see Figure 2.1). From the perspective of the legitimate merchant, there is nothing suspicious — a customer uses a mobile phone to make an NFC payment. The amount of the payment may be near the upper end of the limit, but that is perfectly acceptable (see [37] for a demonstration of successful Mafia attacks on the UK's EMV payment system using contact chip-and-PIN cards; the attacks on the contactless payment work in the same way).

Furthermore, we assume the attacker is able to put the NFC reader in an ambient environment that is very similar to the legitimate reader. In one scenario, the attacker sets up a mobile temporary stall near a shopping mall. He may pretend to sell cheap items such as coffee, tea or confectionery, and shows the buyer a small amount on the

reader's screen. While accepting the buyer's NFC payment, the attacker relays it to one of his accomplices in nearby shop to buy something more expensive. The attacker and his accomplices can avoid detection by constantly changing their location. Once they have made enough profit in a day, they will disappear and repeat the same attack at a different place. Under the above threat model, previous ambient-sensor-based solutions may fail completely. However, despite the assumption of a stronger attacker, we will present a solution that can effectively prevent Mafia attacks under the same condition.

The practical feasibility of such Mafia attacks [37], compounded by the fact that they are undetectable by banks in the backend, can prove problematic. This can have serious implications on the liability of the cardholder and merchant if the security of the system only depends on user vigilance. In practice, if any dispute arises regarding the discrepancy of the amount charged for an NFC payment, users might be to blame by service providers since they are expected to be "vigilant". We believe this is not fair to users. Our solution addresses this problem by providing banks more evidence so they can distinguish a legitimate NFC payment from a Mafia fraud. This is achieved with minimum inconvenience to users, as we explain in the next section.

### 2.3.2 Overview of the solution

An overview of our solution is shown in Figure 2.2. First, the user physically taps the mobile phone against the reader twice to make an NFC payment. The tapping causes transient vibrations at both devices, which are measured by the embedded accelerometer sensors. The user then holds the card close to the reader. At this point, the reader detects the presence of an NFC card within physical proximity and starts a standard challenge-and-response process for the NFC payment. At a high level, this involves the reader sending a challenge to the card, and the card replying with a response authenticated via a MAC computed using a secret shared-key with the issuing bank. Our solution does not alter this existing data flow; however within the card response, we propose to add an additional item *acc_card* to the items being sent by the card. This new item represents the measurement of the vibration by the card accelerometer. When the reader forwards the card's response to the issuing bank through a secure back-end network, it appends *acc_reader*, which is the measurement of the vibration by the reader accelerometer. The bank compares the two measurements and approves the transaction only if it finds the two sufficiently similar. Recall that in Figure 2.1, the user's NFC card and the legitimate NFC reader are honest devices and can perform trustworthy measurements.

TTP suggests two taps because we found it to be the minimum number of taps needed to obtain both sufficiently correlated measurements of the same tapping, and at the same time sufficiently uncorrelated measurements of different tappings. We performed an informal experiment and observed that with one tap we would not achieve the desired accuracy. Similarly after a quick test with more than two taps, as we expected, more features could be extracted, but of course this is at the expense of user convenience. Hence, we chose double-tap as the default setting for our solution and leave further experiments with more taps to future work.

### 2.3.3 Host-based card emulation and Reader emulation

To enable data collection, we developed two Android apps: *Card app* and *Reader app* and installed them on two NFC-enabled smart phones, two Nexus 5 devices[8], which are equipped with a range of different sensors.

Prior to the release of Android 4.4, to emulate a smart card on an Android device, it was required to root the device in order to get access to the Secure Element, a hardware chip capable of performing sensitive cryptographic operations. Android 4.4 introduced host-based card emulation (HCE) as a new method of card emulation that does not involve the secure element. This allows us to develop a card application capable of talking to any NFC reader directly in the way that a smart card does — by holding it in front of a reader[9]. Android 4.4 supports several protocols used by mainstream NFC readers in the market today. Furthermore, NFC-equipped Android devices can function as reader as well. This means that by emulating a card through HCE on one hand and developing an NFC reader on the other hand, we were able to build a complete NFC payment system using only Android devices.

Android 4.4 supports emulating cards that are based on the NFC-Forum ISO-DEP specification (based on ISO/IEC 14443-4) and process Application Protocol Data Units (APDUs) as defined in the ISO/IEC 7816-4 specification. In compliance with the ISO/IEC 7816-4 specification, each HCE application has an Application ID (AID). This ID enables the reader app to select the correct service.

[8]Prototyping of our TTP protocol requires the facility of bidirectional NFC using Host-based Card Emulation (HCE). At the time of experiments, Nexus 5 was the only device allowing that facility.

[9]Note that Android API guide [4] uses "tapping the card against the NFC reader" and "holding the device over the NFC reader" interchangeably. That is, contrary to what we call tapping, i.e., physically "bumping" the card to the reader, in the Android API guide context, tapping is used to simply mean that the card should be held very close to the reader.

HCE API on Android 4.4 (and newer versions) provides services for host-based card emulation. The `HostApduService` class is extended for implementing an HCE service with two abstract methods: `processCommandApdu` and `onDeactivated`. The former is called whenever the card receives an APDU from an NFC reader and enables half-duplex communication with the reader. The latter is called when either the NFC link is broken or the reader wishes to talk to another service. The first APDU is typically for service selection. After a successful service selection, card and reader can exchange any type of data.

For reader emulation on the other hand, Android suggests to use the `IsoDep` class. `IsoDep` provides access to ISO-DEP (ISO 14443-4) properties and I/O operations on the emulated NFC card. The most important method of this class is `transcieve` which sends APDUs to the card and receives the response. When the NFC reader discovers a tag, it sends the service selection APDU, and if successfully selected, the communication starts and continues until the NFC link is lost.

We have implemented our own card and reader applications using `HostApduService` and `IsoDep` classes, respectively, in order to simulate a complete NFC payment system based on the method we propose. These apps are publicly available via the author's homepage[10].

### 2.3.4 Sensor data preprocessing

In the main activity functions of our card and reader apps described above, we have included a *SensorManager* [5] which lets us access the device's sensors. We define a variable type of sensor and assign `Sensor.TYPE_ACCELEROMETER` to it. During tapping events, the sensor measurements (3 dimensions: x, y and z) along with their reading times are saved in an array in both card and reader apps. At the end of each experiment, these measurements are saved to text files. In addition to the complete code, the sensor measurements dataset used for the experiments of this chapter is also released via the author's homepage.

**Accelerometer data.** We use the embedded accelerometer sensor on the mobile phone to capture vibration changes during physical tapping. The accelerometer sensor returns acceleration data in three dimensions, obtained by measuring forces (including the force of gravity) applied to the sensor along the local x, y and z axes. The coordinate system is defined with reference to the phone screen in its portrait orientation; x is horizontal in the plane of the screen from left of the screen towards

[10] homepages.cs.ncl.ac.uk/m.mehrnezhad/

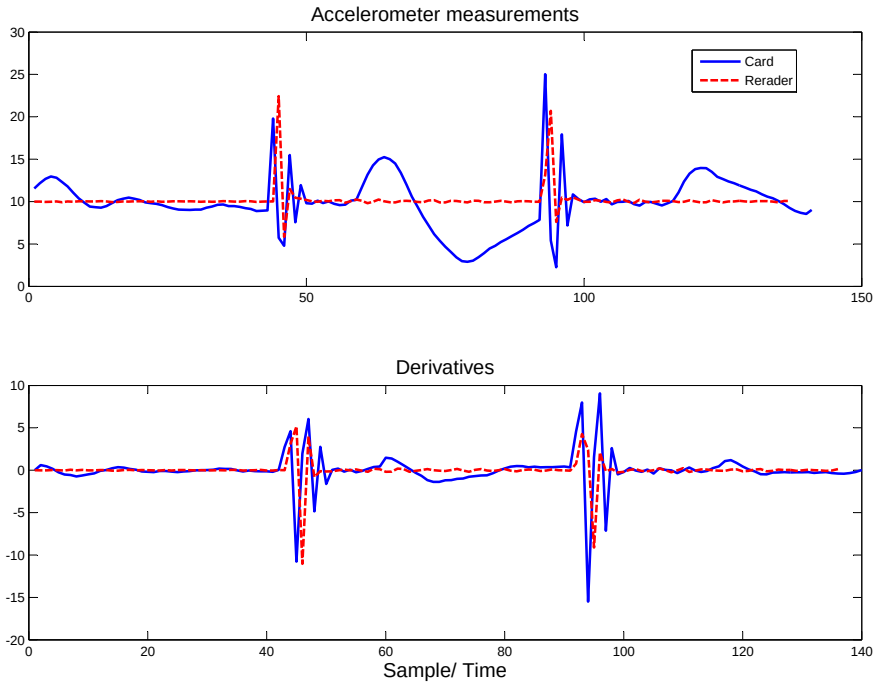


Figure 2.3: Final sequences obtained from Equation 2.1 (top), and their derivatives from Equation 2.2 (bottom) of a sample of double-tapping

right, y vertical from the bottom of the screen towards up, and z perpendicular to the plane of the screen from inside the screen towards outside. We consider the sequence representing the length of the three-dimensional vector obtained through accelerometer measurements calculated from Equation 2.1, where the components represent the $i$-th measurement in the three dimensions $(acc_{xi}, acc_{yi}, acc_{zi})$:

$$acc_i = \sqrt{acc_{xi}^2 + acc_{yi}^2 + acc_{zi}^2} \tag{2.1}$$

Fig. 2.3 (top) shows the above vector length sequences $acc_i$ for a typical double-tapping as measured on a card and a reader. From now on, we refer to the vector length sequence $acc_i$ simply as the accelerometer measurement.

**Derivatives** As shown in Fig. 2.3 (top), the accelerometer measurement made by the card has greater magnitude than that by the reader, since the card is moving in the hand of the user. They are also different in scale, depending on the start status of accelerometers. In order to smooth out irrelevant movements, especially of the card, we apply the following equation (based on [63]) to approximate the first derivatives of the sequences. The results are displayed in Fig. 2.3 (bottom).

$$D_i = \frac{(acc_i - acc_{i-1}) + ((acc_{i+1} - acc_{i-1})/2)}{2} \tag{2.2}$$

**Sequence alignment.** After obtaining the derivatives, we align the two sequences by identifying the peaks. This can be simply achieved by searching for the extreme values (max or min) with a minimum gap between them. The two sequences are then aligned based on the first peak (with a few linear shifts to get the best matching by trial-and-error). Based on our evaluation of the collected data, we found that this simple alignment algorithm is accurate and fast.

After the alignment of the two sequences, we cut a segment of each sequence, starting from 0.2 seconds before the first peak until 0.2 seconds after the second peak. This covers the whole significant variation of the accelerometer data. Our analysis shows that with this setting, the whole recording time is in the range of 0.6 and 1.5 seconds.

### 2.3.5 Similarity comparison

Suggested sensor data comparison methods include correlation coefficients, covariance, cross covariance (e.g. [19]) and cross correlation (e.g. [32] and [53]) in the time domain, and coherence (e.g. [77]) in the frequency domain. After a few informal experiments using the above methods on a small set of data, we found the correlation coefficients in the time domain and the coherence in the frequency domain to be the two most effective methods on our collected data. Moreover, while performing the experiments, we observed different users would tap the phone to the reader with different amount of pressure and speed. Here we use the correlation coefficients and coherence methods along with the energy of the series as well as the distance between the two peaks as the inputs of our suggest TTP decision maker.

**Correlation coefficient (Time domain).** The correlation coefficient is commonly used to compare the similarity of the shapes of two signals. The intuition is that if the two measurements originate from the same double-tap, their signal shapes, especially their tap shapes, would be highly correlated, and otherwise they would not be correlated. Given two sequences $X$ and $Y$ and $\mathrm{Cov}(X, Y)$ denoting covariance between X and Y, the correlation coefficient is computed as below, where $\mathrm{Cov}(X, X) = \sigma_X^2$ and $\mathrm{Cov}(Y, Y) = \sigma_Y^2$:

$$R_{XY} = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Cov}(X, X) \cdot \mathrm{Cov}(Y, Y)}} \tag{2.3}$$

**Coherence (Frequency domain).** To obtain a similarity measure in the frequency domain, we apply the coherence method which indicates the level of matching of features in the frequency domain between two time series. Given two sequences $X$

and $Y$, we compute the magnitude squared coherence based on the following equation, where $P_{XX}(f)$ and $P_{YY}(f)$ are power spectral densities of $X$ and $Y$, and $P_{XY}(f)$ the cross power spectral density between $X$ and $Y$:

$$C_{XY}(f) = \frac{|P_{XY}(f)|^2}{P_{XX}(f) \cdot P_{YY}(f)} \tag{2.4}$$

We define the similarity rate between the two signals based on magnitude squared coherence as the sum of the squares of the magnitudes of coherence values at all frequencies as follows:

$$F_{XY} = \sum_{f} C_{XY}(f) \tag{2.5}$$

**Energy Difference.** Our analysis shows that different users tap devices with different strengths; some taps are very gentle, some are of medium strength, and some are very strong. We found that the total energy levels of the card and reader sequences of the same tap are strongly correlated, while they are distinctive if obtained from different taps. Hence, we use the following measure to capture the distance of two signals $X$ and $Y$ in term of the total signal energy levels:

$$D_{XY} = \left| \sum_{t} X(t)^2 - \sum_{t} Y(t)^2 \right| \tag{2.6}$$

**Peak Gap Difference.** Last but not least, the distance between the two peaks in each measured sequence is an important factor in deciding if two measurements come from the same double-tapping or not. We define $G_{XY}$ in Equation 2.7 where $Gap_X$ is the distance between the two extremums of sequence $X$ and $Gap_Y$ is the distance defined similarly for sequence $Y$:

$$G_{XY} = |Gap_X - Gap_Y| \tag{2.7}$$

**TTP Decision Engine.** Our TTP decision engine has two steps. First, we have an initial check according to the peak gap (threshold T1) defined in Equation 2.7 and then we use a combined method to include the other three similarity measures (threshold T2). We suggest a simple linear fusion method by using the weighted sum of the three measures: correlation coefficient, coherence, and the energy similarity. Therefore, the ultimate decision is made based on comparing the peak gap against a threshold and if successful comparing the weighted sum of the combined method against another threshold. Hence according to the output of the decision engine, the bank decides to authorize or decline the transaction.

We use a simple linear normalisation that maps the three values to the interval $[0, 1]$. Let us denote these normalised versions by $\bar{R}_{XY}$, $\bar{F}_{XY}$, and $\bar{D}_{XY}$, respectively.

**Data**: $R_{XY}, F_{XY}, E_{XY}, G_{XY}, T1, T2, a, b, c$
**Result**: Are the Acc measurements from the same tapping events?

**if** $G_{XY} < T1$ **then**
  $T_{XY} = a \cdot \bar{R}_{XY} + b \cdot \bar{F}_{XY} + c \cdot \bar{E}_{XY}$;
  **if** $T_{XY} < T2$ **then**
    Return *Yes*;
  **else**
    Return *No*;
  **end**
**else**
  Return *No*;
**end**

Figure 2.4: TTP decision engine's algorithm

Since unlike the other two measures, $\bar{D}_{XY}$ decreases with similarity, we define $\bar{E}_{XY}$ as below. Note that $\bar{E}_{XY}$ is also a normalised value belonging to the interval $[0, 1]$.

$$\bar{E}_{XY} = 1 - \bar{D}_{XY} \tag{2.8}$$

Given $\bar{R}_{XY}$, $\bar{F}_{XY}$ and $\bar{E}_{XY}$, $T_{XY}$ calculates the total similarity rate of two signals $X$ and $Y$ as below, where $a$, $b$ and $c$ are the weights of each method:

$$T_{XY} = a \cdot \bar{R}_{XY} + b \cdot \bar{F}_{XY} + c \cdot \bar{E}_{XY} \tag{2.9}$$

The weight parameters are determined through experiments based on the collected user data by testing all possible weights up to two decimal places for a, b, and c — under the condition that the sum of them is equal to 1 — and observing the equal error rate. The values which gave us the best error rate have been fixed as $a = 0.45$, $b = 0.21$, and $c = 0.33$.

The procedure of our TTP decision engine is presented in Fig.2.4, where the T1 are the thresholds of our system which affect our results. Through our experiment we found the value of 7 as an optimal value for T1 and, while T2 varies depending on the desired error rates for the system as we explain in section 2.4.2.

## 2.4 System evaluation

This section contains the details of our experimental setup and data collection, as well as the system evaluation.

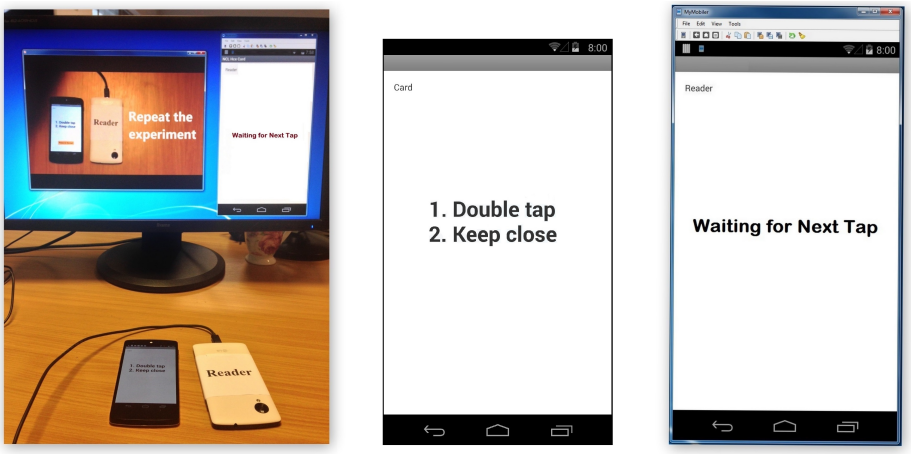


Figure 2.5: Left: Data collection environment, centre: Card app, and right: Reader app

### 2.4.1 Experiment setup and data collection

We implemented a proof-of-concept prototype for the TTP system by developing two Android apps (card and reader). When the user taps the reader, the two apps independently record the accelerometer data. Once the NFC card is detected by the reader in close proximity, the two devices start a two-way NFC communication and simulate an NFC payment.

In order to evaluate the system performance based on real user data, we recruited 23 volunteers (university students and staff, 10 males and 13 females) to participate in the data collection, each performing five double-tapping actions. We made a short self-explanatory training video, which is accessible via the author's homepage, to demonstrate how to do the double-tap and showed it to the users before the experiment. Users generally found the video guide useful in helping them quickly grasp the instruction of "Tap-Tap and Pay".

We fixed the reader phone to the table using double-sided tape, as shown in Fig. 2.5 (left). The front of the phone faced downwards and the back was labelled "Reader". We used MyMobiler[11] to operate the reader through a USB connection. The GUIs of the reader and card apps are shown in Fig. 2.5, right and centre, respectively. After launching the card app, the user just double-tapped the phone to the reader and kept it close to complete an NFC payment. Once she was notified of a successful completion, she could repeat the experiment. The recorded sensor data were saved into a file for further analysis in Matlab.

[11] www.mymobiler.com

| Method | Equal error rate |
|---|---|
| Correlation coefficients | 19.15% |
| Coherence | 27.91% |
| Total energy | 23.48% |
| Peak gap | 14.09% |
| ***TTP decision algorithm*** | ***9.99%*** |

Table 2.1: Equal error rates for different suggested methods

### 2.4.2 Results

We use the False Negative Rate (FNR) and the False Positive Rate (FPR) to evaluate the performance. These are common evaluation measurements used in different contexts e.g. biometric systems [55]. The FNR is the rate that two measurements from the same tap event are determined as not matching. The FPR is the rate that two measurements from two different tap events are determined as matching. FNR and FPR vary according to a threshold. The Equal Error Rate (EER) is the rate where the FNR and the FPR curves intersect. The EER is commonly used as a measure to evaluate the overall performance of a system. We computed the EERs based on the similarity comparison methods described in Section 2.3.5. The results for EER are presented in Table 2.1.

Overall, the Equal Error Rate of our prototype system is 9.99% using the combined method (Table 2.1). Therefore with this setting, we have FNR= FPR= 9.99%. Hence, a legitimate NFC transaction may be falsely rejected with a probability of 9.99%. Then the user would need to try again. On average, it takes $1/(1 - 0.099) = 1.1$ attempts for a legitimate user to complete an NFC payment transaction. On the other hand, if the Mafia attack takes place during the NFC payment, the transaction is more likely to be denied by the bank as a result of inconsistent data measurements. The Mafia may trick the user to try again, but it would require on average $1/0.099 = 10$ attempts to get a fraudulent transaction to come through. However, consecutively failed verifications for a single NFC transaction will likely trigger an alert at the back-end banking network, prompting an investigation. Furthermore, when the user gets repeated denials from the NFC payment (say three times), she might not try further and may choose to query her bank instead. All this can significantly increase the chance of having the Mafia attack exposed.

### 2.4.3 Online and offline modes

So far, the description of our TTP solution assumes that the NFC transaction is *online* i.e. the reader is connected to the banking network, so that the backend system is able to evaluate the received measurements and authorize the payment in real-time. The same assumption is made in other researchers' solutions [53, 73, 98] (which we will detail in Section 2.6).

However in practice, an NFC transaction may be performed *offline*. According to the EMV specifications, an EMV transaction flow includes several steps including *offline* data authentication and *online* transaction authorisation. Depending on the result of the negotiation between the card and the reader, the card may decide to proceed with offline authorisation. This decision is based on a range of possible factors including the transaction value, the type, and the card's record of recent offline transactions. Our solution will be less effective in the offline mode; however, we believe it still provides important added value in preserving critical evidence when a dispute regarding Mafia attacks occurs and a retrospective fraud investigation is needed.

## 2.5 Usability study

In this section, we provide the detail of a usability study on our proposed system.

### 2.5.1 Experiment setup and data collection

We performed a second experiment to evaluate usability aspects of the system. We asked 22 different users (partially overlapping with the previous user set, university students and staff, 15 males and 7 females) to perform two NFC payments; first using the contactless method, and second using TTP. We developed two Android apps (card and reader) to simulate the two tasks. Before the experiment, we presented our users with a study description, including a short introduction of mobile contactless payments using NFC, followed by a general description of mobile payment using TTP (see Appendix A). In the first task, the user was asked to hold the phone near the reader and wait for the confirmation message. In the second task, the user was asked to double-tap the reader, keep the phone near the reader and wait for the confirmation. Figure 2.6 shows the GUIs of the two tasks in this experiment.

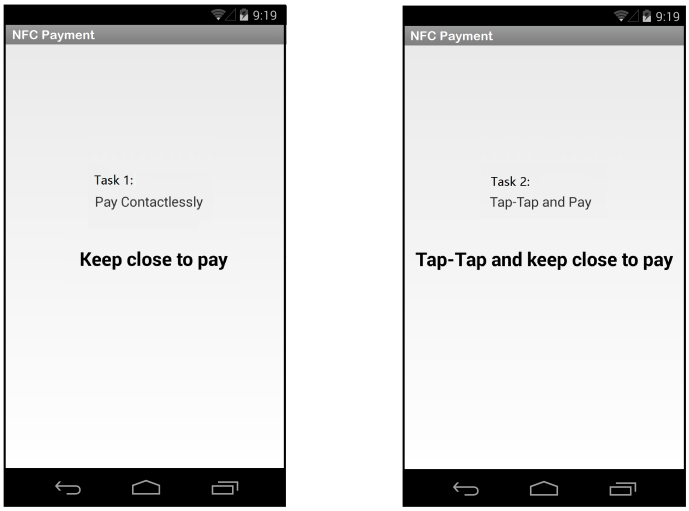


Figure 2.6: User study Card app; task 1: Contactless payment (left), task 2: TTP (right)

### 2.5.2 Findings

After completing the two tasks, the users were asked to fill in a questionnaire and rate the level of convenience, speed, and feeling of the security of each payment method in a Likert scale from level 5 to 1 (corresponding to "strongly agree", "agree", "neutral", "disagree", and "strongly disagree"). They were also asked to write free comments about their experience in this experiment. Figure 2.7 shows the average user rating of using the contactless payments and the TTP method.

As shown in Fig. 2.7, users generally found contactless payment more convenient than TTP. Including a physical action makes it less convenient for some users. As one user commented: "... the fact that I need to keep the device close to the reader after tapping made the experience less convenient".

However, in contrast to convenience, many users considered TTP faster than the contactless method, since they were able to precisely sense the start of the action by tapping, while it took them some time to find the proper distance for the contactless payment. The uncertainty about when contactless payment would start made some people feel that the process took longer than it actually took. As one user commented: "Even [though] I had to tap twice, but the process felt faster comparing to the first one. I feel after tapping I automatically bring the phone close enough to the reader, but in first task, my phone was not close for a while and it took longer".

Moreover, users felt TTP is more secure than contactless payment. By performing a physical tapping action, users felt in control of the transaction and worried less

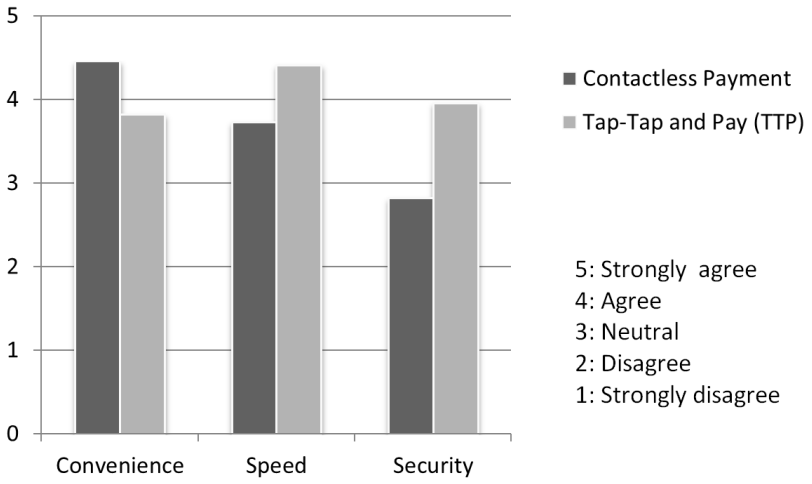


Figure 2.7: The average user rating of contactless payment and TTP

about accidental payments. As one of the users commented: "As before [i.e. task 1] payment is very easy. I like the action of tapping the reader as this made me feel more in control of when the transaction took place. I felt this method [TTP] was more secure due to the action of tapping to start the transaction. This meant I know when the transaction took place". A similar view was expressed by another user in the comment: "The payment [in task 1] is very easy, but I don't know when the connection between wallet and reader is made; range or time, so I would keep my payment device away from the reader to be sure until I want to pay."

## 2.6 Comparison with previous work

Table 2 briefly compares TTP with previous ambience sensing based solutions. In terms of security, TTP is the first solution able to prevent the Mafia attack even when both readers share the same ambient environment. Ambience sensing solutions are inherently incapable of detecting the attack in this condition.

We now review the error rates reported in the previous work based on measuring the ambient environment. Halevi et al. [53] (sensors: audio and light) report false positive and false negative rates of 0% for audio sensor, and around 5% for light sensor for distinguishing different business types (such as library, concert hall, restaurant, etc.). Ma et al. [73] (sensor: GPS) report a 0% false negative rate under the assumption that the attacker is located 20 meters or further, 67.5% when the distance is more that 5 meters, and 100% when the distance is less than one meter. False positive rates are not reported in their work. Shrestha et al. [98] (sensors: multiple sensors)

| Sensor/ Solution | Prevents attacker at same env. | Recording duration (sec) | Embedded mobile sensors | Based on ambience or device |
|---|---|---|---|---|
| Audio [53] | ✗ | 1 | ✓ | Ambience |
| Light [53] | ✗ | 2 | ✓ | Ambience |
| GPS [73] | ✗ | 10 | ✓ | Ambience |
| Temperature (T) [98] | ✗ | instant | ✗ | Ambience |
| Precision Gas (G) [98] | ✗ | instant | ✗ | Ambience |
| Humidity (H) [98] | ✗ | instant | ✗ | Ambience |
| Altitude (A) [98] | ✗ | instant | ✗ | Ambience |
| THGA [98] | ✗ | instant | ✗ | Ambience |
| **Accelerometer (TTP)** | ✓ | **0.6–1.5** | ✓ | **Device** |

Table 2.2: Comparing TTP with related work

report false negative rates approximately in the range of 10%–25% and false positive rates approximately in the range of 15%–30% for individual sensors. By combining the sensor readings, they achieve a false negative rate of about 3% and a false positive rate of about 6%.

The equal error rate of 9.99% in our result is comparable to those reported in the previous work. However, when the two readers are nearby and share the same or similar ambient environments, the reported error rates in [53] [73] [98] are no longer meaningful and all previous ambient-sensor based solutions may fail completely. By contrast, our TTP solution works regardless of whether or not the two readers share similar ambient environments.

In terms of usability, our protocol needs a sensor recording of only 0.6 to 1.5 seconds, which is sufficiently fast for contactless payment. Schemes based on audio and light sensors [53] achieve similar timings. However, the GPS-based protocol [73] requires 10 seconds of sensor recording which makes the system unsuitable for contactless payments. Our scheme is based on accelerometer sensors which are readily available in most mobile devices, as are microphones (audio), light sensors, and GPS. However, meteorological sensors [98] are only available on specialised devices which is a barrier to the adoption of such protocols in practice.

Overall, our solution presents a new approach in tackling the Mafia attack with promising initial results in terms of security, efficiency and usability. Being orthogonal ways to solve the same problem, TTP and ambient-sensor-based solutions could potentially be combined to achieve even better results. We leave this as a subject for further investigation in future.

It is worth mentioning that mobile NFC payment using mobile apps, such as Android Pay and Apple Pay, can enable the user to check the amount on the mobile screen. This would of course improve the security of the payment. However, the user is not always required to check the phone screen for an NFC payment. For example, Android Pay works even if the app is in the background. Therefore, the user only taps her phone to the reader and does not check the transaction amount on the phone screen. On the other hand, other ways of NFC payment such as bPay devices[12] — watches, fitness bands, wristbands, and car fobs — also enable the users to pay contactlessly. These devices, however, are not equipped with a screen for the user to check the amount. Augmenting such devices with an accelerometer is more cost-effective and practical (due to smaller size) in comparison to a display. As a matter of fact, fitness bands are already equipped with accelerometer sensors.

## 2.7 Further related work

In this section, we present other related work that use either a *Tap* gesture or accelerometer sensor data for other security purposes, and explain how TTP differs from them.

**Bump.** Using the tap gesture to establish device-to-device communication has been suggested before. Bump[13] is probably the most well-known example. Two users bumps their mobile phones together to exchange contacts, photos and files. Each phone sends a set of data to a remote server, including the device's location (via GPS), the IP address, the timestamp of bumping and the accelerometer measurement. The server matches the devices based on the received data and transfers the data between the two matched devices. Bump and TTP are clearly distinct as they solve different problems and they assume different threat models. Our threat model assumes a malicious reader, whereas in the Bump model, the two devices bumped to each other are assumed to be both legitimate. Consequently, our main goal is to protect against MITM adversaries whereas Bump's main goal is to identify devices being bumped together. In fact, it has been shown that Bump is vulnerable to MITM attacks [102] due to timing issues. It is worth mentioning that privacy concerns that arise from environment sensing also apply to Bump, since at least the locations and IP addresses of all users in the system are communicated with the Bump server each time the app

[12] shop.bpay.co.uk/
[13] www.bu.mp

is used. Since January 2014, Bump has been discontinued with all apps removed from App Store and Google Play [71].

**Tap identification proposals.** Performing a tap gesture in order to synchronise multiple devices has been proposed in *Synchronous Gestures* [59]. Tap identification using mobile accelerometer is another problem which could also be applied for security purposes. For example *Tap-Wave-Rub* [69] suggests a system for malware prevention for smarphones. Although similar sensors are used in these proposals, they are in general orthogonal to our solution since they are designed to solve an identification problem for legitimate devices, whereas our solution is designed to resist Mafia attacks in an environment where one of the devices behaves maliciously. Consequently, these solutions can be used alongside our proposal to provide a system in which tapping is used to both unlock the device and secure the payment.

**Shake to pair.** The idea of shaking two devices for device pairing has been suggested by many researchers [19,65,66,75–77]. While both TTP and the mentioned works use accelerometer, the amount of entropy produced by shaking, the eventual application, the threat model, and the problem solved by this work are all different from ours. In these works, the user needs to shake the two devices together for a while until both devices generate and agree on a shared key, whereas in our scheme we do not aim to generate shared keys and we only need the user to tap her device to the reader twice. Device pairing, and more generally key exchange cannot prevent Mafia attacks due to the involvement of the malicious reader. Device pairing and securing NFC payments are distinct security problems. While the former has been explored by researchers for a long time [29, 60, 67], the latter is less explored. However, with the impending global deployment of NFC payments, we believe the security of NFC payments deserves more attention by the security community.

## 2.8 Summary

In this chapter, we proposed a simple and effective solution, called "Tap-Tap and Pay" (TTP), to prevent the Mafia attack in NFC payment by sing mobile sensors. Our solution leverages the characteristics of vibration when an NFC card is physically tapped on an NFC reader. We observed that the accelerometer measurements produced by both devices were closely correlated within the same tapping, while they were different if obtained from different tapping events. The experimental results and the user feedback suggest the practical feasibility of the proposed solution. As compared with previous ambient-sensor based solutions, ours has the advantage that

it works even when the attacker's reader and the legitimate reader are in nearby locations or share similar ambient environments.

The TTP solution can be easily integrated into existing EMV standards and requires minimal infrastructural change to the EMV system. The structure of the payment protocol remains the same; only an extra string of accelerometer measurement is added in the transmitted message. In terms of hardware, deploying TTP requires the integration of accelerometer sensors in contactless readers. This can be done progressively by equipping the next generation of the readers with accelerometer sensors which are quite inexpensive (e.g. iPhone 4 accelerometers are estimated to cost 65 cents each [58]). Furthermore, TTP can be rolled out gradually since the protocols remain backward compatible.

So far, we have investigated the possibility of the use of mobile sensors for a security purpose, namely, the use of mobile accelerometers for a secure contactless payment. However, there is the other side of mobile sensors: misusing the mobile sensors for malicious purposes. In the next chapter, we demonstrate a realistic attack that uses an NFC sensor on a mobile device to threaten the privacy of the user's contactless payments.

# Chapter 3

# A Privacy Attack on Contactless Payments

## 3.1 Chapter overview

In a contactless transaction, if more than one card is presented to the payment terminal's field, the terminal does not know which card to choose to proceed with the transaction. This situation is called *card collision*. EMV (which is the primary standard for smart card payments) specifies that the reader should not proceed when it detects a card collision and that instead it should notify the payer. In comparison, the ISO/IEC 14443 standard specifies that the reader should choose one card based on comparing the UIDs of the cards detected in the field. However, our observations show that implementations of contactless readers in practice does not follow EMV's card collision algorithm, nor does it match the card collision procedure specified in ISO.

Due to this inconsistency between the implementation and the standards, we show an attack that may compromise the user privacy by collecting the user's payment details. We design and implement a malicious app, NFC Payment Spy, simulating an NFC card which the user needs to install on her phone. When she aims to pay contactlessly while placing her card close to her phone, this app engages with the terminal before the card does. The experiments show that even when the terminal detects a card collision (the app essentially acts like a card), it proceeds with the EMV protocol. We show the app can retrieve from the terminal the transaction data, which include information about the payment such as the amount and date. The experimental results show that our app can effectively spy on contactless payment transactions, winning the race condition caused by card collisions around 66% of the time when testing with different cards. By suggesting these attacks we raise awareness of privacy and security issues in the specifications, standardisation and implementations of contactless cards and readers.

The work in this chapter is mainly published as follows under the supervision of Dr. Hao. Mohammed A. Ali helped with designing some of the experiments (the attack app) of this chapter. Prof. Aad van Moorsel contributed insights on EMV contactless specifications. Some more details about the experiments are added to this chapter.

- M. Mehrnezhad, M. A. Ali, F. Hao, A. V. Moorsel, "NFC Payment Spy: Privacy attacks on contactless payments using NFC-enabled mobile", In the Proceedings of the Third International Conference of Security Standardisation Research, SSR 2016, USA, December 5-6, 2016, Springer International Publishing, Pages 1-20.

## 3.2 Introduction

NFC payment is now very popular. The statistics show that, as of February 2016, £1,318.3 million was spent in the UK in the month using a contactless card. This is an increase of 19.1% on the previous month and an increase of 306.8% over the year[1]. Apart from contactless cards, other types of technologies for contactless payment are suggested to the users. Examples include mobiles, tablets, watches, bPay bands, and bPay Stickers (bpay.co.uk). In fact, there are more than 350 different types of NFC-enabled devices on the market[2].

NFC technology is based on RFID technology. Security and privacy issues of RFID communication, and in particular NFC, have been studied intensively in the literature. Contactless cards are always open to being engaged in a transaction, and a malicious reader in the proximity of such a device is able to trigger a response from the card without the user's awareness. A number of security and privacy violations have been reported in the literature exploiting such unauthorised readings [40]. More security attacks include various types of relay attacks, such as Man-in-The-Middle and Mafia attacks [54, 79, 98, 107].

On the other hand, many researchers have shown how malicious apps compromise user's security/privacy by listening to mobile sensor data via a background process. Examples include accelerometer and gyroscope [17, 26, 81, 82, 85, 92, 115], camera and microphone [99], light [100], and Geolocation [18]. Most of these attacks work by accessing sensor data through a background process activated by a mobile app, which requires installation and user permission. Users typically install many different apps without even reviewing the app permissions. Thus, even if there is a permission request from the users, they normally ignore it [18]. This behaviour leaves the door open for attackers to obtain access to sensors. In this chapter, we also rely on such a behaviour; we develop an app using the phone's NFC functionality which the user needs to install.

**Contributions.** In this chapter, for the first time, we show that the NFC functionality on the victim's mobile phone can be used to compromise her contactless payment activities. This happens due to a particular situation in contactless payment which is called *card collision* or *card clash*. Card collision is the situation when more than a contactless card is available in the reader's field at the same time. Card

[1] theukcardsassociation.org.uk/contactless_contactless_statistics/
[2] nfcworld.com/nfc-phones-list/

Figure 3.1: Different card holder cases: flip wallet, back cover/stand, Opanable back cover, sticker cover, transparent cover

collision has been explained and addressed by EMV [15] and ISO 14443 [9][3], as the two main contactless payment references for developers to implement contactless systems. We study these standards and propose attacks based on our findings. In particular, the contributions are:

- We explain the race condition caused by card collision and study the approaches to it suggested by EMV and ISO. We perform experiments to discover the behaviour of contactless terminals when a card collision occurs. The results show that implementations on contactless terminals match neither EMV nor ISO.
- We show that, due to this inconsistency, it is possible to track the user's contactless payment activities, for instance through a malicious app. The malicious app would have a chance to intercept payment messages and data if the phone is closely located to the contactless payment card (Fig. 3.1). We propose an attack vector, called NFC Payment Spy, following EMV contactless specifications by requesting the PDOL data from the terminal when the malicious app wins the race and connects with the terminal first.
- We develop an Android app and perform experiments to support our claim. The results show that our attack can effectively break user privacy and discover the pattern of their contactless payment activities.

## 3.3 Card collision

In this section, first we present a real-world example of card collision, called *Card Clash* by Transport for London (TfL) [105]. Next, we explain the approaches suggested by EMV and ISO to handle card collision.

[3]For the rest of this chapter, unless noted otherwise, by ISO standard we mean ISO/IEC 14443, and by EMV standard, we mean EMV Contactless Specifications.

### 3.3.1 Oystercard and bank card clash

Card clash is a well-known phenomenon for a metro traveller. For example in the London metro, a traveller can either use an Oystercard or a contactless bank card[4] to pay for her journey. While swiping a wallet containing Oystercard and bank cards, the reader gets confused and does not know which card to take payment from. This causes inconvenience for users in the following ways [104, 105]:

- The commuter might inadvertently pay for her travel with a card she did not intend to use.
- The reader might refuse to work under this situation and the gate won't open.
- The passenger could be charged two maximum fares for the same journey. This happens when the reader charges one card when she touches in and another card when she touches out.
- Even if the reader opts for the contactless bank card over Oystercards for both the start and end of a journey, the passenger might end up being charged twice since she has already paid for a weekly travelcard on the Oystercard.

The only way to find out if a card clash has happened is to sign into the user online accounts and check the records of payment. If the user has been charged a maximum fare on two separate cards for the same journey, she can apply for a refund provided by TfL [105]. In fact, when TfL introduced card payments as an additional payment method to paper tickets and Oystercards in September 2014, a huge number of double payments occurred in just a few weeks. Many of those were automatically refunded within 3-5 working days. TfL has automatically handed back about £300,000 to about 50,000 customers, with refunds averaging £5.93. Although the Card Clash issue was publicised very well, surprisingly, TFL estimates that around 1,500 instances of it are occurring every day [86]. Accordingly, a range of solutions have been suggested to passengers to avoid card clash [31, 74, 86, 104] including:

- To choose the card that you want to pay with and take it out from the wallet.
- To register the Oystercard online, so that you can regularly check the online account for auditing.

[4]In the rest of this chapter unless noted otherwise, by bank card we mean contactless payment card.

- To check your bank statements regularly to find out if you have been charged on the wrong card.
- In the case of a double payment, to claim the refund by applying to the TfL website.
- To use protective cases for your contactless cards that you do not aim to pay with. Actually, Metro Bank gives free card protectors to all of its customers.
- To switch to contactless payments. TfL has fixed the problem of weekly travelcards by applying them automatically both on Oystercards and contactless bank cards. Hence, the cost would not differ that much if a passenger switches to a contactless bank card. There are reports which show that it is even cheaper if costumers move to contactless bank cards [96].
- To use a Barclaycard contactless bPay wristband (bpay.co.uk) and pay with a wave of your hand. Any UK Visa or MasterCard debit or credit card can be linked to the bPay wristband.

Among the above solutions, those which suggest replacing the Oystercard by contactless cards or bands seem more user friendly. However, not all passengers are happy with paying for a bPay and wearing it all the time. On the other hand, people normally carry multiple bank cards. Hence, even in the absence of the Oystercard, other contactless cards are still subject to card clash. Therefore, we believe that a fundamental approach is needed to overcome this real-world problem.

### 3.3.2 EMV contactless specifications

EMV is the primary protocol standard for smart card payments. The EMV standards are managed by EMVCo (emvco.com), a consortium of multinational companies such as Visa, MasterCard, and American Express. EMV has specifically defined specifications for contactless payment in books A, B, C and D [11–15]. ISO/IEC 14443 on the other hand, is an international standard that defines proximity cards used for identification, and the transmission protocols used for communication between the card and host. Generally, there are two ISO/IEC 14443 communication signal interfaces: Type A and Type B. They use different Radio Frequency Field (RF) modulation methods for the Proximity Coupling Device (PCD, Reader) to Proximity Integrated Circuit Card (PICC, Card) and the PICC to PCD communication. In this chapter, the focus

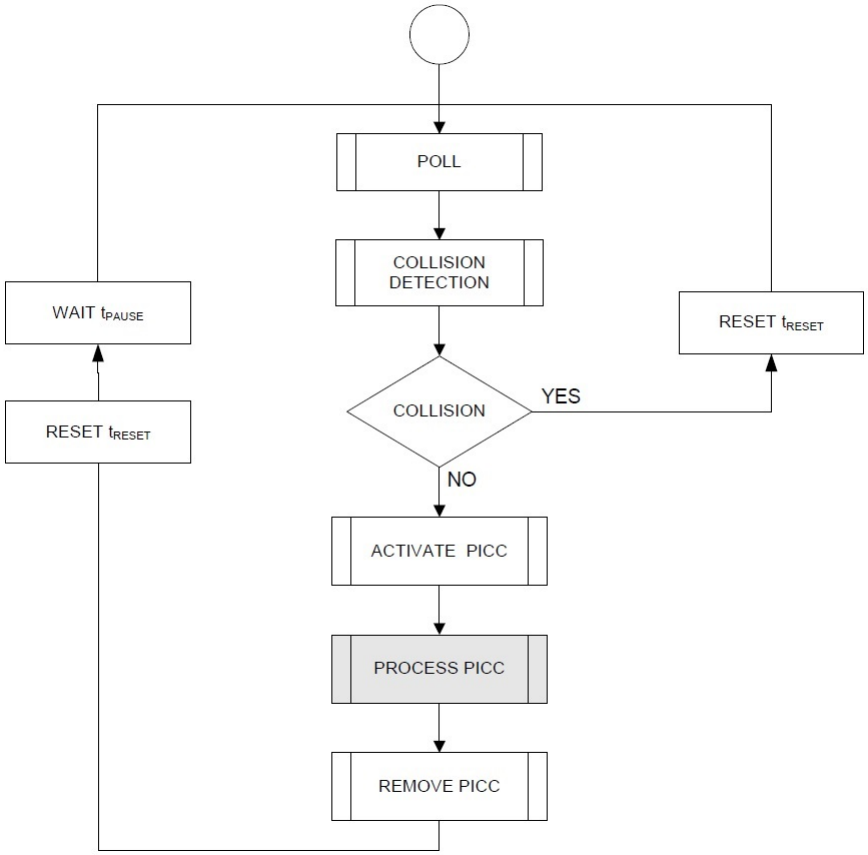


Figure 3.2: Terminal Main Loop, taken from EMV contactless Book D

is Type A, which is the mainstream technology [89]. Android supports it, and all of our tested bank cards are Type A.

EMV Contactless Book D [15] defines *Collision* as follows: "Transmission by two or more PICCs in the same PCD energizing field and during the same time period, such that the PCD [reader] is unable to distinguish from which PICC [card] the data originated". Based on this definition, the aim of the of EMV is to describe how EMV anti-collision mechanism handles the situation when there is more than a card in the field. Here we generally review the whole process for a contactless transaction from the reader's point of view.

According to EMV contactless Book D [15], the terminal is constantly running a main loop as illustrated in Fig 3.2. In the polling phase, the reader ensures that there only exists one type of technology (Type A or B) in the field by using a Wake UP command, e.g. WUPA for type A. Then it checks if there is only one card from the same technology in the field. If so, it activates the card. Remember that contactless bank cards are passive, and the reader creates an energising RF (the operating field)

that enables the card to power up. Next, the terminal application performs the transaction.

On the other hand, if there exists more than a card in the field, a collision is detected. Accordingly, the terminal will not initiate a transaction in this situation. The collision detection procedure is applicable both to cards of different technologies (Type A, B, and others), or to multiple cards with the same technology. If more than one technology is in the field, the reader must report a collision, reset the operating field, and return to the polling phase. For Type A collision detection, the terminal performs a specific procedure as follows (illustrated in Fig. 3.3). Type A cards respond to Wake UP command synchronously using Manchester coding. This allows the terminal to detect the collision in the bit level. After the terminal waits for an interval $t_p$, it sends a WUPA command. In all parts of this algorithm, if the terminal detects a transmission error in response to the WUPA and Anti-Collision (AC) commands, it reports a collision, resets, and returns to the polling procedure. Otherwise, the reader sends an AC command which is used to obtain the complete UID of a Type A card, and to detect whether more than one Type A card is in the field. Depending on the UID size of the card, the response to the AC is different. In summary, regardless of the card collision procedure, according to EMV, **once a collision is detected, the terminal should not proceed; instead it should reset the field and go back to the polling procedure**.

### 3.3.3 ISO/IEC 14443

Payment cards including contact and contactless cards are based on ISO/IEC 7816 [16] and ISO/IEC 14443 [7–10]. Mobile NFC payment technologies, such as Android Host-based Card Emulation (HCE) [4], are also based on ISO/IEC 14443, which is an international standard in four parts, defining the technology-specific requirements for proximity cards [7–10]. The third part of this standard [9], namely, Part 3: Initialization and anticollision, presents the same definition for collision as EMV. However, handling collision is different, as we next explain (presented in Fig. 3.4).

In this standard, anticollisions are detected based on a conflict in the bits of the UIDs (started from uid0 as the most significant byte). The least significant bit (LSB) of each byte is transmitted first. As an example, consider two cards as follows. Card 1: UID size = 4 bytes (single), value of uid0 =‘10’, and Card 2: UID size = 7 bytes (double). After both cards respond to the reader’s command, the terminal performs the first cascade level for the anticollision loop. As its response, the first card sends back the four UID bytes (uid0 uid1 uid2 uid3) plus some extra data. However since

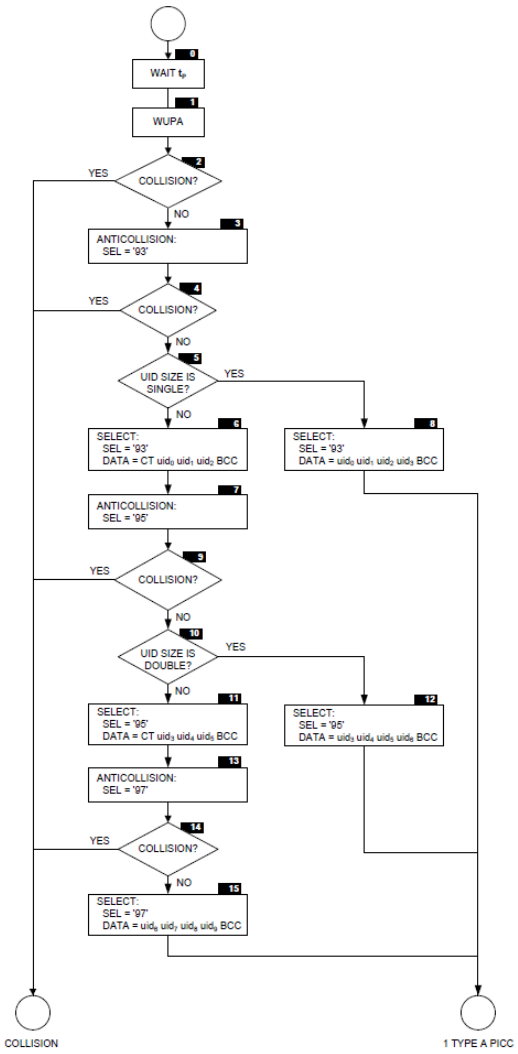


Figure 3.3: Type A collision detection, taken from EMV contactless Book D

the second card's UID is double, it sends back the cascade tag (CT) and the first three bytes ('88' uid0 uid1 uid2), plus some extra data. Hence the bits received by the terminal are: (00001000)b and (00010001)b, respectively. If the implementation pads (1)b (which is what a typical implementation does [9]) to the previous similar bits, the terminal chooses the second card over the first one and continues with it.

Therefore, unlike EMV, **ISO specifies no termination in the case of a collision. Instead, a race condition is created in which, depending on the implementation of the terminal and the UIDs of the cards available in the field, one card would be selected.** This inconsistency between EMV and ISO might cause confusion when it comes to practical implementations of these systems. We believe this is an important issue and should be addressed by the community.

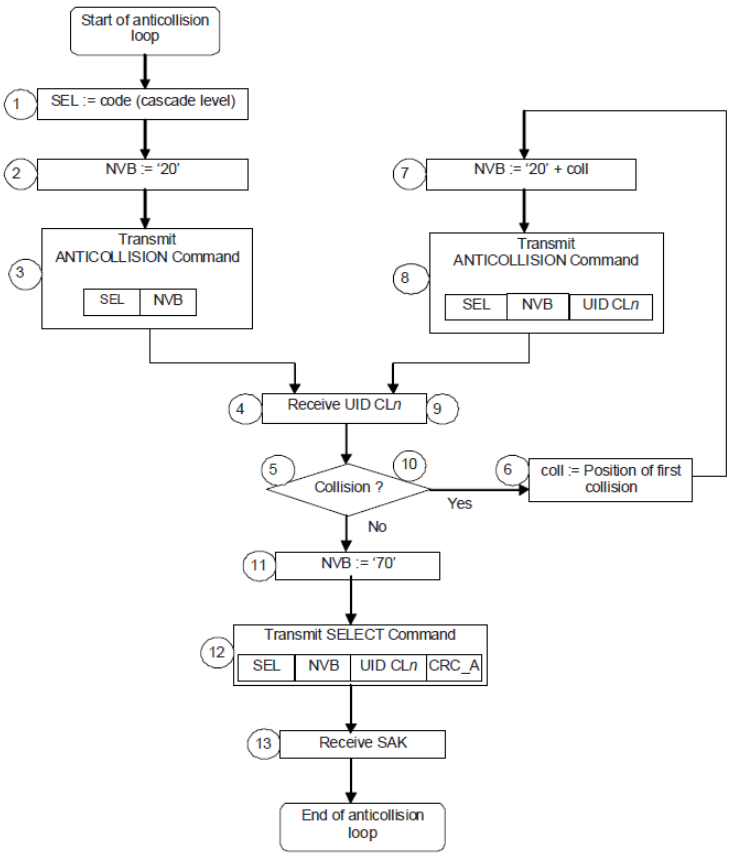


Figure 3.4: Anticollision loop, flowchart for PCD, taken from ISO/IEC 14443-3

## 3.4 Experiments on contactless readers in practice

In this section, we examine the anticollision procedure in the contactless terminals as implemented in practice. We already know that in the case of a card clash in the London metro system, the card reader may either not proceed or pick one card over another without any particular pattern [105]. It is also reported that the cards that are picked up at the start and the end of a journey may be different (in this case the passenger can apply for a refund). This suggests that the implementation in practice is not consistent with either EMV or ISO. To investigate this issue further, we performed an experiment to observe how payment terminals actually handle card collisions.

### 3.4.1 Experiment setup

In this experiment, we examined 3 pairs of contactless cards as presented in Table 3.1. Each were requested and issued from the same banks and roughly at the same

| Card | Tech. | UID size | UID0 Hex | UID0 Binary (LSB) | ISO winner |
|---|---|---|---|---|---|
| TSB- Card 1 | A | 4 | 0x35 | (10101100)b | ✓ |
| TSB- Card 2 | A | 4 | 0x65 | (10100110)b | ✗ |
| Barclays- Card 1 | A | 4 | 0xE7 | (11100111)b | ✓ |
| Barclays- Card 2 | A | 4 | 0x87 | (11100001)b | ✗ |
| barclaycard- Card 1 | A | 4 | 0x67 | (11100110)b | ✗ |
| barclaycard- Card 2 | A | 4 | 0xDF | (11111011)b | ✓ |
| Nexus 5 | A | 4 | x08 | (00010000)b | ✗ |

Table 3.1: Cards' information, LSB: Least Significant Bit, TSB: TSB visa debit, Barclays: Barclays visa debit, and barclaycard: barclaycard Platinum visa

time. These tested cards including the two TSB visa debit, and the two Barclays visa debit, were requested at the exact same time, and the two barclaycard Platinum visa, were requested and received within a month. The TSB card 1 had been in use more than card 2, and the barclaycard card 2 had been in use much more than card 1.

Before running the experiment, we tested the NFC chipsets on the cards and the phones that we used in the experiments of this chapter by writing a simple reader app using the getId() function[5]. All the tested bank cards including TSB visa debit, Barclays visa debit, and barclaycard visa have fixed 4-byte UIDs, as presented in Table 3.1. We also present the UID of two Nexus 5 mobile phones in this table. We will later use these phones for the experiments in Section 3.5. They both returned random 4-byte UIDs which always start with '08'. The first byte represents the brand of the technology [91].

### 3.4.2 Experiment results and analysis

We presented each pair of the cards listed in Table 3.1 to different contactless terminals (in multiple metro stations) several times in order to put them in race conditions. We made sure that both cards were attached to each other from the same side -contactless chipsets on each other. More specifically, when tapping the cards together to the reader, we put one of the cards on top of the other one for half of the experiments, and exchanged them for the rest of the tests. We performed the experiments of this section with the three pairs of cards in two different metro stations and at least with two different terminals (ticket machines or POS) in each station. We ended up having 27 different records. While we were doing the experiment, we carefully observed and

[5]developer.android.com/reference/android/nfc/Tag.html#getId

| No. | POS | Issuing bank | Facing card to reader | Result | Msg |
|---|---|---|---|---|---|
| 1 | MS 1, POS 1 | TSB | Card 1 | No operation | |
| 2 | MS 1, POS 1 | TSB | Card 2 | No operation | |
| 3 | MS 2, POS 1 | TSB | Card 1 | No operation | |
| 4 | MS 2, POS 1 | TSB | Card 2 | No operation | |
| 5 | MS 1, POS 2 | TSB | Card 1 | No operation | |
| 6 | MS 1, POS 2 | TSB | Card 2 | Card 1 won | msg1 |
| 7 | MS 1, POS 2 | TSB | Card 1 | Card 2 won on 2nd try | msg1 |
| 8 | MS 2, POS 2 | TSB | Card 2 | Card 1 won | |
| 9 | MS 2, POS 2 | TSB | Card 1 | No operation | |
| 10 | MS 2, POS 2 | TSB | Card 1 | No operation | |
| 11 | MS 1, POS 2 | Barclays | Card 2 | Card 1 won | |
| 12 | MS 1, POS 2 | Barclays | Card 1 | Card 2 won | |
| 13 | MS 1, POS 2 | Barclays | Card 2 | Card 1 won | msg1 |
| 14 | MS 1, POS 2 | Barclays | Card 1 | Card 2 won | |
| 15 | MS 2, POS 1 | Barclays | Card 2 | Card 1 won | |
| 16 | MS 2, POS 1 | Barclays | Card 1 | Card 2 won | msg1 |
| 17 | MS 2, POS 1 | Barclays | Card 2 | Card 1 won | msg1 |
| 18 | MS 1, POS 3 | barclaycard | Card 2 | Card 1 won | |
| 19 | MS 1, POS 3 | barclaycard | Card 1 | Card 1 won | |
| 20 | MS 1, POS 3 | barclaycard | Card 2 | Card 1 won | |
| 21 | MS 1, POS 3 | barclaycard | Card 1 | Card 1 won | |
| 22 | MS 2, POS 2 | barclaycard | Card 2 | Card 1 won | |
| 23 | MS 2, POS 2 | barclaycard | Card 1 | Card 1 won | |
| 24 | MS 1, POS 1 | barclaycard | Card 2 | Card 1 won on 2nd try | msg2 |
| 25 | MS 1, POS 1 | barclaycard | Card 1 | Card 1 won | |
| 26 | MS 2, POS 3 | barclaycard | Card 2 | Card 1 won | |
| 27 | MS 2, POS 3 | barclaycard | Card 1 | Card 1 won | |

Table 3.2: The results of putting card pairs in the race condition. MS stands for Metro Station. In the case of No operation, the cards were presented 3 times to the POS (point of sale, or ticket machine) for the same transaction. msg1: "Only present one card", msg2: "Card read failed"

manually logged the behaviour of the reader in each phase. The results are presented in Table 3.2.

As can be seen, these results do not match the anticollision algorithms suggested by either EMV or ISO. For example, regardless of the station and the terminal, most of the cases for TSB cards did not go through the payment and there was no message displayed[6] (records 1 to 5, and 9 to 10 in Table 3.2). They were two cases (records 6

[6]We only mean anti-collision messages in this context.

and 7 in the table), however, where the terminal showed this message: "only present one card", but it completed the payment. Also there was only one case (record 8) where one of the cards went through the payment without any message. On the other hand, when performing experiments with Barclays cards, the one which was not facing to the terminal won over the facing card, and this pattern was repeated (records 11 to 17 in the table). And finally between the two barclaycards, without any message shown on the terminal screen (except one case, record 24), one of the cards always won.

Generally, we can not find any specific pattern in the behaviour of these terminals when facing more than a card. Interestingly, in a few cases, the terminal shows this message: "only present one card", yet it proceeds with the payment. Based on this observation, next we demonstrate an attack which can compromise user privacy.

## 3.5 Attack design

In this section, first we present the context of the attack. Then, we explain the feasibility of our attack by designing it based on the existing contactless payment specifications.

### 3.5.1 Threat model and attack scenario

The context of this attack is when a user aims to pay for something by her contactless card where her card and phone are close to each other and both are presented to the reader's field. If the phone manages to hijack a few initial NFC signals that the card is meant to receive from the terminal, the attack is successful. In this situation, the phone is able to learn a lot about this contactless payment by requesting the PDOL data (details in section 3.5.2). The data can then be sent to a remote server controlled by the attacker. However, the malicious app would not continue further communication with the reader at some point (since it does not simulate the entire payment) and the user would realise that the payment is not being processed. In order to not disappoint the user on her second effort to pay, the NFC service on the mobile should be turned off for a few minutes once it hears from the reader. In this way, the user is able to complete the payment on the second try.

There are various ways in which the user might keep her card very close to her phone. For example, different models of card holder mobile cases are now available. These cases are capable of containing a few cards, as shown in Fig 3.1. These types of wallet are already very popular with users since they offer an easy way to travel

light and keep wallet essentials close to hand. When it comes to contactless payment, these accessories are even more popular since users do not even need to take the card out of the case. Users can slide their contactless card that is kept inside the mobile case and easily tap it against the reader for daily purchases. After the increase of the cap limit from £20 to £30 in 2015, more retailer started to accept contactless payments for small item purchases[7]. Furthermore, as shown in Fig. 3.1 (right), bPay Sticks provided by Barclays are offered to users for attachment to the back of their mobile phones. In these ways, the phone is physically close to the card.

Third parties are very interested in the sort of information that can be recovered this way, e.g. for advertising purposes. The collected information could be used in several ways. Third parties normally stimulate the users to purchase items by providing them customized ads based on this information. In addition, they can perform data mining programs to extract the patterns of individual shopping behaviours. An advanced attack might even pretend to be the user's bank by presenting this shopping information to her and tricking her to reveal her credentials via social engineering techniques. The attack we describe can be even more impactful if the malicious app turns into the reader mode and extracts the card's information, as suggested by Emms et al. [40]. Once the information is extracted, the app goes to the card mode for the rest of the attack. In this way, the attacker can easily pretend to be the user's bank by having her card information and her shopping records. We believe that these sorts of information are private to the users and should not be collected and shared without their permission.

### 3.5.2 Designing the attack based on NFC payment protocols

In this section, we cover a few key points in relation to contactless payment protocols relevant to our implementation. EMV Contactless Book B [12] covers the Entry Point Specification. This specification defines the reader requirements necessary to enable the discovery and selection of a contactless application, and activation of the appropriate kernel for processing the transaction. Different kernels are used for different Application Definition File (ADF) names (e.g. for a MasterCard ADF name, Kernel 2 is used, and for a Visa ADF name, Kernel 3 is used). Based on the chosen Kernel, different procedures will run to complete a payment. However, the entry point protocols are the same for all card schemes.

[7] theukcardsassociation.org.uk/Contactless_(our_views)/index.asp

Entry Point is designed around the use of a Proximity Payment System Environment (PPSE) as the selection mechanism. For multi-brand acceptance, this allows a reader to quickly obtain all the available brands and applications with a single command and to make an immediate choice based on priority and kernel availability. The Entry Point command and response Application Protocol Data Units (APDUs) are presented in Fig. 3.5. The File Control Information (FCI) as the response to the PPSE command from the card includes the Directory Definition File (DDF) covering a product supported by the card, the Kernel Identifier of the kernel required for the specific application underpinning the product (conditional), and the priority of the Combination (conditional). The product is indicated by its ADF name in the card. Hence, it is the card which decides what kernel to choose and talk to. Entry Point finds Combinations by matching pairs of data elements (ADF Name and Kernel Identifier) in the card with pairs of data elements in the reader (AID and Kernel ID). Once all supported Combinations have been found and the highest priority Combination has been identified, Entry Point selects the associated card application by sending a SELECT (AID) command with the ADF Name of the selected Combination. Depending on the selected AID and the kernel in the selected Combinations, a specific kernel is called to take care of the rest of the payment.

As part of the response to SELECT AID command, the card requests the Processing Options Data Object List (PDOL). Generally, a Get Processing Option (GPO) command is returned in response to this FCI command (SELECT AID) which includes the Terminal Transaction Qualifiers (TTQ), Unpredictable Number, Amount, Authorised, Transaction Currency Code, and other tags [42].

As shown in Fig. 3.5 and we explain in the next section, our attack app will take the proper action in response to each command from the terminal in order to retrieve as much as information as possible about each transaction. As mentioned earlier, when the phone and the card are closed enough to each other (see Fig. 3.1), if the phone manages to hijack a few initial NFC signals that the card is meant to receive from the terminal, the attack is successful. In this case, our malicious app follows our suggested sequence diagram as shown in Fig. 3.5. After exchanging the SELECT commands, our app requests for PDOL data which includes information about the transaction. When the terminal responds to the app with returning the requested data, our app goes to flight mode to allow the user to successfully pay with his card when he tries next time.

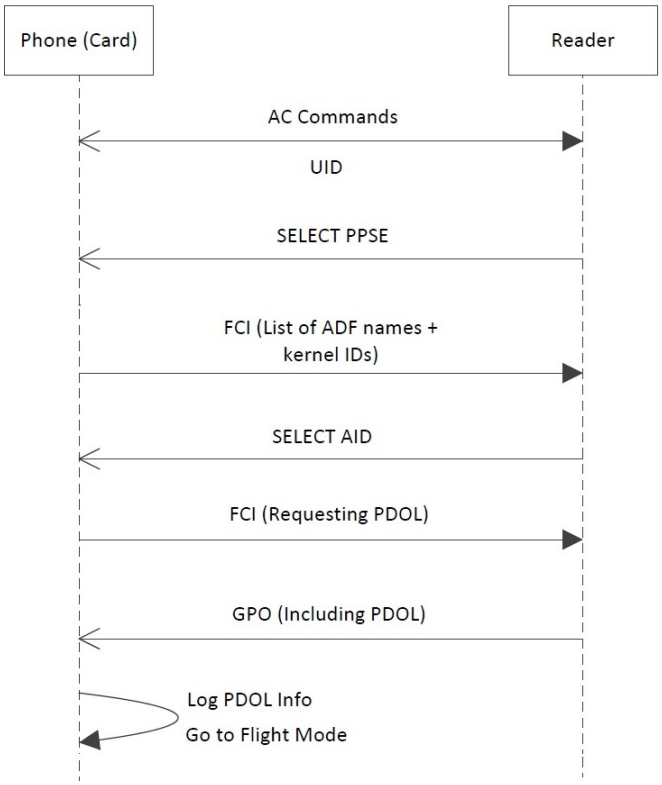


Figure 3.5: The sequence diagram of the communication between our app and the reader

## 3.6 Implementation

In this section we present the technical implementation of our attack.

### 3.6.1 Android HCE

Android supports emulating cards that are based on the NFC-Forum ISO-DEP specification (based on ISO/IEC 14443-1 to 4) and processes Application Protocol Data Units (APDUs) as defined in the ISO/IEC 7816-4 specification. In compliance with ISO/IEC 7816-4, each HCE application has an Application ID (AID). This ID enables the reader app to select the correct service.

In our implementation, we declared an AID group including an AID filter of a Visa card (0xA0000000031010) in an XML resource to be pointed by a `SERVICE_META_DATA` tag in the manifest declaration. On the other hand, Android does not interpret the PPSE selection command and, consequently, it does not generate or send a list of available payment applications. Hence we have to handle the PPSE command in the app. Typically, an HCE payment application based on EMV stan-

dards would register for both: the payment application AID and the PPSE ADF name. Note that from a protocol perspective there is no difference between an ADF name and an AID, so we can register for it in our service XML file with an AID filter for the ADF name ("2PAY.SYS.DDF01") in its ASCII hexadecimal representation of 0x325041592E5359532E4444463031. In the same file, we set the `android:requireDeviceUnlock` attribute to `false` in order to avoid the user being asked for unlocking her device.

The `HostApduService` class is extended for implementing an HCE service with two abstract methods: `processCommandApdu` and `onDeactivated`. The former is called whenever the card receives an APDU from an NFC reader and enables half-duplex communication with the reader. The latter is called when either the NFC link is broken or the reader wishes to talk to another service. According to EMV, the first two APDUs (SELECT PPSE and SELECT AID) are for service selection. That is where we request PDOL, as shown in Fig. 3.5. After a successful service selection, the card and reader can exchange any type of data. When the app receives the first GPO command including the requested data, it logs the data in a file, and the attack terminates. Accordingly, our app turns the NFC off by going to the flight mode to allow the user to complete the purchase on the second try.

### 3.6.2 Android flight mode

Android does not offer any API for turning the NFC controller on/off programmatically. Therefore, developers usually set the NFC settings in a way that prompts the user to turn it on/off manually. In our attack, once our app hears from the terminal, it needs to turn off the NFC, so that the user can successfully pay on her second try. One possible way to control the NFC adapter is to change the phone's airplane mode setting. However, only those apps with the superuser permissions can change the Airplane mode setting which requires `WRITE_SETTINGS` and `WRITE_SECURE_SETTINGS` to be declared in the manifest file. Starting from Android 4.2 , turning on/off airplane mode is no longer supported by android APIs. Hence, this part of our attack only works on a rooted device.

On the other hand, this attack needs to keep the phone's screen on since, at the moment, NFC does not work when the phone is off [4]. An advanced attack would turn the screen on only when the user wants to pay by using accelerometer and gyroscope sensor measurements in order to recognise such a gesture. Li et al. [69] show that it is effectively possible to use the tap gesture to unlock the phone for NFC applications based on accelerometer data. By augmenting such a gesture recognition

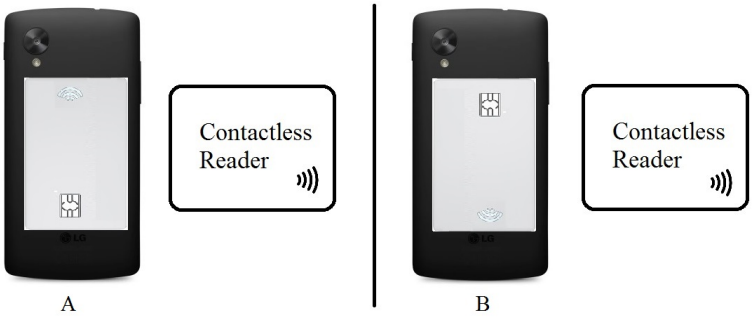


Figure 3.6: Contactless card attached to the phone in two different positions for the experiments; A (left): the NFC chipset was down, B (right): the NFC chipset was up

feature to our code, we will have a complete application that is able to compromise users privacy in contactless payments.

## 3.7 Experiments and results

We performed an experiment by installing the app on two Android phones (Nexus 5). We attached the card to the back of the phone in two different positions, as shown in Fig 3.6. The position that the card was attached to the phone was important in our experiments since it effected the results, as we explain later. In all experiments, the back of the phone was faced to the terminal (hence, the card was in a closer distance to the terminal than the phone).

### 3.7.1 Expected results

According to the EMV specifications, regardless of the UID of the card and the phone, the terminal should not proceed in the case of a card collision. ISO standards, however, suggest to select one of the UIDs (typically with higher values) in the race condition. The first UID byte (UID0) of mobile phones that we tested is always '08' (LSB: 00010000), and it is a single UID. As presented in Table 3.1, all of our cards should always win over the phone if it is a typical ISO implementation. In the following experiments, we show that the expected behaviour does not happen in practice, and the phone wins with a high probability.

### 3.7.2 Experiment A: card and phone collision

In this experiment, we tested a few different contactless cards by presenting each card with the phone to a few terminals (contactless metro ticket machines). We tested

multiple cards including two TSB visa debit, and two Barclays visa debit on different machines. During this experiment, we asked two users (colleagues from university) to pay for metro tickets with different contactless cards (from Table 3.1) that we provided to them. These cards were attached to mobile phones (Nexus 5). These participants were generally informed of the purpose of the experiment, but were not asked to follow any particular procedure. We explained to them that we want to test what is the behaviour of the terminal when both card and phone are tapped to it. We asked them to naturally pay contactlessly.

We made sure that all of our 6 tested cards were tested in different metro stations and at least with two different ticket machines. We especially made sure that both positions (A and B) were equally tested through our experiments. This lead us to end up with 44 experiments. We observed the behaviour of the terminals as summarized in Table 3.4 and 3.3. For example, in Table 3.4, the first experiment shows that our user has presented a TSB card, while it was attached to the phone in position A, to the reader. Without displaying any message on the reader, the phone managed to hear from the terminal before the card, and hence was the winner of this experiment. As another example, record number 40 in Table 3.4 is a sample of one of the cases that the card won. When a Barclayse card was attached to the phone in position B and presented to the reader by our user, on the second attempt and after displaying massage 1 ("Card read failed"), the payment went through and the card was charged.

The results show that when the card is attached to the phone in position A (the card's NFC chipset is down), the phone can hear the reader's signal first with a very high probability. On the other hand, when the card's NFC chipset is positioned to the top of the phone (position B), the chance of the card winning is slightly more than the phone. Nevertheless, an average user might put the card in any of these two positions close to the phone. Based on our experiment, generally our app is able to recognise about 66% of the user's contactless payment activities. Over time, this success rate would allow the attacker to accumulate information about the user's contactless payment patterns.

Our observations show that contactless terminals present different messages on the display based on the situation. When select to pay, it displays: "Insert, swipe or tap for GBP 0.80" as the first message. If it can not choose either the card or the phone it displays: "Card read failed", and it goes back to the first message. The fail message happened when our users tapped the card and the phone very quickly, hence none of them were presented to the field for a sufficient time. Similar to our

| No. | Card | Terminal | Position | Winner | Msg |
|---|---|---|---|---|---|
| 1 | TSB 1 | MS 1, POS 2 | A | Phone | |
| 2 | TSB 1 | MS 2, POS 2 | A | Phone | |
| 3 | TSB 1 | MS 2, POS 2 | A | Card | |
| 4 | TSB 1 | MS 2, POS 2 | A | Phone | |
| 5 | TSB 1 | MS 1, POS 1 | B | Card | |
| 6 | TSB 1 | MS 1, POS 1 | B | Card | |
| 7 | TSB 1 | MS 1, POS 1 | B | Phone | |
| 8 | TSB 1 | MS 1, POS 1 | B | Phone | |
| 9 | TSB 1 | MS 2, POS 2 | B | Card | |
| 10 | TSB 1 | MS 2, POS 2 | B | Card | |
| 11 | TSB 2 | MS 1, POS 2 | A | Phone | |
| 12 | TSB 2 | MS 1, POS 2 | A | Phone | |
| 13 | TSB 2 | MS 1, POS 2 | A | Phone | |
| 14 | TSB 2 | MS 1, POS 2 | A | Phone | |
| 15 | TSB 2 | MS 1, POS 2 | A | Phone | |
| 16 | TSB 2 | MS 3, POS 1 | A | Phone | |
| 17 | TSB 2 | MS 3, POS 2 | B | Card | |
| 18 | TSB 2 | MS 3, POS 2 | B | Phone | |
| 19 | TSB 2 | MS 3, POS 2 | B | Phone, 2nd try | msg1 |
| 20 | TSB 2 | MS 3, POS 2 | B | Card, 2nd try | msg1 |
| 21 | TSB 2 | MS 3, POS 2 | B | Phone | |

Table 3.3: Results of experiment A for TSB cards, ms1: “Card read failed”, msg2: “Only present one card”

experiments in Section 3.4.1, the terminal may show another message: “Only present one card”, but it still proceeds with the transaction.

### 3.7.3 Experiment B: PDOL data

In order to show the impact of the attack more visibly, we performed another experiment. While purchasing a ticket, we presented our final app to a payment terminal in a metro station. Our app logged the PDOL data of the transaction and then went to the Airplane mode. We built our card app in a way that it responded to the two SELECT commands – PPSE and AID – before asking for PDOL data (see Fig. 3.5).

The exchanged commands and responses APDUs are shown in Table 3.5. As it can be seen, when the card sends the second FCI, by sending PDOL tag (‘9F38’), it requests different sort of information about the transaction such as the amount (tag=‘9F02’, Amount, Authorised (Numeric)) and transaction date (tag=‘9A’). Accordingly, the terminal responds with the first GPO command including the requested

| No. | Card | Terminal | Position | Winner | Msg |
|---|---|---|---|---|---|
| 22 | Barclays 1 | MS 1, POS 1 | A | Phone | |
| 23 | Barclays 1 | MS 1, POS 1 | A | Phone | |
| 24 | Barclays 1 | MS 1, POS 1 | A | Phone, 2nd try | msg1 |
| 25 | Barclays 1 | MS 1, POS 1 | A | Phone | |
| 26 | Barclays 1 | MS 1, POS 1 | A | Phone | |
| 27 | Barclays 1 | MS 1, POS 1 | A | Phone | |
| 28 | Barclays 1 | MS 1, POS 1 | B | Card | |
| 29 | Barclays 1 | MS 1, POS 1 | B | Phone | |
| 30 | Barclays 1 | MS 1, POS 2 | B | Card, 2nd try | msg1 |
| 31 | Barclays 1 | MS 1, POS 2 | B | Phone | |
| 32 | Barclays 1 | MS 1, POS 2 | B | Card | |
| 33 | Barclays 1 | MS 1, POS 2 | B | Phone | |
| 34 | Barclays 2 | MS 1, POS 2 | A | Phone | |
| 35 | Barclays 2 | MS 1, POS 2 | A | Phone | |
| 36 | Barclays 2 | MS 1, POS 2 | A | Phone | |
| 37 | Barclays 2 | MS 1, POS 2 | A | Phone | |
| 38 | Barclays 2 | MS 1, POS 2 | A | Card | msg2 |
| 39 | Barclays 2 | MS 1, POS 2 | B | Card | msg2 |
| 40 | Barclays 2 | MS 1, POS 2 | B | Card, 2nd try | msg1 |
| 41 | Barclays 2 | MS 1, POS 2 | B | Phone | |
| 42 | Barclays 2 | MS 1, POS 1 | B | Card | |
| 43 | Barclays 2 | MS 1, POS 1 | B | Card | |
| 44 | Barclays 2 | MS 1, POS 1 | B | Phone, 2nd try | msg1 |

Table 3.4: Results of experiment A for Barclays cards, msg1: "Card read failed", msg2: "Only present one card"

items for PDOL ('83') i.e. amount ('000000000080' = 0.80 pence) and date ('160523' = 2016 May 23) [42].

As it can be seen, the attacker can easily build such a table for all transactions and discover the user's payment patterns.

## 3.8 Summary

In this chapter, we discussed a real world problem concerning the card collision when making contactless payments. We studied the EMV and ISO standards on card collision, and by performing experiments we discovered that the implementation in practice matches neither of them. Based on this inconsistency, we described and implemented an attack on the privacy of contactless payments. In this attack, we simulated a card within an app and tracked the user's contactless payment transac-

| Sender | APDU | Command |
|---|---|---|
| Terminal | 00A404000E325041592E5359532E E444446303100 | SELECT PPSE |
| Phone | 6F3C840E325041592E5359532E44 44463031A52ABF0C2761254F07A0 0000000310108701015010424152 434C41594341524420564953 41BF 6304DF2001809000 | FCI |
| Reader | 00A4040007A000000003101000 | SELECT AID |
| Phone | 6F4B8407A0000000031010A54050 10424152434C4159434152442056 495341870101**9F38**189F6604**9F02** 069F03069F1A0295055F2A02**9A**03 9C019F37045F2D02656EBF0C089F 5A0531082608269000 | FCI including PDOL request |
| Terminal | 80A8000023**83**2130000000**000000** **000080**000000000000826000000 00000826**160523**001612673900 | GPO including PDOL data |

Table 3.5: Exchanged APDUs of the PDOL experiment

tions by requesting PDOL data from the terminal. When the phone and the card were both presented to a contactless terminal, our app could successfully win the race condition over the card in the majority of test cases.

Our findings suggest vulnerabilities in the current infrastructure which needs to be addressed. More specifically, the results of our experiments show that when tapping the terminal with more than one card, in most cases (Tables 3.2, 3.4, and 3.3), the terminal does not even identify the card collision. Nevertheless, even if the terminal identifies the presence of multiple cards in the field (by showing a message), it still proceeds with the transactions. The selection of the card appears random. A countermeasure to this identified privacy attack is updating the implementation of the payment terminals according to EMV's card collision algorithm: i.e. the process should not proceed when more that one card is detected in the field. Updating some parts of EMV's protocol and protecting the PDOL data would also mitigate the introduced attack. Finally, the EMV and ISO standards would need to be updated to have a consistent algorithm to handle card collision.

Our findings in this chapter show that by using a malicious app, it is possible to impose security and privacy risks to mobile users. In the next chapter however, we show that the vulnerabilities associated with mobile sensors are not limited to native

apps which need installation by users. We will demonstrate novel side channel attacks that steal the user's private information via JavaScript code. In this way, we do not even need the user to install the malicious app on his phone; once the attack content is loaded in the mobile browser, our program starts listening to the sensor data, and by the use of classification methods, it discovers user's phone call timing, physical activities, touch actions, and even his PINs.

# Chapter 4

# Identification of User Touch Actions and PINs via JavaScript

## 4.1 Chapter overview

Mobile web browsers that conform to W3C specifications [112] allow JavaScript code in a web page to access *motion and orientation* sensor data without the user's permission. The associated risks to user security and privacy are, however, not considered in W3C specifications. In this chapter, for the first time, we show how user security can be compromised using these sensor data via browser, despite that the data rate is 3 to 5 times lower than what is available in app. We examine multiple popular browsers on Android and iOS platforms and study their policies in granting permissions to JavaScript code with respect to access to motion and orientation sensor data.

Based on our observations, we identify multiple vulnerabilities, and propose *TouchSignatures* which implements an attack where malicious JavaScript code on an attack tab listens to such sensor data measurements. Based on these streams, *TouchSignatures* is able to distinguish the user's touch actions (i.e. tap, scroll, hold, and zoom) and her entered PIN digits, allowing a remote website to learn client-side user activities. We demonstrate the practicality of this attack by collecting data from real users and reporting high success rates using our proof-of-concept implementations. Moreover, we implement a more advanced attack on full 4-digit user PINs (as opposed to digits only) by introducing PINlogger.js. Based on a test set of fifty 4-digit PINs, PINlogger.js is able to correctly identify PINs at the first attempt with a success rate of 74%, which increases to 86% and 94% on the second and third attempts, respectively. The high success rates of stealing user PINs on mobile devices via JavaScript indicate a serious threat to user security.

We also present a set of potential solutions to address the vulnerabilities. The W3C community and major mobile browser vendors including Mozilla, Google, Apple and Opera have acknowledged our work, and have implemented some of our countermeasures, as we will explain in Section 4.10.

The work in this chapter is mainly published as follows under the supervision of Dr. Hao and Dr. Shahandashti. Some of the JavaScript code for the data collections of this chapter was developed by Ehsan Toreini. He also helped on recruiting users for the data collection.

- M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "TouchSignatures: Identification of User Touch Actions based on Mobile Sensors via JavaScript", In the Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS 2015, Singapore, April 14-17, 2015, ACM, Pages 673-673.

| Work | Sensor | Identification item | Access |
|---|---|---|---|
| PIN Skimmer [99] | Camera, Mic | PINs | in-app |
| PIN Skimming [100] | Light | PINs | in-app |
| Keylogging by Mic [88] | Mic | Keyboard, PINs | in-app |
| ACCessory [92] | Acc | Keyboard, Area taps | in-app |
| Tapprints [85] | Acc, Gyr | Keyboard, Icon taps | in-app |
| Acc side channel [17] | Acc | PINs, Patterns | in-app |
| Motion side channel [27] | Acc, Gyr | Keyboard, PINs | in-app |
| TapLogger [115] | Acc, Ori | PINs | in-app |
| TouchLogger [26] | Ori | PINs | in-app |
| **TouchSignatures** | **Motion, Ori** | **Touch actions, PINs** | **in-browser** |
| **PINLogger.js** | **Motion, Ori** | **4-digit PINs** | **in-browser** |

Table 4.1: Brief description of TouchSignatures and PINLogger.js and in-app sensor-based password/ PIN identifiers. Acc: accelerometer, Gyr: gyroscope, and Ori: Orientation. Motion streams are a set of measurements which are accessible within browsers and include accelerometer, accelerometer-including-gravity, and rotation rate (see Section 4.5.2).

- M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "TouchSignatures: Identification of User Touch Actions and PINs based on Mobile Sensors via JavaScript", Journal of Information Security and Applications, Volume 26, February 2016, Pages 23-38.

- M. Mehrnezhad, E. Toreini, S. F. Shahandashti, F Hao, "Stealing PINs via Mobile Sensors: Actual Risk versus User Perception", The 1st European Workshop on Usable Security, EuroUSEC 2016, Workshop at the Privacy Enhancing Technologies Symposium (PETS 2016), July 18, 2016, Germany.

- M. Mehrnezhad, E. Toreini, S. F. Shahandashti, F Hao, "Stealing PINs via Mobile Sensors: Actual Risk versus User Perception", International Journal of Information Security, Springer, April 2017, Pages 1-23.

## 4.2 Introduction

In this section, first we present the types of accesses to mobile sensors, i.e. within app and within browser, and compare them. Then we give an overview the attacks presented in this chapter.

### 4.2.1 Mobile sensors access

Developers can have access to mobile sensors either by 1) writing native code using mobile OS APIs [52], 2) hybrid apps i.e. recompiling HTML5 code into a native app [61], or 3) using standard APIs provided by the W3C which are accessible through JavaScript code within a mobile browser[1]. The last method has the advantage of not needing any app-store approval for releasing the app or making future updates. More importantly, the JavaScript code is platform independent, i.e. once the code is developed it can be executed within any modern browser on any mobile OS.

### 4.2.2 Access to mobile sensors within app

Listening to mobile sensor data via a background process either for improving user security [20, 21, 34, 53, 69, 95, 98, 106] or attacking it [26, 84, 85, 85, 92, 115] has been always interesting for researchers. Listening to the sensor data through a malicious background process may enable the app to compromise user security. Here, we present Table 4.1 and briefly describe existing in-app sensor-based password/PIN identifiers. Some of the existing work listed in Table 4.1 try to identify PINs and passwords by using sensors such as light, camera and microphone [88, 99, 100]. In this work, we are interested in the use of accelerometer and gyroscope sensors as a side channel to learn about user PINs and passwords [17, 26, 85, 92, 115].

### 4.2.3 Access to mobile sensors within browser

All these attacks involve obtaining sensor data through a background process activated by a mobile app, which requires installation and user permission. By contrast, we suggest recording sensor measurements via JavaScript code without any user permission. This is the first report of such a JavaScript-based attack. This attack is potentially more dangerous than previous app-based attacks as it does not need any user permission for installation to run the attack code.

Mobile web applications are increasingly provided access to more mobile resources, particularly sensor data. Client-side scripting languages such as JavaScript are progressively providing richer APIs to access mobile sensor data. As can be seen in Table 1.2, currently, mobile web applications have access to many sensors such as: geolocation [110], multimedia (video cameras, microphones, webcams) [109], light [111], and device motion and orientation [112].

[1] w3.org/TR/#tr_Javascript_APIs

The W3C specifications discuss security and privacy issues for some mobile sensors, such as GPS and light. For example, the working draft on ambient light events explicitly discusses security and privacy considerations as follows [111]: "The event defined in this specification is only fired in the top-level browsing context to avoid the privacy risk of sharing the information defined in this specification with contexts unfamiliar to the user. For example, a mobile device will only fire the event on the active tab, and not on the background tabs or within iframes". The geolocation API on the other hand, requires explicit user permission to grant access to the web app due to security and privacy considerations.

On the other hand, security and privacy issues regarding motion and orientation sensor data have not been as readily evident to the W3C community and browser vendors as those of the sensors discussed above. Interestingly, in contrast to the geolocation and ambient light sensors, there is no *security and privacy considerations* section in the W3C working draft on motion and orientation sensors [112]. JavaScript code in a web page is given full access to motion and orientation sensor streams on mobile devices without needing to ask for user permission. This opens the door for attackers to compromise user security by listening to the motion and orientation sensor data, as we present here.

### 4.2.4 Access to mobile sensors within app vs. browser

The in-browser sensor data access that the W3C specification allows is heavily restricted in multiple ways. First, the access is restricted to only two types of streams: the *device orientation* which supplies the physical orientation of the device, and the *device motion* which represents the acceleration of the device[2]. Motion data includes sequences from accelerometer, accelerometer-including-gravity, and rotation rate [112]. The orientation sensor, on the other hand, derives its data by processing the raw sensor data from the accelerometer and the geomagnetic field sensor[3].

More importantly, access is also restricted to *low-rate* streams which provide data with lower frequencies as compared to those provided in-app. Here, we present two tables (Tables 4.2 and 4.3) on sampling frequencies on different platforms and popular browsers. The in-app frequency rates in Table 4.2 for Android are obtained from running an open source program (*MPLSensor.cpp* file) available in the Android Git repository[4]. And the in-app frequency rates for iOS are from `system.setAccelerometer`

[2]At the time of the writing of this thesis, W3C is developing more specification documents for sensors (w3.org/TR/generic-sensor/).

[3]developer.android.com/guide/topics/sensors/sensors_position.html#sensors-pos-orient

[4]android.googlesource.com/platform/hardware/invensense/+/android-5.0.1_r4

| **Device/mOS** | **Accelerometer** Freq. (Hz) | **Gyroscope** Freq. (Hz) |
|---|---|---|
| Nexus 5/Android 5.0.1 | 200 | 200 |
| iPhone 5/iOS 8.2 | 100 | 100 |

Table 4.2: Maximum in-app sampling rates

| **Device** | **OS** | **Browser** | **Motion Freq. (Hz)** | **Orientation Freq. (Hz)** |
|---|---|---|---|---|
| Nexus 5 | Android 5.0.1 | Chrome | 60 | 44 |
| | | Opera | 60 | 52 |
| | | Firefox | 50 | 50 |
| | | Dolphin | NA | 151 |
| | | UC Browser | NA | 15 |
| iPhone 5 | iOS 8.2 | Safari | 20 | 20 |
| | | Chrome | 20 | 20 |
| | | Dolphin | 20 | 20 |
| | | UC Browser | 20 | 20 |

Table 4.3: Maximum in-browser sampling rates

`Interval()` and `system.setGyroscopeInterval()` functions available on Coronalabs[5]. For obtaining the in-browser accelerometer and gyroscope sampling rates presented in Table 4.3, we implemented our own JavaScript code (see Section 4.3.1). We observed the amount of data recordable during a second in different mobile operating systems (mobile OS) and browsers.

As can be seen in Table 4.2, iOS and Android limit the maximum sampling rates to 100 Hz and 200 Hz, respectively. However, the hardware is capable to sample the sensor signals at much higher frequencies (up to thousands of Hz) [84]. This reduction is to save power consumption. Moreover, according to the results of our tests in Table 4.3, we discovered that all currently available versions on different mobile browsers reduce the sampling rate even further — 3 to 5 times lower, regardless of the engine (Webkit, Blink, Gecko, etc.) that they use. Our observations on the sampling rates of different mobile browsers are mostly consistent with the results reported in [84].

The tight restrictions for in-browser access on sensor-related data streams seem to be put in place as a measure to strike a balance between providing too little data to be useful, and too much data which can potentially compromise user security. Indeed, the low-rate and processed device orientation and motion data streams provided in-browser give the impression of being the minimum needed to make applications such

[5]docs.coronalabs.com/api/library/system

as game control possible in-browser, and might project a sense of security in using such in-browser access to sensor-related data in practice. However, in this work, for the first time, we show how user security can be compromised using device motion and orientation data provided in-browser as a side channel. We demonstrate how an inactive or even a minimised web page, using JavaScript, is able to listen to and silently report the device motion and orientation data about a user who is working on a separate tab or a separate app on the device. Moreover, we show that the reported data, although restricted in multiple ways as discussed before, is sufficient to recognise the user's *touch actions* such as tapping, holding, scrolling (up, down, left, and right), and zooming (in and out), and eventually the user's PINs on the separate tab/app.

Note that neither Android nor iOS explicitly require user permission to access such sensor data at the time when the browser is installed. Furthermore, none of the browsers seek user permission or even notify the user when such sensor data is provided to a JavaScript-enabled web page. Consequently, the user is completely oblivious to such an attack, that may compromise her security. At the same time, users increasingly use web browsers on their mobile devices to access services such as online banking and healthcare services which involve personal and highly sensitive information. These facts demonstrate the potential damage that may be caused by attacks such as ours and stress the urgent need for major mobile operating systems and browser developers, and also W3C standards, to address this problem.

### 4.2.5 Contributions

In this chapter, we describe the first study on the possibility of attacks compromising user security via web content, and demonstrate weaknesses in W3C standards, and also mobile OS and browser policies which leave the door open for such exploits. In particular, the main contributions of this work are as follows:

- We examine multiple popular browsers on both Android and iOS platforms and study 1) their sampling frequencies, and 2) their policies in granting permissions to JavaScript code with respect to access to orientation and motion sensor data. Based on these examinations, we identify multiple vulnerabilities which could be potentially exploited in different attack scenarios.

- Based on our findings, we propose *TouchSignatures*, which includes attacks that compromise user security through malicious JavaScript code by listening to orientation and motion sensor data streams. Our attack is designed in two phases: 1) identifying user's touch actions (e.g. tap, scroll, hold, and zoom),

and 2) identifying user PINs. We demonstrate the practicality of the above two-phase attack by collecting data from real users and reporting high success rates using our proof-of-concept implementations.

- Finally, we introduce *PINLogger.js*, an attack on full 4-digit PINs as opposed to only single digits. We show that improving the features of our neural network system by enriching its feature vector would further improve the attack results and even extend it to full 4-digit PINs.

## 4.3 Examining mobile browsers

In this section, we report our findings for a range of mobile OSs and mobile browsers with respect to policies for providing access to device motion and orientation sensor data to active web content. We developed JavaScript code (see 4.3.1) that listens to and records the above sensor data streams and carried out tests on different combinations of mobile OSs and browsers. We considered both Android and iOS, and on each mobile OS we tested a range of widely used browsers.

### 4.3.1 JavaScript code to access motion and orientation data

The JavaScript code, used in the experiments described in this chapter, sends the orientation and motion sensor data of the mobile device, if accessible through the testing browser, to our NoSQL database on mongolab.com. When the event listener fires, it establishes a socket (by using Socket.IO) between the client and the server and continuously transmits the sensor data to the database. This part of the code is presented in Fig 4.1.

```
function socketInit(){
//initial settings
socket= io.connect();
socket.on('connected', function(){
if (window.DeviceOrientationEvent){
window.addEventListener('deviceorientation', function(event){
var gamma= event.gamma;
var beta= event.beta;
var alpha= event.alpha;
socket.emit('OX', gamma);
socket.emit('OY', beta);
socket.emit('OZ', alpha); }); }
if (window.DeviceMotionEvent){
window.addEventListener('devicemotion', function(event){
var acceleration= event.acceleration;
var gacc= event.accelerationIncludingGravity;
var rotationRate= event.rotationRate;
```

```
var interval= event.interval;
var ax= acceleration.x;
var ay= acceleration.y;
var az= acceleration.z;
var ralpha= rotationRate.alpha;
var rbeta= rotationRate.beta;
var rgama= rotationRate.gamma;
var gx= gacc.x;  var gy= gacc.y;  var gz= gacc.z;
socket.emit('MX', ax);
socket.emit('MY', ay);
socket.emit('MZ', az);
socket.emit('rAlpha', ralpha);
socket.emit('rBeta', rbeta);
socket.emit('rGama', rgama);
socket.emit('MGX', gx);
socket.emit('MGY', gy);
socket.emit('MGZ', gz);
socket.emit('interval', interval); }); }
socket.on('disconnect', function(){
alert("Disconnected!"); }); }
```

Figure 4.1: A part of our js code used for sensor reading in our experiments

We provided a help document for a sample data collection process with full details as presented in Appendix B. This file has links to our data collection code and some of our datasets which are publicly available via the project page on github [6] and the author's homepage.

### 4.3.2 Popular browsers

We tested several browsers including three major browsers on Android: Chrome, Firefox, and Opera, and three major browsers on iOS: Safari, Chrome, and Opera. Other Android browsers were also included in the study due to their high download counts on the Google Play Store. The full list of tested Android browsers and their download counts can be seen in Table 4.4. There are a number of browsers with high numbers of downloads but limited capabilities, e.g. specialised search engine browsers or email-based browsers. Since these browsers do not support features such as multi-tab browsing, they are excluded from our study. The iOS App Store does not report the number of downloads, hence we used a combination of user ratings, iTunes Charts, and checking the availability of the listed Android browsers on iOS to discover and select a list of popular browsers on iOS. On both platforms, we only considered browsers that are available free of charge from the official app stores.

[6] github.com/maryammjd/Reading-sensor-data-for-fifty-4digit-PINs

| Name | Version | #Downloads |
|---|---|---|
| Chrome | 40.0.2214.89 | 500,000,000+ |
| Opera Mini Fast Browser | 7.6.40234 | 100,000,000+ |
| Opera browser for Android | 20.0.1656.87080 | 50,000,000+ |
| Firefox | 34.0.1 | 50,000,000+ |
| Dolphin | 11.3.4 | 50,000,000+ |
| UC Browser for Android | 10.1.0.527 | 50,000,000+ |
| UC Browser Mini for Android | 9.7.0.520 | 10,000,000+ |
| UC Browser HD | 3.4.3.532 | 10,000,000+ |
| Baidu Browser (fast and secure) | 4.6.0.6 | 10,000,000+ |
| CM Browser Fast & Secure | 5.1.44 | 10,000,000+ |
| Mobile Classic (Opera-based) | N/A | 10,000,000+ |
| Photon Flash Player & Browser | 4.8 | 10,000,000+ |
| Maxthon Browser Fast | 4.3.7.2000 | 5,000,000+ |
| Boat Browser for Android | 8.2.1 | 5,000,000+ |
| Next Browser for Android | 1.17 | 5,000,000+ |
| Yandex.Browser | 14.12 | 5,000,000+ |

Table 4.4: Popular Android web browsers with full capabilities. Browsers with limited capabilities that do not support multi-tab browsing are excluded. The numbers of downloads were obtained from the Google Play Store, Jan 2015.

### 4.3.3 Mobile browser access results

Table 4.5 shows the results of our tests as to whether each browser provides access to device motion and orientation sensor data in different conditions. The culumn(s) list the device, mobile OS (mOS), and browser combination under which the test has been carried out. In case of multiple versions of the same browser, as for Opera and Opera Mini, we list all of them as a family in one bundle since we found that they behave similarly in terms of granting access to the sensor data with which we are concerned. The "yes" indications under "active/same" show that all browsers provide access to the mentioned sensor data if the browser is *active* and the user is working on the *same* tab as the tab in which the code listening to the sensor data resides. This represents the situation in which there is perhaps a common understanding that the code *should* have access to the sensor data. In all other cases, as we discuss bellow, access to the sensor data provides a possible security leakage vector through which attacks can be mounted against user security. In the following we give more details on these results.

**Browser-active iframe access.** HTML *frames* are commonly used to divide a browser window into multiple segments, each of which can independently load a sepa-

| Device/OS/Browser | | Active | | | Bg | | Locked | |
|---|---|---|---|---|---|---|---|---|
| | | **same** | **iframe** | **other** | **same** | **other** | **same** | **other** |
| Nexus 5/Android 5.0.1 | Chrome | yes | *yes* | — | — | — | — | — |
| | Opera † | yes | *yes* | — | — | — | — | — |
| | Firefox | yes | *yes* | — | — | — | — | — |
| | Dolphin | yes | *yes* | — | — | — | — | — |
| | UC Browser † | yes | *yes* | *yes* | — | — | — | — |
| | Baidu | yes | *yes* | *yes* | *yes* | *yes* | *yes* | *yes* |
| | CM Browser | yes | *yes* | *yes* | *yes* | *yes* | *yes* | *yes* |
| | Photon | yes | *yes* | *yes* | *yes* | — | *yes* | *yes* |
| | Maxthon | yes | *yes* | *yes* | *yes* | *yes* | *yes* | *yes* |
| | Boat | yes | *yes* | *yes* | *yes* | *yes* | *yes* | *yes* |
| | Next | yes | *yes* | *yes* | *yes* | *yes* | *yes* | *yes* |
| | Yandex | yes | *yes* | — | *yes* | — | *yes* | — |
| iPhone 5/iOS 8.2 | Safari | yes | *yes* | — | — | — | *yes* | — |
| | Chrome | yes | *yes* | *yes* | — | — | — | — |
| | Dolphin | yes | *yes* | *yes* | — | — | — | — |
| | UC Browser | yes | *yes* | — | *yes* | — | *yes* | — |
| | Baidu Browser | yes | *yes* | *yes* | *yes* | *yes* | *yes* | *yes* |
| | Maxthon | yes | *yes* | *yes* | — | — | — | — |
| | Yandex | yes | *yes* | *yes* | — | — | — | — |
| | Mercury | yes | *yes* | *yes* | — | — | — | — |

Table 4.5: Mobile browser access to the orientation and motion sensor data on Android and iOS under different conditions. A yes indicates the browser support to access to these sensors. A † indicates a family of browsers (e.g. Opera and Opera Mini are considered to be in the same Opera family). A *yes* (in italics) indicates a possible security leakage vector. A *yes* (in italics and underlined) indicates a possible security leakage vector only in the case when the browser was active before the screen is locked.

rate web document possibly from a different web origin. We embedded our JavaScript listener into an HTML frame, namely an `iframe`, which resided within a web page at a different web address. The test was to find out whether or not the listener in a separate segment of the browser window was able to access the sensor data streams if the user was interacting (via touch actions) with the content within the same tab but on a different segment of the browser window. Figure 4.2 (left) gives an example on how an `iframe` works inside a page. The `iframe` content is loaded from a different source and is able to collect sensor data using JavaScript. Through experiments, we found that all the browsers under test provided access to the sensor data streams in this case. The findings are listed in the column under "active/iframe" in Table 4.5 indicating such an access.

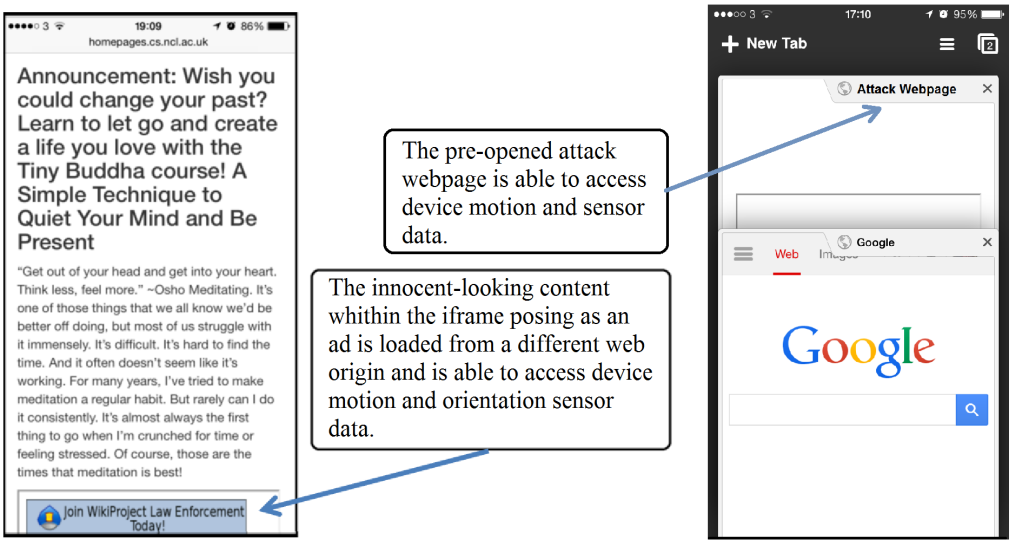


Figure 4.2: Left: An example of a page that includes an `iframe` (at the bottom of the page). Right: An example of a pre-opened attack page while the user is working on a different tab. These two examples demonstrate why *iframe* and *other tab* accesses can be threats to user security.

**Browser-active different-tab access.** In this test, we had the browser active and our JavaScript listener opened in a tab while the user was interacting with the content on a separate tab. Figure 4.2 (right) gives an example of this condition. Interestingly, we found that in addition to most of the other browsers on Android and iOS, some major browsers such as Google Chrome on iOS provided different-tab access to the sensor data streams in this case. The findings are listed in the column under "active/other" in Table 4.5 indicating browser-active different-tab access.

**Browser-in-background access.** In this test, we first opened a web page containing our JavaScript listener and then minimised the browser. While the browser was still running in the background, the user would interact with another app (via touch actions), or try to unlock the screen by providing a PIN or pattern input. We ran the test in two cases: 1) the browser had only the tab containing our JavaScript listener open, or 2) the browser had multiple tabs open including one containing our JavaScript listener. Surprisingly, we found that a few browsers on both the Android and iOS provided access to the sensor data streams when the user was interacting with another app. The findings are listed in the column under "background" in Table 4.5 indicating browser-in-background access.

**Screen-locked access.** In this test, we first opened a web page containing our JavaScript listener and then locked the screen. We found that a few browsers on both Android and iOS, including Safari, provided access to the sensor data streams even when the screen was locked. The findings are listed in the column under "locked" in Table 4.5 indicating screen-locked access.

We emphasise that none of the tested browsers (on Android or iOS) asked for any user permissions to access the sensor data when we installed them or while performing the experiments.

The above findings suggest possible attack vectors through which malicious web content may gather information about user activities and hence breach user security. In particular, *browser-active iframe access* enables active web content embedded in HTML frames, e.g. posing as an advertisement banner, to discretely record the sensor data and determine how the user is interacting with other segments of the host page. *Browser-active different-tab access* enables active web content that was opened previously and remains in an inactive tab, to eavesdrop the sensor data on how the user is interacting with the web content on other tabs. *Browser-in-background* and *screen-locked access* enable active web content that remains open in a minimised browser to eavesdrop the sensor data on how the user is interacting with other apps and on user's actions while carrying the device.

## 4.4 Identifying user activities

The potential threats to the user security posed by an unauthorised access to the described sensor data are not immediately clear. Here we demonstrate two simple scenarios which show that sensitive user information such as phone calls timing and physical activities can be deduced from device orientation and motion sensor data obtained from JavaScript.

Users tend to move their mobile devices in distinctive manners while performing certain tasks on the devices, or by simply carrying them. Examples of the former include answering a call or taking a photo, while the latter covers their transport mode. In both cases, an identifiable succession of movements is exhibited by the device. As a result, a web-based program which has access to the device orientation and motion data may reveal sensitive facts about users such as the exact timing information of the start and end of phone calls and that of taking photos. On the other hand, while the user is simply carrying her device, the device movement pattern may reveal information about the user's movement pattern, e.g. if the user is stationary

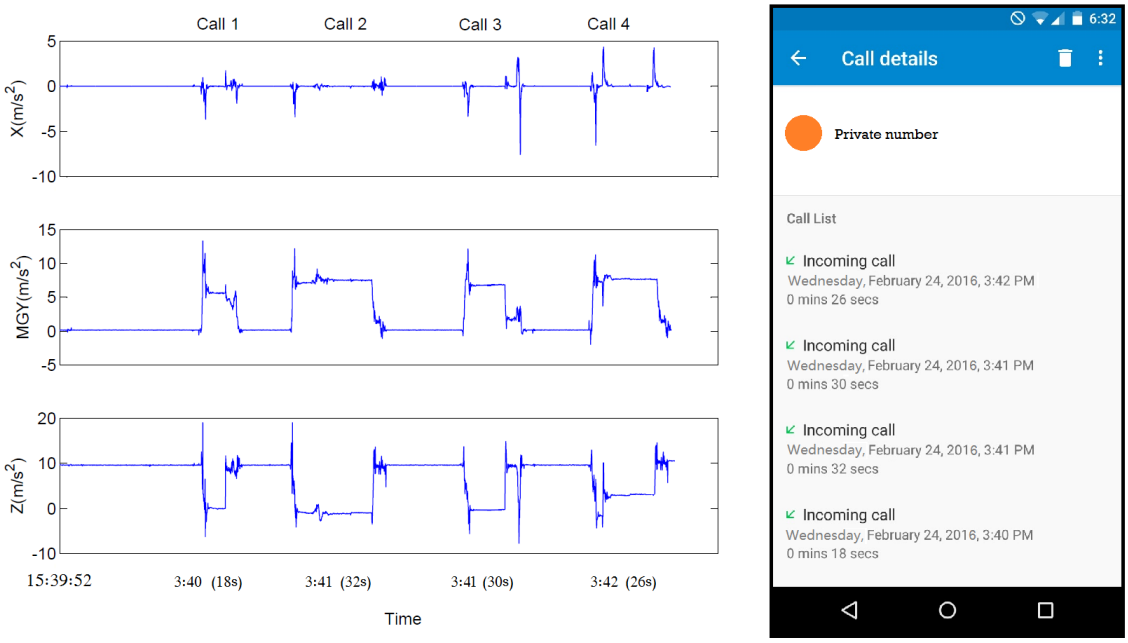


Figure 4.3: Left: Three dimensions (x, y, and z) of acceleration data including gravity (from the motion sensor). The start time, duration, and end time of four phone calls are easily recognisable from these measurements. Right: The screenshot of the call history of the phone during the experiment

in one place, walking, running, on the bus, in a car, or on the train. We present the results of two initial experiments that we performed on a Nexus 5 using *Maxthon Browser* (as an example of a browser that provides access to sensor data even when the screen is locked).

**Phone call timing.** In the first experiment, we opened the website carrying our Javascript code, then locked the screen and put the phone on a desk. The Javascript code continued to log orientation and motion data while the Android phone was on a desk. For this experiment, we used another phone to call the Android phone four times with a few seconds gap between the calls. We picked up the calls and after a few seconds (without talking) we ended the calls and returned the phone to the desk. As demonstrated in Fig. 4.3 (left), the 4 distinct phone calls along with their timing are recognisable from the three dimensions of acceleration (including gravity) which come from the device motion sensor. For a better comparison, Fig. 4.3 (right) shows the received call history of the phone during the experiment with their start times and durations. As shown in this figure, the captured sensor data match the call history.

**User physical activities.** In the second experiment, we again locked the phone, and while holding the phone in a typical pose in hand, we recorded the sensor data

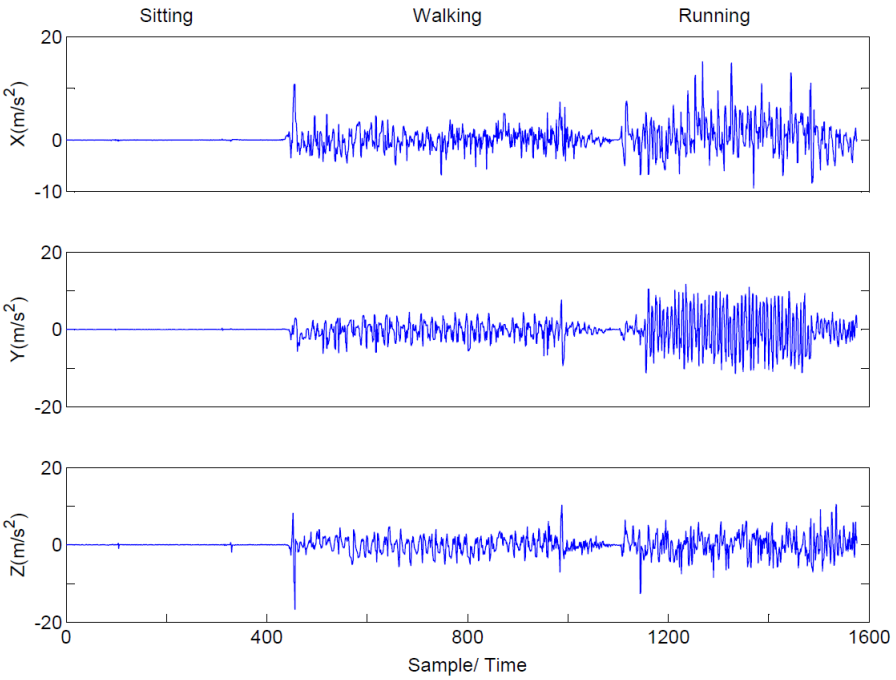


Figure 4.4: Three dimensions (x, y, and z) of acceleration data (from the motion sensor) during 22 s of sitting, 34 s of walking and 25 s of running

during 22 seconds of sitting, 34 seconds of walking and 25 seconds of slow running. We observed that these activities have visibly distinctive sensor streams. As an example, Fig. 4.4 shows the acceleration data from a motion sensor measurement. As can be seen, the mentioned activities are recognisable from each other since they are visibly different in the sensor measurements.

Our initial evaluations suggest that discovering device movement related information such as call times and user's mode of transport can be easily implemented. However, as we will explain, distinguishing user PINs is a lot harder as the induced sensor measurements are only subtly different. In the following sections we will demonstrate that, with advanced machine learning techniques, we are able to remotely infer the entered PINs on a mobile with high accuracy.

## 4.5 TouchSignatures: Identifying touch actions and PIN digits

To show the feasibility of our security attack, in the following sections, we will demonstrate that, with advanced machine learning techniques, we are able to distinguish the user's touch actions and PINs with high accuracy when the user is working with

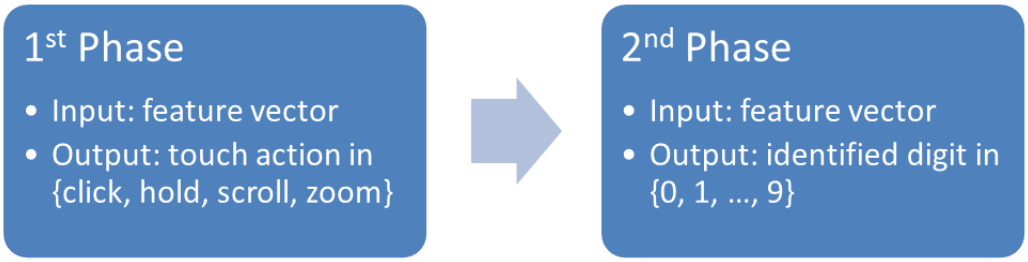


Figure 4.5: TouchSignatures overview

a mobile phone.

### 4.5.1 Overview

Each user touch action, such as clicking, scrolling, and holding, and even tapping characters on the mobile soft keyboard, induces device orientation and motion traces that are potentially distinguishable from those of other touch actions. The identification of such touch actions may reveal a range of activities about user interaction with other webpages or apps, and ultimately PINs. A user's touch actions may reveal what type of web service the user is using as the patterns of user interaction are different for different web services: e.g. users tend to mostly scroll on a news web site, while they tend to mostly type on an email client. On known web pages, a user's touch actions might reveal which part of the page the user is more interested in. Combined with identifying the position of the click on a page, which is possible by using the differing signatures produced by clicking different parts of the screen, the user's input characters could become identifiable. This in turn reveals what the user is typing on a page by leveraging the redundancy in human languages, or it may dramatically decrease the size of the search space to identify user passwords.

We introduce *TouchSignatures*, which distinguishes user touch actions and learns PIN digits in two phases. Figure 4.5 shows a top level view of the two phases of *TouchSignatures*. The input to the *TouchSignatures* system is a feature vector, which we will explain later, and the output is the type of the touch action (click, hold, scroll, and zoom) in phase one and the PIN digits (0 to 9) in phase two. This is the first attack in the literature that compromises user security through JavaScript access to sensor data.

Bearing in mind that this is only the first investigation of JavaScript access to sensor data on mobile devices, we limit the scope of our investigation in the following ways. First, we only identify digital PINs rather than alphanumeric passwords. We

expect it to be possible to extend our work to recognize the full alphanumeric soft keyboard, but the classification techniques would probably be different. Second, in the proof-of-concept implementation of the attack, we focus on working with active web pages, which allows us to easily identify the start of a touch action through the JavaScript access to the *onkeydown* event. A similar approach is adopted in other works (e.g. TouchLogger [26] and TapLogger [115]). In a general attack scenario, a more complex segmentation process is needed to identify the start and end of a touch action. This may be achieved by measuring the peak amplitudes of a signal, as done in [88]. However, the segmentation process will be more complex, and we leave that to future work.

### 4.5.2 In-browser sensor data detail

The attack model we consider is malicious web content spying on a user via JavaScript. The web content is opened as a web page or embedded as an HTML frame in a segment of a web page. The user may be interacting with the browser or any other app given that the browser is still running in the background. We assume that the user has access to the Internet as reasonably implied by the user launching the browser app. *TouchSignatures*'s client-side malicious web content collects and reports sensor data to a server which stores and processes the data to identify the user's touch actions.

The sensor data measurements available as per the W3C specifications [112], i.e. device motion and orientation, as follows:

- device *orientation* which provides the physical orientation of the device, expressed as three rotation angles: alpha, beta, and gamma, in the device's local coordinate frame,
- device *acceleration* which provides the physical acceleration of the device, expressed in Cartesian coordinates: x, y, and z, in the device's local coordinate frame,
- device *acceleration-including-gravity* which is similar to acceleration except that it includes gravity as well,
- device *rotation rate* which provides the rotation rate of the device about the local coordinate frame, expressed as three rotation angles: alpha, beta, and gamma, and

- *interval* which provides the constant rate with which motion-related sensor readings are provided, expressed in milliseconds.

The device's local coordinate frame is defined with reference to the screen in its portrait orientation: x is horizontal in the plane of the screen from left of the screen towards right; y is vertical in the plane of the screen from the bottom of the screen towards up; and z is perpendicular to the plane of the screen from inside the screen towards outside. Alpha indicates device rotation around the z axis, beta around the x axis, and gamma around the y axis, all in degrees.

To design *TouchSignatures*, we employ the supervised learning approach, i.e. train a machine learning system based on labelled data collected from the field. Consistent with the attack model discussed above, we developed a suite of applications including a client-side JavaScript program in a web page that records the sensor data and a server-side database management system (DBMS) that captures and stores user sensor data in real-time. Subsequently, we recruited different groups of users and collected sensor data samples for different touch actions and PINs, using our client-side web page that we developed for data collection purposes, while in real-time the captured sensor data was reported to and stored at our server-side database. Eventually, we extracted a set of descriptive features from the sensor data and trained a machine learning system for *TouchSignatures* which includes multiple classifiers. In the following, we give the details of our application implementation, experiments, feature extraction, and training algorithms.

### 4.5.3 Application implementation

**Client side.** On the client side, we developed a *listener*, which records sensor data streams, and a web page *interface*, which is used to collect labelled data from the subjects in our experiment. The implementation is in JavaScript. The listener mainly includes the following components: an event listener which is fired on page load and establishes a socket connection between the client and server using `Socket.IO`[7], an open source JavaScript library supporting real-time bidirectional communication which runs in browser on the client side; and two event listeners on the window object, fired on device motion and device orientation Document Object Model (DOM) events (called `devicemotion` and `deviceorientation`), which send the raw sensor data streams to the server through the established socket. The sensor data streams

[7]socket.io

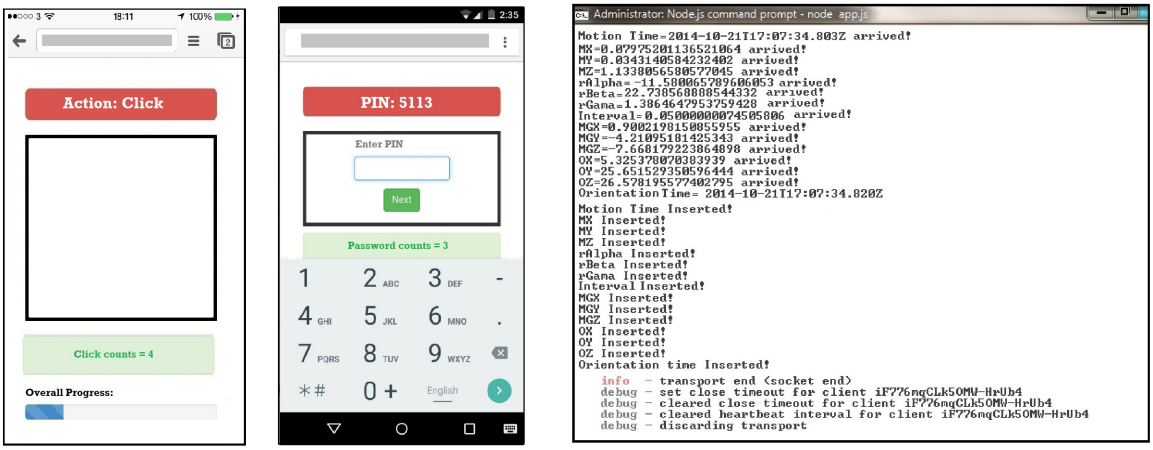


Figure 4.6: The client side GUIs presented to the user during data collections (left for Touch actions, and centre for PINs), and the data received at the server side (right)

are sent continuously until the socket is disconnected, e.g. when the tab that loads the listener is closed. The code is presented in 4.3.1.

The (user) interface sits on top of the listener and is used for data collection. We developed an HTML 5 compliant page including JavaScript using `bootstrap`[8] (a popular framework for web app creation). Data collection occurs in two rounds (for touch actions and PINs) and multiple steps. In each step the user is instructed to perform a single touch action or enter a 4-digit PIN. Sensor data from the touch actions and PINs are collected from the user successively. The label describing the type of the task or the digits in the PIN and timing information for the tasks is reported to the server. The GUI includes a concise instruction to the user as to what the user needs to do at each step. Snapshots of the GUIs presented to the users in the two different phases are illustrated in Figure 4.6 (left, and centre). More details can be found in Sections 4.6.2 and 4.7.2.

**Server side.** On the server side, we developed a *server* to host the data and handle communications, and a *database* to handle the storage of the captured sensor data continuously. The server is implemented using `Node.js`[9], which is capable of supporting data intensive applications in real-time. The `Socket.IO` JavaScript library sits on `Node.js` and handles the communications with the client; see Figure 4.6 (right). For the DBMS we have opted to implement a NoSQL database on MongoLab[10]. NoSQL

[8] getbootstrap.com
[9] nodejs.org
[10] mongolab.com

databases are document-oriented, rather than relational. They are known for being capable handling high-speed streams of data in real-time. MongoLab is a cloud-based database-as-a-service NoSQL database management system (DBMS).

### 4.5.4 Feature extraction

In this section, we discuss the features we extract to construct the feature vector which subsequently will be used as the input to the classifier. Various type of features are proposed by researchers in the literature including time domain and frequency domain features (e.g. see [17,84,85,92,95]). After a few set of informal experiments and testing some of the features proposed by previous works, we decided to consider both time domain and frequency domain features as we explain here. The captured data include 12 sequences: acceleration, acceleration-including-gravity, orientation, and rotation rate, with three sequences for each sensor measurement. Before extracting features, to cancel out the effect of the initial position and orientation of the device, we subtract the initial value in each sequence from subsequent values in the sequence.

**Time domain features.** In the time domain, we consider both the raw captured sequences and the (first order) derivative of each sequence. The rationale is that each sequence and its derivative include complementary information on the touch action. To calculate the derivative, since we have low frequency sequences, we employ the basic method of subtracting each value from the value appearing immediately afterwards in the sequence. That is, if the sequence values are represented by $v_i$, the derivative sequence is defined as $d_i = v_i - v_{i-1}$.

For the device acceleration sequences, we furthermore consider the Euclidean distance between consecutive readings as a representation of the change in device acceleration. This is simply calculated as the following sequence:

$$c_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2}$$

This gives us a sequence which we call the device acceleration change sequence, or DAC sequence for short.

First we consider basic statistical features for all sequences, their derivative, and the DAC sequence. These features include *maximum*, *minimum*, and *mean (average)* of each sequence and its derivative, plus those of the DAC sequence. We also consider the total energy of each sequence and its derivative, plus that of the DAC sequence, calculated as the sum of the squared sequence values, i.e. $E = \sum v_i^2$. Here, in total

we get 102 features for each sensor reading in the time domain. Later we add further features to the input of the first phase (touch actions) in Section 4.6.3.

**Frequency domain features.** To distinguish between sequences with different frequency contents, we applied the Fast Fourier transform (FFT) [25] of the sequences. We calculated the maximum, minimum, mean, and energy of the FFT of each sequence and consider them as our frequency domain features, i.e. a total of 48 frequency domain features.

### 4.5.5 Classification method

To decide which classification method to apply to our data, we implemented various classification algorithms to assess their efficiency. Our test classifiers included discriminant analysis, naive Bayes, classification tree, kNN, and ANN [36]. Different classifiers work better in the different phases of *TouchSignatures* (touch actions and PINs). The chosen classifiers in each phase are presented in Sections 4.6.3 and 4.7.3. In both phases, we consider a generic approach and train our algorithms with the data collected from multiple users. Hence, our results are not user-dependent.

## 4.6 Phase 1: Identifying touch actions

In this section we present the first phase of *TouchSignatures* that is able to distinguish user touch actions given access to the device orientation and motions sensor data provided by a mobile browser.

### 4.6.1 Touch actions set

We consider a set of 8 commonly used touch actions through which users interact with mobile devices. These actions include: *click, scroll (up, down, right, left), zoom (in, out), and hold*. They are presented in Table 4.6 along with their corresponding descriptions. Our experiments show that by applying machine learning techniques these actions are recognisable from their associated sensor measurements.

### 4.6.2 Experiments

We collected touch action samples from 11 users (university staff and students) using Google Chrome on an iPhone 5. We presented each user with a brief description of the project as well as the instruction to perform each of the 8 touch actions (details

| Touch Action | Description |
|---|---|
| Click | Touching an item momentarily with one finger |
| Scroll<br>– up, down, right, left | Touching continuously and simultaneously sliding in the corresponding direction |
| Zoom<br>– in, out | Placing 2 fingers on the screen and sliding them apart or toward each other, respectively |
| Hold | Touching continuously for a while with one finger |

Table 4.6: The description of different touch actions users perform on the touch screen of a mobile device.

in Appendix C). The users were provided with the opportunity of trials before the experiment to get comfortable using the web browser on the mobile phone. They also could ask any question before and during the experiments. We asked the user to remain sitting on a chair in an office environment while performing the tasks. The provided GUI instructed the user to perform a single touch action in each step, collecting 5 samples for each touch action in successive steps with a three-second wait between steps. During the experiment, the user was notified of her progress in completing the expected tasks by the count of touch actions in an overall progress bar, as shown in Figure 4.6 (left).

Data were collected from each user in two settings: *one-hand mode* and *two-hand mode*. In the *one-hand mode*, we asked the users to hold the phone in one hand, and use the same hand's thumb for touching the screen. In the *two-hand mode*, we asked them to use both hands to perform the touch actions. With these two settings, we made sure that our collected data set is a combination of different modes of phone usage. Note that zoom in/out actions can only be performed in the two-hand mode. Still, we distinguish two postures: 1) when a user holds the phone using one hand and performs zoom in/out actions by using the thumb of that hand and any finger of the other hand, and 2) when a user holds the phone using both hands and performs zoom in/out by using the thumbs of both hands. We collected data for both postures.

We had 10 samples of each of the following actions: click, hold, scroll down, scroll up, scroll right and scroll down. Five samples were collected in the one-hand mode and 5 in the two-hand mode. In addition, we collected 10 samples for each of the following two actions: zoom in and zoom out. All 10 samples were collected in the two-hand mode, with half for each of the two postures. Each user's output was a set of 80 samples. With 11 users, we ended up with 880 samples for our set of touch actions. The experiment took each user on average about 45 minutes to complete. Each user received a £10 Amazon voucher for their contribution to the work.

### 4.6.3 Classification algorithm

Before discussing the algorithms used in this phase, we add another 14 features to the *TouchSignatures*' time domain features. To differentiate between touch actions with a longer "footprint" and those with a shorter footprint, we consider a feature which represents the length (i.e. number of readings) of each dimension of the acceleration and acceleration-including-gravity sequences that contain 70% of the total energy of the sequence. The reason behind this choice is that, based on our observations, while the length of some the touch actions were similar, the length of the energised parts were different. To calculate this length, we first find the "centre of energy" of the sequence as follows: $CoE = \sum(i\, v_i^2)/E$, where $E$ is the total energy as calculated before. We then consider intervals centred at $CoE$ and find the shortest interval containing 70% of the total energy of the sequence. Therefore, considering both time domain and frequency domain features from Section 4.5.4 in addition to the new ones, *TouchSignatures*' final vector for phase one has 164 features in total.

Our evaluations show that the $k$-nearest neighbour ($k$-NN) algorithm [30] gives the best overall identification rate for our data. $k$-NN is a type of lazy learning in which each object is assigned to the class to which the majority of its $k$ nearest neighbours are assigned, i.e. each feature vector is assigned to the label of the majority of the $k$ nearest training feature vectors. A distance function is used to decide the nearest neighbours. The most common distance function is the *Euclidean distance*, but there are other distance functions such as the *city block distance* (a.k.a. Manhattan or taxicab distance). For two given feature vectors $(f_1, f_2, \ldots, f_n)$ and $(f'_1, f'_2, \ldots, f'_n)$, the Euclidean distance is defined as $\sqrt{\sum(f_i - f'_i)^2}$ and the city block distance as $\sum|f_i - f'_i|$.

Based on the results of our evaluations, we decide to use two classifiers in two stages. In the first stage, the data is fed to the first classifier which is a *1-NN classifier using Euclidean distance*. This classifier is responsible for classification of the input data into 5 categories: click, hold, zoom in, zoom out, and scroll. In the second stage, if the output of the first stage is scroll, then the data is fed into the second classifier which is a *1-NN classifier using city block distance*. This classifier is responsible for classification of a scroll into one of the 4 categories: scroll up, scroll down, scroll right, and scroll left. We used a 10-fold cross validation approach for all the experiments.

| Touch action | Click | Hold | Scroll | Zoom in | Zoom out |
|---|---|---|---|---|---|
| Click | **78.18%** | 5.45% | 2.73% | 0% | 0% |
| Hold | 10.90% | **88.18%** | 0.68% | 1.81% | 1.82% |
| Scroll | 7.27% | 2.72% | **95.91%** | 0.90% | 0.90% |
| Zoom in | 0% | 1.82% | 0.23% | **71.82%** | 20.90% |
| Zoom out | 3.64% | 1.82% | 0.45% | 25.45% | **76.36%** |
| Total | 100% | 100% | 100% | 100% | 100% |

Table 4.7: Confusion matrix for the first classifier for different touch actions

### 4.6.4 Results

In this section we show the results from the cross validation of the collected user data by presenting the identification rates and confusion matrices for both classifiers. As mentioned before, in our experiments, we input the data collected from all users to our classifiers. Therefore, these results are the outputs of our classifiers for multiple users. In a 10-fold cross validation approach, the classifier randomly segments the input data into 10 parts, trains the system with 9 parts of it, and tests the system with the remaining part. This process is repeated 10 time for each segment separately, and the results is the average of these 10 tests. Considering all scrolls (up, down, right, and left) in one category, the overall identification rate is 87.39%.

Table 4.7 shows the confusion matrix for our first classifier. In each cell, the matrix lists the probability that the classifier correctly labels or mislabels a sample in a category. The actual and classified categories are listed in the columns and rows of the table, respectively. As shown in Table 4.7, the worst results are for the pairs of Click and Hold (10.9% and 5.45%), and also pairs of Zoom in and Zoom out (25.45% and 20.9%). This is expected since click and hold are very similar actions: and hold is basically equivalent to a long click. Zoom in and zoom out also require the user to perform similar gestures. Another significant value is the classifier's confusion between click and scroll (7.27%, 2.73%), which again is not surprising since scroll involves a gesture similar to a click. Apart from the mentioned cases, the rest of the confusion probabilities are nearly negligible.

Table 4.8 shows the identification rates and confusion matrix for our second classifier, respectively. Overall, our second classifier is able to correctly identify the scroll type with a success rate of 61.59%. The classifier mostly mislabels the pairs (down, up), and (right, left), which is somehow expected since they involve similar gestures.

The obtained results show that attacks on user privacy and security by eavesdropping sensor data through web content are feasible and are able to achieve accurate

| Touch action | Scroll down | Scroll up | Scroll right | Scroll left |
|---|---|---|---|---|
| Scroll down | **57.27%** | 19.09% | 12.73% | 4.55% |
| Scroll up | 26.36% | **69.09%** | 16.36% | 6.36% |
| Scroll right | 9.09% | 4.55% | **48.18%** | 17.27% |
| Scroll left | 7.27% | 7.27% | 22.73% | **71.82%** |
| Total | 100% | 100% | 100% | 100% |

Table 4.8: Confusion matrix for the second classifier for different scroll types

results. Further security risks could be imposed to the users if the attack tries to identify what character has been pressed on the touch screen. In phase 2 of *TouchSignatures*, we show that it is indeed possible to succeed such an attack by identifying the digits entered for the user's PINs.

## 4.7 Phase 2: Identifying PIN digits

In this section, we present the second phase of *TouchSignatures* which is able to identify user PINs based on the motion and orientation sensor data provided by JavaScript code. As mentioned in Section 4.2, classifying characters entered on a touch screen virtual keyboard has already been explored by other researchers based on the sensor data accessible through native apps. In this chapter, for the first time, we show that it is also possible to achieve the same result using the sensor data obtained via JavaScript, despite the fact that the available frequency is much lower (see Table 4.3).

In this phase, we present the results of our attack on both Android (Nexus 5) and iOS (iPhone 5) devices and we describe the training of two different classifiers (neural networks) for them. Note that JavaScript is able to obtain specific information about a mobile device. For example the browser platform and the screen size are accessible via Navigator DOM[11] and Screen DOM[12] objects, respectively. Even more, by using Navigator userAgent Property[13], the brand of the device (along with some other information) is accessible. The obtained values for the tested devices are summarized in Table 4.9. Hence, though the experiments are performed using specific mobile devices, the results have general implications on all devices.

[11] w3schools.com/js/js_window_navigator.asp
[12] w3schools.com/js/js_window_screen.asp
[13] https://www.w3schools.com/jsref/prop_nav_useragent.asp

| Attribute | iPhone 5 | Nexus 5 |
|---|---|---|
| navigator.platform | iPhone | Linux armv7l |
| screen.width | 320 pixs | 360 pixs |
| screen.height | 568 pixs | 640 pixs |
| navigator.userAgent | iPhone 5 | Nexus 5 |

Table 4.9: The device information accessible via JavaScript

### 4.7.1 Digit set

We consider a numerical keypad and leave the attack on the full keyboard as future work. A numerical keyboard includes a set of 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9, and a few more characters such as -, ., and #, depending on the mobile OS. For example Figure 4.6 (centre) shows a numerical keypad on an Android device. The idea is to identify the pressed digits in a PIN. Hence from a top view, once the first phase of *TouchSignatures* distinguishes that the user is "clicking" digits on a soft keyboard, the second phase is started in order to obtain the entered digits.

### 4.7.2 Experiments

Similar to the first experiment, we asked a group of users (university student and staff) including 12 users to participate in our experiment in two parts. The first part was on an iPhone 5 and the second part was on a Nexus 5, both using Chrome. After giving a brief description about the study to the users, they were presented with a simple GUI (Figure 4.6, centre) asking them to enter 25 4-digit PINs on both devices. The 4-digit PINs were designed in a way that each number was exactly repeated 10 times in total. After entering each 4-digit PIN, the user could press a *next* button to go to the next PIN. They also could keep track of their progress as the number of PINs they have entered so far was shown on the page.

In this experiment, we asked the users to remain sitting on a chair and hold the phone in the way that they felt comfortable. The collected data contained a mixture of *one-hand mode* and *two-hand mode* records. In the one-hand mode, the user pressed the digits with one of the fingers of the same hand with which they were holding the phone. In the two-hand mode, they pressed the digits with either the free hand, or both hands. We had 10 samples of each digit for each user. Since we had 10 digits, each user's output was a set of 100 samples for each device. With 12 users, the input of our classifiers was 1200 records for iPhone 5 and 1200 records for Nexus 5. It took each user 2 minutes on average to complete each part of the experiment with

| | | | |
|---|---|---|---|
| **1** (54%) | **2** (64%) | **3** (63%) | - |
| **4** (81%) | **5** (67%) | **6** (73%) | . |
| **7** (57%) | **8** (74%) | **9** (79%) | x |
| *# | **0** (73%) | English | > |

Nexus 5 (Ave. iden. rate: 70%)

| | | |
|---|---|---|
| **1** (70%) | **2** (50%) | **3** (59%) |
| **4** (70%) | **5** (46%) | **6** (56%) |
| **7** (53%) | **8** (48%) | **9** (67%) |
| + * # | **0** (41%) | > |

iPhone 5 (Ave. iden. rate: 56%)

Table 4.10: Identification rates of digits in Nexus 5 and iPhone 5

preparation and explanations. It took each user less than 10 minutes to finish the whole experiment.

### 4.7.3 Classification algorithm

After performing some informal analysis, among different classification methods (classification tree, kNN, ANN, etc.), we observed that ANN (Artificial Neural Network) works significantly better than other classifiers on our dataset. A neural network system for recognition is defined by a set of input neurons (nodes) which can be activated by the information of the intended object to be classified. The input can be either raw data, or pre-processed data from the samples. In our case, we have preprocessed our samples by building a feature vector as described in Section 4.5.4. Therefore, as input, *TouchSignatures*' ANN system receives a set of 150 features for each sample.

A neural network can have multiple layers and a number of nodes in each layer. Once the first layer of the nodes receives the input, ANN weights and transfers the data to the next layer until it reaches the output layer which is the set of the labels in a classification problem. For better performance and to stop training before overfitting, a common practice is to divide the samples into three sets: training, validation, and test sets.

We trained a neural network with 70% of our data, validated it with 15% of the records and tested it with the remaining 15% of our data set. We trained our data by using pattern recognition/classifying network with one hidden layer and 10,000 nodes. Pattern recognition/classifying networks normally use a scaled conjugate gradient (SCG) back-propagation algorithm for updating weight and bias values in training. SCG [87] is a fast supervised learning algorithm based on conjugate directions. The results of the second phase of *TouchSignatures* are obtained according to these settings.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** (54%) | (8%) | (0%) | (8%) | **2** (64%) | (12%) | (10%) | (5%) | **3** (63%) | - |
| (15%) | (0%) | (0%) | (0%) | (12%) | (0%) | (0%) | (0%) | (5%) | |
| (15%) | (8%) | (0%) | (0%) | (4%) | (0%) | (5%) | (5%) | (0%) | |
| - | (0%) | - | - | (0%) | - | - | (5%) | - | |
| (0%) | (0%) | (6%) | (7%) | (27%) | (0%) | (0%) | (0%) | (7%) | . |
| **4** (81%) | (0%) | (6%) | (0%) | **5** (67%) | (0%) | (0%) | (7%) | **6** (73%) | |
| (6%) | (0%) | (0%) | (0%) | (0%) | (0%) | (0%) | (0%) | (13%) | |
| - | (0%) | - | - | (0%) | - | - | (0%) | - | |
| (7%) | (0%) | (14%) | (0%) | (0%) | (0%) | (0%) | (0%) | (0%) | x |
| (14%) | (7%) | (0%) | (0%) | (0%) | (5%) | (0%) | (7%) | (3%) | |
| **7** (57%) | (0%) | (0%) | (5%) | **8** (74%) | (5%) | (0%) | (3%) | **9** (79%) | |
| - | (0%) | - | - | (11%) | - | (- | (7%) | - | |
| *# | | | (0%) | (0%) | (0%) | English | | | > |
| | | | (7%) | (0%) | (7%) | | | | |
| | | | (0%) | (7%) | (7%) | | | | |
| | | | - | **0** (73%) | - | | | | |

Table 4.11: Confusion matrices in Nexus 5

### 4.7.4 Results

Here, we present the output of the suggested ANN for Nexus 5 and iPhone 5, separately. Table 4.10 shows the accuracy of the ANN in classifying the digits presented in two parts for the two devices. The average identification rates for Nexus 5 and iPhone 5 are 70% and 56%, respectively. In general, the resolution of the data sequences on Android was higher than iOS. We recorded about 37 motion and 20 orientation measurements for a typical digit on Android, while there were only 15 for each sequence on iOS. This can explain the better performance of *TouchSignatures* on Android than on iOS. It is worth mentioning that attacks on iPhone 5 actually are the ones with the lowest sampling rates that we observed in Table 4.3 (20Hz for both motion and orientation). Interestingly, even with readings on the lowest available sampling rate, the attack is still possible.

In Tables 4.11 and 4.12, we show the identification results of each digit (bold in each cell), as well as confusion matrices on both devices. The general forms of the tables are according to Android and iOS numpads. As demonstrated, each digit is presented with all possible misclassifiable digits. As it can be observed, most misclassified cases are either in the same row or column, or in the neighbourhood of each expected digit.

Note that the probability of success in finding the actual digit will significantly improve with more tries at guessing the digit. In fact, while the chance of the attack

| | | |
|---|---|---|
| **1** (70%) (0%) (4%)<br>(4%) (0%) (0%)<br>(4%) (17%) (0%)<br>- (0%) - | (0%) **2** (50%) (0%)<br>(17%) (11%) (5%)<br>(5%) (11%) (0%)<br>- (0%) - | (0%) (6%) **3** (59%)<br>(0%) (6%) (12%)<br>(0%) (6%) (6%)<br>- (6%) - |
| (5%) (10%) (5%)<br>**4** (70%) (0%) (0%)<br>(0%) (5%) (0%)<br>- (5%) - | (8%) (8%) (0%)<br>(15%) **5** (46%) (8%)<br>(8%) (0%) (8%)<br>- (0%) - | (0%) (6%) (0%)<br>(0%) (19%) **6** (56%)<br>(6%) (6%) (0%)<br>- (6%) - |
| (13%) (0%) (13%)<br>(0%) (0%) (7%)<br>**7** (53%) (0%) (0%)<br>- (13%) - | (0%) (10%) (0%)<br>(0%) (5%) (10%)<br>(0%) **8** (48%) (5%)<br>- (24%) - | (0%) (0%) (10%)<br>(0%) (5%) (0%)<br>(10%) (5%) **9** (67%)<br>- (5%) - |
| + ∗ # | (0%) (6%) (0%)<br>(12% (0%) (12%)<br>(0% (15%) (12%)<br>- **0** (41%) - | > |

Table 4.12: Confusion matrices in iPhone 5

| Digits<br>Attempt No. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| First | 73% | 54% | 64% | 63% | 81% | 67% | 73% | 57% | 74% | 79% |
| Second | 80% | 69% | 76% | 74% | 88% | 93% | 87% | 71% | 84% | 86% |
| Third | 87% | 85% | 88% | 79% | 94% | 100% | 93% | 86% | 89% | 93% |
| Forth | 93% | 92% | 96% | 84% | 100% | 100% | 100% | 93% | 98% | 97% |
| Fifth | 100% | 100% | 100% | 89% | 100% | 100% | 100% | 100% | 100% | 100% |
| Sixth | 100% | 100% | 100% | 95% | 100% | 100% | 100% | 100% | 100% | 100% |
| Seventh | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% |

Table 4.13: Identification rate based on the number of guesses that the attacker makes on Nexus 5 for each digit separately

succeeding is relatively good on the first guess, it increases on further guesses as shown in Tables 4.13 and 4.14. Figure 4.7 shows the average identification rates based on the number of guesses in Nexus 5 and iPhone 5 compared to random guessing. As shown on the figure, *TouchSignatures* can predict the correct touched digits on average in almost 90% of the cases on Nexus 5 and 80% of the cases on iPhone 5 in the third guess.

The high identification rates prove the feasibility of the suggested attack by *TouchSignatures* and show that it is practical for a remote attacker to significantly reduce the search space for the user's PIN using JavaScript code.

| Digits<br>Attempt No. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| First | 41% | 70% | 50% | 59% | 70% | 46% | 56% | 53% | 48% | 67% |
| Second | 56% | 87% | 67% | 71% | 80% | 62% | 75% | 67% | 71% | 76% |
| Third | 69% | 91% | 78% | 76% | 85% | 69% | 81% | 80% | 81% | 86% |
| Forth | 81% | 96% | 89% | 82% | 90% | 78% | 88% | 93% | 90% | 90% |
| Fifth | 94% | 100% | 94% | 88% | 95% | 85% | 94% | 100% | 95% | 95% |
| Sixth | 100% | 100% | 100% | 94% | 100% | 92% | 100% | 100% | 100% | 100% |
| Seventh | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% |

Table 4.14: Identification rate based on the number of guesses that the attacker makes on iPhone 5 for each digit separately

| **Work** | **Sensor(s)** | **Rate** | **Access** |
|---|---|---|---|
| TapLogger [115] | Acc, Orientation | 36.4% | in-app |
| TouchLogger [26] | Orientation | 71.5% | in-app |
| **TouchSignatures** | **Motion, Orientation** | **77.0%** | **in-browser** |

Table 4.15: Identification rate of phase two of TouchSignatures (PIN) under the similar test condition as in-app attacks.

### 4.7.5 Comparison with related work

In this section we compare the second phase of *TouchSignatures*, the identification of PIN digits, with previous in-app sensor-based PIN identifiers. Among the work described in Table 4.1, we choose to compare *TouchSignatures* with TouchLogger [26], and TapLogger [115], since they use similar sensors for identifying digits on soft numerical keyboards.

Taplogger performs its experiments on Android devices and identifies 36.4% of the digit positions in the first attempt by using accelerometer and orientation sensors. On the other hand, TouchLogger is able to identify the digits with 71.5% accuracy on an Android device by using device orientation.

TouchLogger collects around 30 samples per digit from one user, while Taplogger has the input of one user for 20 random 16-digit sequences in 60 rounds. However, we noticed that in these works the data has been collected from only one user. In general, data obtained form a single user are more consistent than those collected from a diversified group of users. To verify this, we performed another experiment by simulating the same test condition as described above with the Android device (Nexus 5) and asked only one user to repeat the experiment 3 times. We collected 30 samples for each digit. The results are presented in Table 4.15. As expected, the identification rate of *TouchSignatures* increased to 77% in this situation, which is

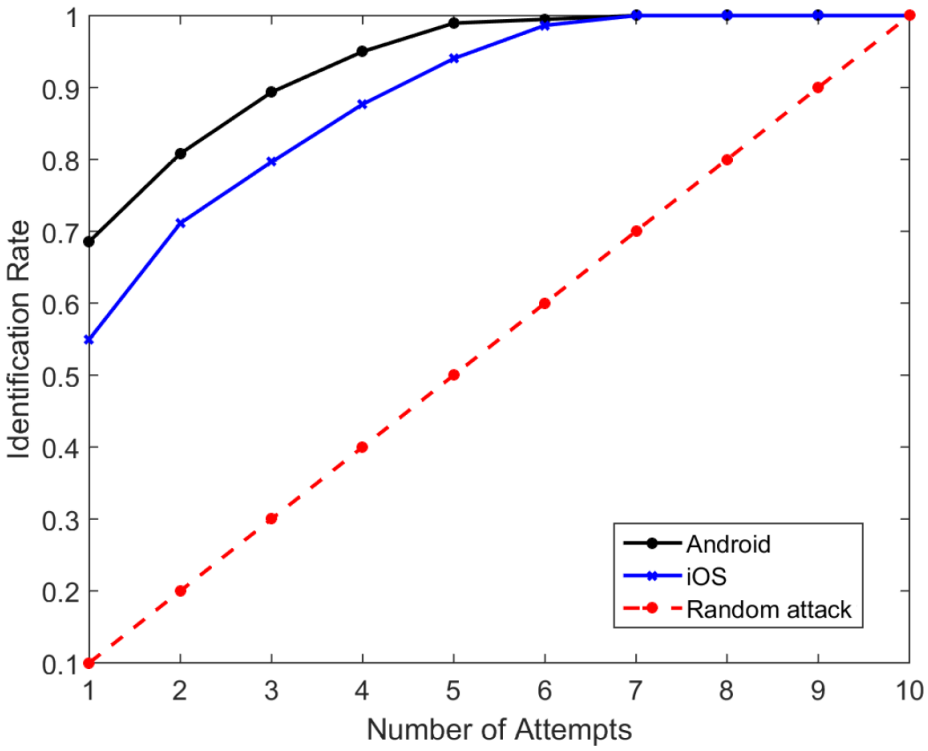


Figure 4.7: Average identification rate based on the number of attempts on Android and iOS vs. random guess

better than the results reported in TapLogger and TouchLogger.

Our results demonstrate the practicality of distinguishing the user's PIN by listening to sensor data via JavaScript code. Consequently, *TouchSignatures* highlights the limitations of the security policies in mobile operating systems and web browsers. As a result, urgent modifications are needed in updating the security policies for granting permissions to mobile web browsers to access sensor data.

## 4.8 PINLogger.js: Identifying full 4-digit PINs

In this section, we describe an advanced attack on user's PINs by introducing PINlogger.js. Similar to previous section, we consider an attacker who wants to learn the user's PIN tapped on a soft keyboard of a smartphone via side channel information. However, in opposite to single digits, in this section we consider 4-digit PINs since they are popular passwords used by users for many purposes such as unlocking phone, SIM PIN, NFC payments, bank cards, other banking services, gaming, and other personalised applications such as healthcare, insurance, etc.

In order to uncover when the user enters his PIN, we need to classify his touch

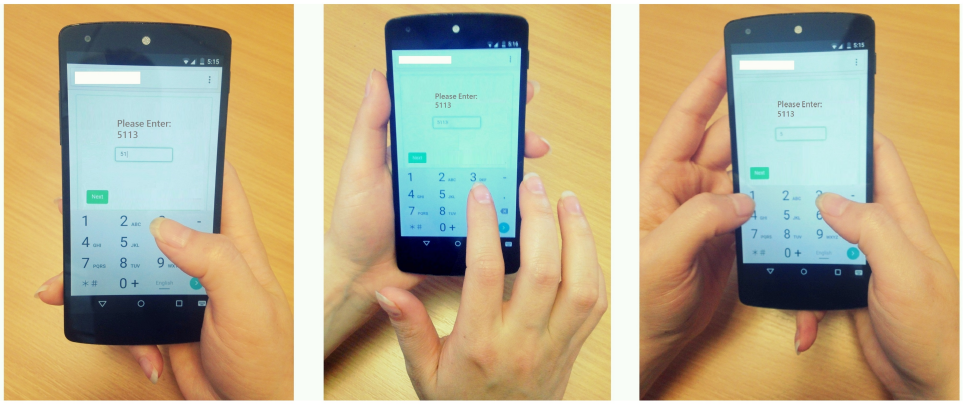

Figure 4.8: Different input methods used by the users for PIN entrance

actions such as click, scroll, and zoom. We already have shown in TouchSignatures [81, 82] that with the same sensor data and by applying classification algorithms, it is possible to effectively identify user's touch actions. Here, we consider a scenario after the touch action classification. In other words, our attacker already knows that the user is entering his PIN. Similar to TouchSignatures, unless explicitly noted, we consider a generic attack scenario which is not user-dependent. This means that we do not need to train our machine learning algorithm with the same user as the subject of the attack. Instead, we have a one-round training phase with data from multiple voluntary users. This approach has the benefit of not needing to trick individual users to collect data for training.

### 4.8.1 PINs set

Following the approach of Aviv et al. [17] and Spreitzer [100], we consider a set of 50 fixed PINs with uniformly distributed digits. We created these PINs in a way that all digits are repeated about the same time (around 20 times). As mentioned before, the data collection code is publicly available via the project's github page. The technical details of the data collection process and the collected data are publicly available too[14] (see Appendix B). These PINs include: 5113, 3268, 2917, 8460, 8508, 2427, 7497, 8030, 1887, 3695, 7203, 2486, 2641, 4051, 9422, 9336, 6034, 5586, 1095, 2098, 5159, 6491, 7107, 6343, 5779, 6875, 9431, 4654, 8331, 8160, 4971, 2274, 6138, 6575, 9602, 9951, 9496, 1035, 2278, 9140, 6258, 2880, 7772, 1096, 4303, 2516, 3484, 3200, and 1986.

[14] github.com/maryammjd/Reading-sensor-data-for-fifty-4digit-PINs

### 4.8.2 Experiments

Similar to our implementation in the previous phases, we implemented a web page with embedded JavaScript code in order to collect the data from voluntary users. On the client side, we developed a GUI in HTML5 which shows our 4-digit PINs to the users and activates a numpad for them to enter the PINs as shown in Figure 4.8.

We conducted our user studies using Chrome on an Android device (Nexus 5). The experiments and results are based on the collected data from 10 users, each entering all the 50 4-digit PINs for 5 times. Our voluntary participants were university students and staff and performed the experiments at university offices. We simply explained to them that all they needed was to enter a few PINs shown in a web page.

In relation to the environmental setting for the data collection, we asked the users to remain sitting in a chair while working with the phone. We did not require our users to hold the phone in any particular mode (portrait or landscape) or work with it by using any specific input method (using one or two hands). We let them choose their most comfortable posture for holding the phone and working with it as they do in their usual manner. While watching the users during the experiments, we noticed that all of our users used the phone in the portrait mode by default. Users were either leaning their hands on the desk or freely keeping them in the air. We also observed the following input methods used by the users.

- Holding the phone in one hand and entering the PIN with the thumb of the same hand (Figure 4.8, left).
- Holding the phone in one hand and entering the PIN with the fingers of the other hand (Figure 4.8, centre).
- Holding the phone with two hands and entering the PIN with the thumbs or fingers of both hands (Figure 4.8, right).

In the first two cases, users exchangeably used either their right hands or left hands in order to hold the phone. In order to simulate a real world data collection environment, we took the phone to each user's workspace and briefly explained the experiment to them, and let them complete the experiment without our supervision. All users found this way of data collection very easy and could finish the experiments without any difficulties.

### 4.8.3 Feature extraction

In order to build the feature vector as the input to our classifier algorithm, we consider both time domain and frequency domain features. We improve our suggested feature vectors in previous section by adding some more complex features such as the correlation between the measurements. This addition improves the results, as we will discuss in Section 4.8.5. As discussed before, 12 different sequences obtained from the collected data include orientation (ori), acceleration (acc), acceleration-including-gravity (accG), and rotation rate (rotR) with three sequences (either x, y and z, or $\alpha$, $\beta$ and $\gamma$) for each sensor measurement. As a pre-processing step and in order to remove the effect of the initial position and orientation of the device, we subtract the initial value in each sequence from subsequent values in the sequence.

We use these pre-processed sequences for feature extraction in time domain directly. In frequency domain, we apply the Fast Fourier transform (FFT) on the pre-processed sequences and use the transformed sequences for feature extraction. In order to build our feature vector, first we obtain the maximum, minimum, and average values of each pre-processed and FFT sequences. These statistical measurements give us $3 \times 12 = 36$ features in the time domain, and the same number of features in the frequency domain. We also consider the total energy of each sequence in both time and frequency domains calculated as the sum of the squared sequence values, i.e. $E = \sum v_i^2$ which gives us 24 new features.

The next set of features are in time domain and are based on the correlation between each pair of sequences in different axes. We have 4 different sequences; ori, acc, accG, and rotR, each represented by 3 measurements. Hence, we can calculate 6 different correlation values between the possible pairs; (ori, acc), (ori, accG), (ori, rotR), (acc, accG), (acc, rotR), and (accG, rotR), each presented in a vector with 3 elements. We use the Correlation coefficient function in order to calculate the similarity rate between the mentioned sequences. The correlation coefficient method is commonly used to compare the similarity of the shapes of two signals (e.g. [19]). Given two sequences $A$ and $B$ and $\mathrm{Cov}(A, B)$ denoting covariance between $A$ and $B$, the correlation coefficient is computed as below:

$$R_{AB} = \frac{\mathrm{Cov}(A, B)}{\sqrt{\mathrm{Cov}(A, A) \cdot \mathrm{Cov}(B, B)}} \tag{4.1}$$

The correlation coefficient of two vectors measures their linear dependence by using covariance. By adding these new 18 features, our feature vector consists of a total of 114 features.

### 4.8.4 Classification algorithm

Similar to the previous section, we apply a supervised machine learning algorithm by using an ANN system to solve this classification problem. The input of an ANN system could be either raw data, or pre-processed data from the samples. In our case, we have preprocessed our samples by building a feature vector as described before. Therefore, as input, our ANN receives a set of 114 features for each sample. As explained before, we collected 5 samples per each 4-digit PIN from 10 users. While reading the records, we realised that some of the PINs have been entered wrongly by some users. This was expected since each user was required to enter 250 PINs. Since we recorded both expected and entered PINs in our data collection, we could easily identify these PINs and exclude them from our analysis. Overall, out of 2500 records collected from 10 users, 12 of the PINs were entered wrongly. Hence we ended up with 2488 samples for our ANN.

The feature vectors are mapped to specific labels from a finite set: i.e. 50 fixed 4-digit PINs. We train and validate our algorithm with two different subsets of our collected data, and test the neural network against a separate subset of the data. We train the network with 70% of our data, validate it with 15% of the records and test it with the remaining 15% of our data set. We use a pattern recognition/classifying network in Matlab with one hidden layer and 1000 nodes.

### 4.8.5 Results

In this section we present the results of our attack on 4-digit PINs in two different forms: multiple-users mode, and same-user mode. We also train separate ANN systems to learn individual digits of PINs and compare these results with other works.

**Multiple-users mode** The second column of Table 4.16 shows the accuracy of our ANN trained with the data from all users. In this mode, the results are based on training, validating, and testing our ANN using the collected data from all of our 10 participants. As the table shows, in the first attempt PINlogger.js is able to infer the user's 4-digit PIN correctly with accuracy of 74.43%, and as expected it gets better in further attempts. By comparison, a random attack can guess a PIN from a set of 50 PINs with the probability of 2% in the first attempt, and 6% in three attempts.

**Same-user mode** In order to study the impact of individual training, we trained, validated and tested the network with the data collected from one user. We refer to this mode of analysis as the same-user mode. We asked our user to enter 50

| Attempts | Multiple-users | Same-user |
|---|---|---|
| One | 74% | 79% |
| Two | 86% | 93% |
| Three | 94% | 97% |

Table 4.16: PINlogger.js's PIN identification rates in different attempts

random PINs, each five times, and repeated the experiment for 10 times (rounds). The reason we have repeated the experiments is that the classifier needs to receive enough samples to be able to train the system. Interestingly, our user used all three different input methods shown in Figure 4.8 during the PIN entrance. As expected, our classifier performs better when it is personalized: the accuracy reaches 79.23% in the first attempt, and increases to 93.52% and 97.71% in two and three attempts, respectively.

In the same-user mode, convincing the users to provide the attacker with sufficient data for training customised classifiers is not easy, but still possible. Approaches similar to gaming apps such as Math Trainer[15] could be applied. Math-based CAPTCHAs are possible web-based alternatives. Any other web-based game application which segments the GUI similar to a numerical keypad will do as well. Nonetheless, in this thesis we mainly follow the multiple-users approach.

**Guessing the PIN from the entire PIN space.** One might argue that the attack should be evaluated against the whole 4-digit PIN space. However, we believe that the attack could still be practical when selecting from a limited set of PINs since users do not select their PINs randomly [22]. It has been reported that around 27% of all possible 4-digit PINs belong to a set of 20 PINs[16], including straightforward ones like '1111', '1234', or '2000'. Nevertheless, we present the results of our analysis of the attack against the entire search space for the two experiment modes discussed above. We considered 10 classes of the entered digits (0–9) from the data we collected on 4-digit PINs used in Section 4.8.5.

In the multiple-users mode, we trained, validated, and tested our system with data from all 10 users. In the same-user mode, we trained personalised classifiers for each user. Unlike the test condition of Section 4.8.5, we did not have to increase the number of rounds of PIN entry here since we had enough samples for each digit per user. In the same-user mode in this section, we used the average of the results of our 10 users. The average identification rates of different digits for the two different approaches are presented in Table 4.17.

[15] play.google.com/store/apps/details?id=com.solirify.mathgame
[16] datagenetics.com/blog/september32012/

| Attempts | Multiple-users | Same-user |
|---|---|---|
| One | 70% | 79% |
| Two | 83% | 90% |
| Three | 92% | 96% |

Table 4.17: Average digit identification rates in different attempts

| Features | Sensor | Access | Training | Rate on $i^{th}$ try | | |
|---|---|---|---|---|---|---|
| Work | | | | 1 | 2 | 5 |
| PIN Skimming [100] | Light | in-app | user-dep | NA | 50% | 65% |
| PIN Skimmer [99] | Cam, Mic | in-app | user-dep | NA | 30% | 50% |
| Keylogging Mic [88] | Mic, Gyr | in-app | use-dep | 94% | NA | NA |
| TapLogger [115] | Acc, Ori | in-app | user-dep | 40% | 75% | 100% |
| Acc side channel [17] | Acc | in-app | user-dep | 18% | NA | 43% |
| PINlogger.js | Motion, Ori | in-browser | user-indep | 74% | 86% | 98% |
| | | | user-dep | 79% | 93% | 99% |

Table 4.18: Comparison of PINlogger.js with related works

The results in our multiple-users mode indicate that we can infer the digits with a success probability of 70.75%, 83.27% and 92.06% in the first, second, and third attempts, respectively. This means that for a 4-digit PIN and based on the obtained sensor data, the attacker can guess the PIN from a set of $3^4 = 81$ possible PINs with a probability of success of $0.9206^4 = 71.82\%$. A random attack, however, can only predict the 4-digit PIN with the probability of 0.81% in 81 attempts. By comparison, PINlogger.js achieves a dramatically higher success rate than a random attacker.

Using a similar argument, in the same-user mode the success probability of guessing the PIN in 81 attempts is 85.46%. In the same setting, Cai and Chen report a success rate of 65% using accelerometer and gyroscope data [3] and Simon and Anderson's PIN Skimmer only achieves a 12% success rate in 81 attempts using camera and microphone [99]. Our results in digit recognition in this work are also better than what is achieved in TouchSignatures [82]. In summary, PINlogger.js performs better than all sensor-based digit-identifier attacks in the literature.

### 4.8.6 Comparison with related works

Obtaining sensitive information about users such as PINs based on mobile sensors has been actively explored by researchers in the field [70, 113]. In particular, there is a number of research which use mobile sensors through a malicious app running in the background to extract PINs entered on the soft keyboard of the mobile device. For example, GyroPhone, by Michalevsky et al. [84], shows that gyroscope data is

sufficient to identify the speaker and even parse speech to some extent. Other examples include Accessory [92] by Owusu et al. and Tapprints [85] by Miluzzo. They infer passwords on full alphabetical soft keyboards based on accelerometer measurements. Touchlogger [26] is another example by Cai and Chen [3] which shows the possibility of distinguishing user's input on a mobile numpad by using accelerometer and gyroscope. The same authors demonstrate a similar attack in [27] on both numerical and full keyboards. The only work which relies on in-browser access to sensors to attack a numpad is our previous work, TouchSignatures [82]. All of these works, however, aim for the individual digits or characters of a keyboard, rather than the entire PIN or password.

Another category of works directly target user PINs. For example, PIN skimmer by Simon and Anderson [99] is an attack on a user's numpad and PINs using the camera and microphone on the smartphone. Spreitzer suggests another PIN Skimming attack [100] and steals a user's PIN based on the measurements from the smartphone's ambient light sensor. Narain et al. introduce another attack [88] on smartphone numerical and alphabetical keyboards and the user's PINs and credit card numbers by using the smartphone microphone. TapLogger by Xu et al. [115] is another attack on the smartphone numpad which outputs the pressed digits and PINs based on accelerometer and orientation sensor data. Similarly, Aviv et al. introduce an accelerometer-based side channel attack on the user's PINs and patterns in [17]. We choose to compare PINlogger.js with the works in this category since they have the same goal of revealing the user's PINs. Table 4.18 presents the results of our comparison.

As shown in Table 4.18, PINlogger.js is the only attack on PINs which acquires the sensor data via JavaScript code. In-browser JavaScript-based attacks impose even more security threats to users since unlike in-app attacks, they do not require any app installation and user permission to work. Moreover, the attacker does not need to develop different apps for different platforms such as Android, iOs, and Windows. Once the attacker develops the JavaScript code, it can be deployed to attack all mobile devices regardless of the platform. Moreover, Touchlogger.js is the only work which uses the data coming from multiple users. By contrast, the results from other works are mainly based on training the classifiers for individual users. In other words, they assume the attacker is able to collect input training data from the victim user before launching the PIN attack. We do not have such an assumption as the training data is obtained from multiple users in the experiment. In terms of accuracy, with the exception of [88], PINlogger.js generally outperforms other works with an identification

rate of 74% in the first attempt. This is a significant success rate (despite that the sampling rate in-browser is much lower than that available in-app) and confirms that the described attack imposes a serious threat to the users' security and privacy.

## 4.9 Possible solutions

To be able to suggest appropriate countermeasures, we need to first identify the exact entity responsible for the access control policy in each situation. Mobile OS access control policy decides whether the browser gets access to the device motion and orientation sensor data in the first place, no matter if the browser is active or not. If access is provided, then mobile browser access control policy decides whether a web app gets access to the sensor data, no matter if the web app is open in the same tab and in the same segment, in the same tab but in a different segment, or in a different tab. Hence any effective countermeasure must address changes in both mobile OS and browser policies with respect to access to such sensor data.

One approach to protect user security would be to require the mobile OS to deny access to the browser altogether when the browser is not active, and require the browser to deny access to web content altogether when it is running in an inactive tab or in a segment of the page with the different web origin. However, this approach may be considered too restrictive as it will disallow many potential web applications such as activity monitoring for health and gaming.

A more flexible approach would be to *notify* the user when a web page is requesting access to such sensor data, and provide *control* mechanisms through which the user is able to set their preferences with respect to such requests. This is the approach currently taken by both the mobile operating systems and browsers with respect to providing access to the device location (i.e. GPS sensor data [110]) when a web page requests such access. We believe similar measures for device motion and orientation would be necessary in order to achieve a suitable balance between usability and security. Possible (mock-up) interfaces for this countermeasure, based on existing solutions for GPS sensor data, are presented in Figure 4.9. In particular, we think the user should have three options: either allow access to the browser (in the mobile OS setting) or web pages (in the browser setting) indefinitely, or allow access only when the user is working on the browser (in the mobile OS settings) or interacting with the web page (in the browser settings), or deny access indefinitely. These three options provided to the user seem to be neither too few to render the access control ineffective, nor too many to exhaust the user attention span.

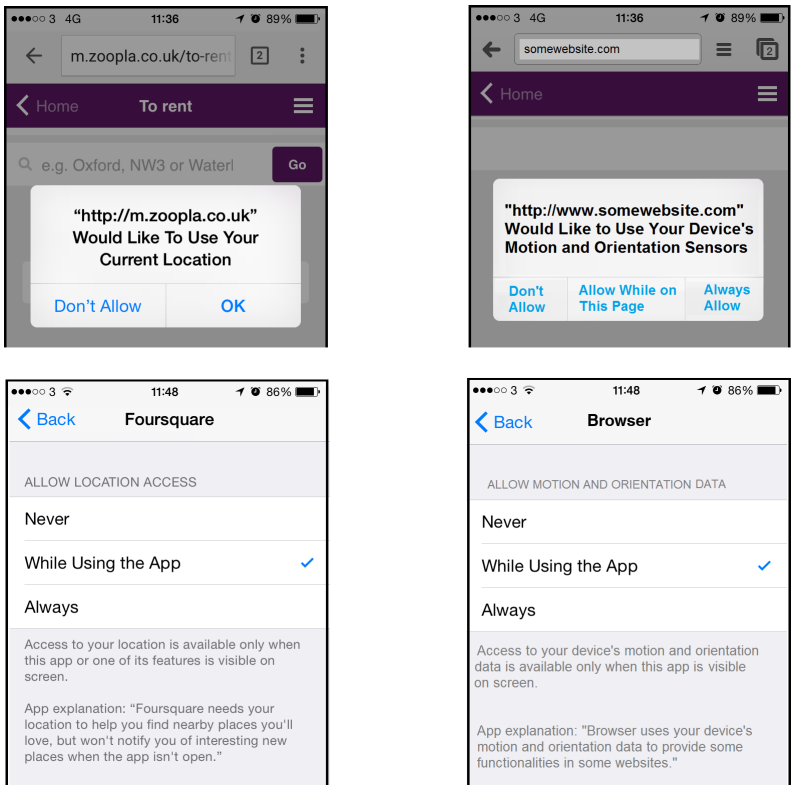


Figure 4.9: Left: The existing interfaces to allow the web page to access Geolocation in browser (top) and in mobile OS (down). Right: Our suggested mock-up interfaces to allow web page (top) and OS setting (down) to access Motion and Orientation data in browser

Furthermore, we believe raising this issue in the W3C specification would help the browsers to consider it in a more systematic and consistent way. Our suggestion for the new version of the specification is to include a section for security and privacy considerations and discuss these issues in that section properly.

## 4.10 Industry feedback

We reported the results of this research to the W3C community and mobile browser vendors including Mozilla, Opera, Chromium and Apple. We discussed the identified issues with them and received positive feedback as summarized below.

**Mozilla.** After we reported to Mozilla about Firefox allowing JavaScript access to sensor data within an iframe on Bugzilla, a senior platform engineer from Mozilla stated that: "Indeed, and it should be fixed consistently across all the browsers and

also the spec [W3C specification] needs to be fixed"[17]. Subsequently, a patch was proposed and implemented by Mozilla which was out on Firefox 46, April 26, 2016. test[18]. The impact of our reported bug has been categorised as "high" in Mozilla Foundation Security Advisory 2016-43.

**Chrome & Opera.** Opera uses the Chromium engine's implementation for device orientation. Therefore, fixing the problem on Opera is dependent on the fix on Chromium. We reported to both Chrome and Opera about their browsers allowing JavaScript access to sensor data within an iframe and in the other-tab. After discussing this issue on the Chromium forum, a security team member of Chrome stated that: "It [i.e. this research] sounds like a good reason to restrict it [i.e. sensor reading] from iframes"[19]. At the time of the writing this thesis, the status of our reported bug in Chromium is "assigned". Commenting on the JavaScript access to sensor data through *other-tab*, a member of the Opera security team forwarded their response to us via email stating that: "Opera on iOS giving background tabs access to the events does seem like an unwanted bug".

**Safari.** We reported to Apple about Safari allowing JavaScript access to sensor data within an iframe and also when the phone is locked. The Apple security team acknowledged the problem via email stating that: "We have reviewed your paper and are working on the mitigations listed in the paper". Accordingly, this problem was fixed by Apple in iOS 9.3[20] with an acknowledgement to our research.

**W3C.** After we disclosed the identified problems to the W3C community, the community acknowledged the attack vectors introduced in this work and stated that: "This would be an issue to address for any future iterations on this document [i.e. W3C specification on mobile orientation and motion [112]]". A security issue was be taken into account by W3C in this regard[21]. The community discussed this issue in their latest meeting and suggested to add a security section to the specification in response to the findings of our work[22]. Finally, a security section has been added to this specification with a reference to our research[23].

[17] bugzilla.mozilla.org/show_bug.cgi?id=1197901 (login required)
[18] mozilla.org/en-US/security/advisories/mfsa2016-43/
[19] bugs.chromium.org/p/chromium/issues/detail?id=523320#c18
[20] support.apple.com/en-gb/HT206166
[21] github.com/w3c/deviceorientation/issues/13
[22] w3.org/2015/10/26-geolocation-minutes.html#item03
[23] w3.org/TR/2016/CR-orientation-event-20160818/#security-and-privacy

## 4.11 Summary

In this chapter we introduced the first practical attack that was able to distinguish user touch actions as well as learning her PIN through JavaScript code embedded in a web page. We designed and implemented *TouchSignatures* and *PINLogger.js*: two simple and effective JavaScript-based attacks which when loaded within the browser were able to listen to the device orientation and motion sensor data streams and send the data back to a remote server for analysis. We demonstrated that our attack systems were able to distinguish different user touch actions through a $k$-NN classifier, and PINs through ANN system, respectively. The results show that our attack can classify user touch actions and identify her PINs with high success rates.

Our results highlight major shortcomings in W3C standards, mobile operating systems, and browsers access control policy with respect to user security. As a countermeasure which strikes a balance between security and usability, we suggest that device orientation and motion data be treated similarly to GPS sensor data. Effective user notification and control mechanisms for access to such sensor data should be implemented both in mobile operating systems and in mobile browsers. The positive industry feedback confirms that serious damage could be caused exploiting the introduced attack vectors. As a matter of fact, some of the browser vendors such as Mozilla and Apple have already provided the mitigations suggested in this work. Moreover, as a result of our communication, W3C has added a new section to the associated specification named: "Security and privacy considerations".

During our discussions with the mobile industry, we realised that our solution might not be the most usable way to manage sensors on mobile platforms. In particular, Google Chrome was not convinced enough to fix the problem as we suggested, as stated by one of the team members: "It's tempting to just copy mobile safari, but we're concerned about limiting valuable scenarios. Eg. embedded spherical videos, embedded maps using orientation data, etc.[f].

On the other hand, we discussed the user opt-in option with Chrome. However, we agree with them that: "opt-in is a pretty poor tradeoff" since it needs lots of infrastructural changes, it doesn't address a lot of the scenarios, and in long term it might not even decrease the amount of risk since the users would get used to the permission messages and skip them without realising the risks.

We believe that usability plays a critical role when it comes to a practical solution supported by industry. For this reason, in the next chapter, we study the human

dimensions of mobile sensors to provide a better insight for a practical solution for the problem of sensor management.

# Chapter 5

# Human Dimensions of Mobile Sensors Security

## 5.1 Chapter overview

In this chapter, we study users' perception of the risks associated with mobile phone sensors. With the technical understanding of the information leakage caused by mobile phone sensors, we study users' perception of the risks associated with these sensors. We design user studies to measure the general familiarity with different sensors and their functionality, and to investigate how concerned users are about their PIN being discovered by an app that has access to all these sensors. Our studies show that there is significant disparity between the actual and perceived levels of threat with regard to the compromise of the user PIN. We confirm our results by interviewing our participants using two different approaches, within-subject and between-subject, and compare the results. We discuss how this observation, along with other factors, renders many academic and industry solutions ineffective in preventing such side channel attacks.

The work in this chapter (with some parts of the previous chapter) was published as follows under the supervision of Dr. Hao and Dr. Shahandashti. Ehsan Toreini and Dr. Shahandashti helped with participant recruitment and interviews of this chapter.

- M. Mehrnezhad, E. Toreini, S. F. Shahandashti, F Hao, "Stealing PINs via Mobile Sensors: Actual Risk versus User Perception", The 1st European Workshop on Usable Security, EuroUSEC 2016, Workshop at the Privacy Enhancing Technologies Symposium (PETS 2016), July 18, 2016, Germany.

- M. Mehrnezhad, E. Toreini, S. F. Shahandashti, F Hao, "Stealing PINs via Mobile Sensors: Actual Risk versus User Perception", International Journal of Information Security, Springer, April 2017, Pages 1-23.

## 5.2 Introduction

**Actual and perceived risks of mobile sensors.** We showed in earlier chapters, that via different means e.g. in-app and in-browser access, we can attack users' privacy and security based on mobile sensors such as motion and orientation. For example, many popular browsers such as Safari, Chrome, Firefox, Opera and Dolphin have already implemented access to the above sensor data. As we demonstrated in previous chapters, all of these mobile browsers allow such access when the code is placed in any part of the active tab including *iframes* (Figure 5.1, a). In some cases such as Chrome and Dolphine on iOS, an inactive tab including the sensor listeners

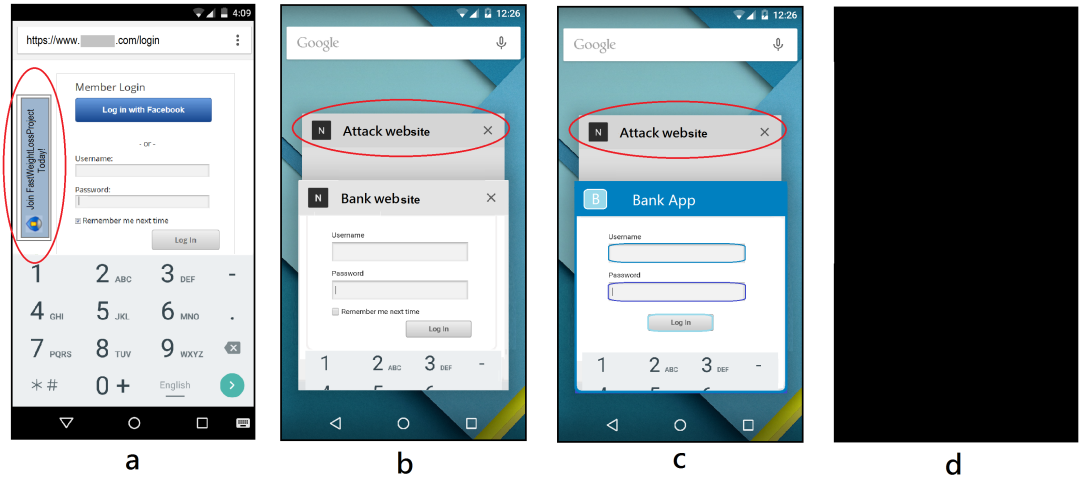


Figure 5.1: Potential JavaScript-based attack scenarios; a) the malicious code is loaded in an iframe and the user is on the same tab, b) the attack tab is already open and the user is on a different tab, c) the attack content is already open in a minimised browser, and the user is on an installed app, d) the attack content is already open in a (minimised) browser, and the screen is locked. The attacker listens to the side channel motion and orientation measurements of the victim's mobile device through JavaScript code, and uses machine learning methods to discover the user's sensitive information such as activity types and PINs.

have access to the sensor measurements as well (Figure 5.1, b). Even worse, some browsers such as Safari allow the inactive tabs to access the sensor data, when the browser is minimised (Figure 5.1, c), or even when the screen is locked (Figure 5.1, d). Mobile operating systems and browsers do not seem to be implementing consistent access control policies in regard to mobile orientation and motion sensor data and other sensors (see Table 1.2). As we showed in Table 4.18, researchers have already shown the risks of mobile sensors including the movement sensors to the users' PINs.

While sensors on mobile platforms are getting more powerful, and starting to collect more information about the users and their environment, we want to evaluate the general knowledge about these sensors among the mobile users. We are particularly interested to know the level of concern people may have about these sensors being able to threaten their privacy and security.

**Contributions.** In this chapter, we contribute to the study of human dimensions of mobile sensors security as follows:

- We conduct user studies to investigate users' understanding about these sensors and also their perception of the security risks associated with them. We show

| Android motion sensors | Description | Unit | W3C def. |
|---|---|---|---|
| Accelerometer | Acceleration force along 3 axes | $m/s^2$ | Acceleration with gravity |
| Gravity | Force of gravity along 3 axes | $m/s^2$ | NA |
| Gyroscope | Rate of rotation around 3 axes | rad/s | Rotation rate |
| Uncalibrated gyroscope | Rate of rotation (no drift compensation), and<br>Estimated drift around 3 axes | rad/s<br>rad/s | NA<br>NA |
| Linear accelerometer | Acceleration force excluding gravity along 3 axes | $m/s^2$ | Acceleration |
| Rotation vector | Rotation vector component along 3 axes | Unitless | NA |
| Step counter | Number of user's steps since last reboot | Steps | NA |

Table 5.1: Motion sensors supported by Android and their corresponding W3C definitions

that users in fact have fewer security concerns about these sensors comparing to more well-known ones.

- We study and challenge current suggested solutions, and discuss why our studies show they cannot be effective. We argue that a usable and secure solution is not straightforward and requires further research.

## 5.3 Sensor management complexity

Although reports of side channel attacks based on the in-browser access to mobile sensors via JavaScript are relatively recent, similar attacks via in-app access to mobile sensors have been known for years. Yet the problem has not been fixed. Here, we discuss some of potential reasons why such a vulnerability has remained unfixed for a long time.

### 5.3.1 Unmanaged sensors

In an attempt to explain multiple sensor-related in-app vulnerabilities, Xu et al. argue that "the fundamental problem is that sensing is unmanaged on existing smartphone platforms" [115]. There are multiple in-app side-channel attacks that support this

| **Android position sensors** | **Description** | **Unit** | **W3C def.** |
|---|---|---|---|
| Game rotation vector | Rotation vector component along 3 axes | Unitless | NA |
| Geomagnetic rotation vector | Rotation vector component along 3 axes | Unitless | NA |
| Geomagnetic magnetic field | Geomagnetic field strength along 3 axes | $\mu T$ | NA |
| Uncalibrated magnetic field | Geomagnetic field strength (no hard iron calibration) | $\mu T$ | NA |
| | and Iron bias estimation along 3 axes | $\mu T$ | NA |
| Orientation | Angles around 3 axes | Degrees | Orientation |
| Proximity | Distance from object | cm | NA |

Table 5.2: Position sensors supported by Android and their corresponding W3C definitions. Note: Orientation sensor was deprecated in Android 2.2 (API Level 8)

argument, as we discussed in the previous section. Our work shows that the problem of in-app access to "unmanaged sensors" is now spreading to in-browser access. Here we present the "unmanaged" motion and orientation sensor case which shows how the technical mismanagement of these sensors causes serious user privacy consequences when it comes to unregulated access to such sensors via JavaScript.

**W3C vs. Android.** According to W3C specifications [112], the motion and orientation sensor streams are not raw sensor data, but rather high-level data which are agnostic to the underlying source of information. Common sources of information include gyroscopes, compasses and accelerometers. In Tables 5.1 and 5.2, we present raw (low-level) and synthesized (high-level) motion sensors supported by Android [52] along with their descriptions and units, as well as their corresponding W3C definitions [112].

As it can be seen from the tables, different terminologies have been used for describing the same measurements in-app and in-browser. For example, while in-app access uses the raw sensor terminology, i.e., accelerometer, gyroscope, magnetic field, the in-browser access uses synthesized sensor terminology, i.e., motion and orientation [112]. This creates confusion for users (as we will explain later) and developers (as we experienced it ourselves). One of the W3C's specifications on mobile sensors, "Generic Sensor API" [51], dedicates a few sections to the issue of naming sensors, and low-level and high-level sensors. It discusses how the terminology for in-browser access has been high-level so far. It also mentions that the low-level use cases are increasingly popular

among the developers. As stated in this specification: "The distinction between high-level and low-level sensor types is somewhat arbitrary and the line between the two is often blurred". And, "Because the distinction is somewhat blurry, extensions to this specification are encouraged to provide domain-specific definitions of high-level and low-level sensors for the given sensor types they are targeting". We believe due to the rapid increase of mobile sensors, it is necessary to come up with a consistent approach.

### 5.3.2 Unknown sensors

We believe another contributing factor is that users seem to be less familiar with the relatively newer (and less advertised) sensors such as motion and orientation, as opposed to their immediate familiarity with well-established sensors such as camera and GPS. For example, a user has asked this question on a mobile forum: "... What benefits do having a gyroscope, accelerometer, proximity sensor, digital compass, and barometer offer the user? I understand it has to do with the phone orientation but am unclear in their benefits. Any explanation would be great! Thanks!"[1].

We design and conduct user studies in this work in order to investigate to what extent are these sensors and their risks known to the users.

**List of mobile sensors**. We prepared a list of different mobile sensors by inspecting the official websites of the latest iOS and Android products, and the specifications that W3C and Android provide for developers. We also added some extra sensors as common sensing mobile hardware which are not covered before.

- iPhone 6[2]: Touch ID, Barometer, Three-axis gyro, Accelerometer, Proximity sensor, Ambient light sensor.
- Nexus 6P[3]: Fingerprint sensor, Accelerometer, Gyroscope, Barometer, Proximity sensor, Ambient light sensor, Hall sensor, Android Sensor hub.
- Android [52]: Accelerometer, Ambient temperature, Gravity (software or hardware), Gyroscope, Light, Linear Acceleration (software or hardware), Magnetic Field, Orientation (software), Pressure, proximity, Relative humidity, Rotation vector (Software or Hardware), Temperature.

[1] forums.androidcentral.com/verizon-galaxy-nexus/171482-barometer-accelerometer-how-they-useful.html

[2] apple.com/uk/iphone-6/specs/

[3] store.google.com/product/nexus_6p

- W3C[4] [112]: Device orientation (software), Device motion (software), Ambient light, Proximity, Ambient temperature, Humidity, Atmospheric Pressure.
- Extra sensors (Common sensing hardware): Wireless technologies (WiFi, Bluetooth, NFC), Camera, Microphone, Touch screen, GPS.

Unless specified otherwise, all the listed sensors are hardware sensors. We added the last category of the sensors to this list since they indeed sense the device's surrounding although in different ways. However, they are neither counted as sensors in mobile product descriptions, nor in technical specifications. These sensors are often categorised as OS resources [114], and hence different security policies apply to them.

## 5.4 User studies on general knowledge about mobile sensors

In this section, we aimed to observe the amount of knowledge that mobile users have about mobile sensors. We prepared a list of sensors based on what we explained above and asked volunteer participants to rate the level of their familiarity with each sensor. All of our experiments and user studies were approved by Newcastle University's ethical committee.

### 5.4.1 Recruitment and participants demography

We recruited 60 participants (in two groups as explained in Section 5.5) to take part in this study via different means including mailing lists, social networking, vocational networks, and distributing flyers in different places such as different schools in the university, colleges, local shops, churches and mosques. A sample of our call for participation is available in Fin. 5.2.

Among our participants, 28 self-identified themselves as male and 32 as female, from 18 to 67 years old, with a median age of 33.85. None of the participants were studying or working in the field of mobile sensor security. Our university participants were from multiple degree programs and levels, and the remaining participants worked in a different range of fields. Moreover, our participants owned a wide range of mobile devices, and had been using a smartphone/tablet for 5.6 years on average. Our participants were from different countries, and all could speak English. We interviewed our participants at a university office and gave each an Amazon voucher

[4] w3.org/2009/dap/


Figure 5.2: Sample of flyer distributed for participant recruitment.

(worth £10) at the end for their participation. Details of the interview template can be found in Table 5.3.

### 5.4.2 Study approach

For a list of 25 different sensors, we used a five-point scale self-rated familiarity questionnaire as used in [62]: "I've never heard of this", "I've heard of this, but I don't know what this is", "I know what this is, but I don't know how this works", "I know generally how this works", and "I know very well how this works". The list of sensors was randomly ordered for each user to minimize bias. In addition, we needed to observe the experiments to make sure users were answering the questions based on their own knowledge in order to avoid the effect of processed answers. Full descriptions of all studies are provided in Appendix D.

We went through the questionnaires filled by our participants and analysed them in an Excel file. We have published this Excel file on the author's homepage. In this file, we have five separate sheets representing the results of different parts of this chapter (sheet 1: general sensor knowledge- study one, sheet 2: perceived concern before knowing the sensor description- study one, sheet 3: perceived concern after knowing the sensor description- study one, sheet 4: general sensor knowledge- study two, and sheet 5: perceived concern after knowing the sensor description- study two). Each sheet includes a list of 25 different sensors and the values chosen by different users for each of these sensors. We coded the options of the questionnaire of this section in the following form: "I've never heard of this" = 1, "I've heard of this, but I don't know what this is" = 2, "I know what this is, but I don't know how this works" = 3, "I know generally how this works" = 4, and "I know very well how this

| Sex | Age | Job/Bg | Mobile (y) | Sex | Age | Job/Bg | Mobile (y) |
|---|---|---|---|---|---|---|---|
| f | 23 | Civil Eng. | Nokia (0) | f | 27 | Teacher | HTC(3) |
| f | 28 | Customer Sup. | HTC (2) | m | 30 | Services | iPhone (4) |
| f | 22 | Media | Sony (3) | m | 26 | Computer | Samsung (7) |
| m | 43 | IT | iPhone (9) | m | 30 | Teacher | Blackberry(7) |
| f | 27 | Media | iPhone (9) | m | 52 | Nanotech | Nokia (0) |
| m | 18 | Mathematics | Samsung (3) | m | 41 | Nanotech | HTC (10) |
| f | 30 | Management | iPhone (7) | m | 47 | Lecturer | Samsung (2) |
| m | 22 | Medical | iPhone (10) | f | 39 | Physics | iPhone (4) |
| f | 27 | Human Mng. | Huawei (9) | f | 31 | Biology | Samsung(10) |
| f | 21 | Literature | Samsung (4) | m | 39 | Student | iPhone (6) |
| m | 35 | Media | Samsung (6) | f | 30 | Civil Eng. | iPhone (5) |
| f | 20 | Languages | Samsung (3) | m | 20 | Student | Samsung (4) |
| f | 59 | Services | iPhone (3) | f | 52 | Admin | Samsung (3) |
| m | 40 | IT | LG (7) | f | 30 | Admin | Samsung (5) |
| m | 21 | Biomedical | Samsung (4) | f | 58 | Admin | iPhone (12) |
| f | 22 | Biomedical | OnePlus (6) | f | 44 | Admin | Samsung (3) |
| m | 30 | Civil Eng. | Samsung (3) | f | 27 | Student | Motorola (5) |
| m | 29 | Geodesy | Samsung (7) | f | 47 | Services | iPhone (5) |
| m | 28 | Medical | Sony (5) | m | 67 | Teacher | Nokia (0) |
| f | 38 | Computer | Samsung (5) | m | 23 | Student | Nexus (5) |
| f | 30 | Animation | iPhone (9) | m | 46 | Cable Maker | iPhone (5) |
| f | 56 | Business Mng. | iPhone (11) | m | 35 | Services | Samsung (5) |
| f | 29 | Admin | Samsung (5) | f | 39 | Admin | iPhone(5) |
| f | 30 | Admin | Samsung (6) | f | 24 | Student | Gionee (3) |
| m | 47 | Driving Inst. | Sony (11) | f | 34 | Education | iPhone (4) |
| f | 28 | Admin | Motorola (7) | m | 32 | Student | OnePlus (6) |
| m | 40 | Education | LG (5) | f | 37 | Researcher | Honor (3) |
| m | 32 | Computer | iPhone (6) | m | 33 | Management | iPhone(12) |
| f | 25 | Law | HTC (3) | f | 33 | Math | Samsung (3) |
| m | 30 | Student | Nexus (5) | m | 27 | Student | iPhone (18) |

Table 5.3: Participants' self-reported demographics in the two studies, (y) indicates the years of owning a smartphone

works" = 5 (similarly, in Section 5.5, we use the following coding: "Not concerned" = 1, "A little concerned" = 2, "Moderately concerned" = 3, "Concerned" = 4, and "Extremely concerned" = 5).

Note that although we use numbers for saving the results of our interviews in an Excel file, we never apply any quantitative calculation on these numbers. We only use the counts of these values for the stacked charts and the Spearman's correlation values presented in the following sections.

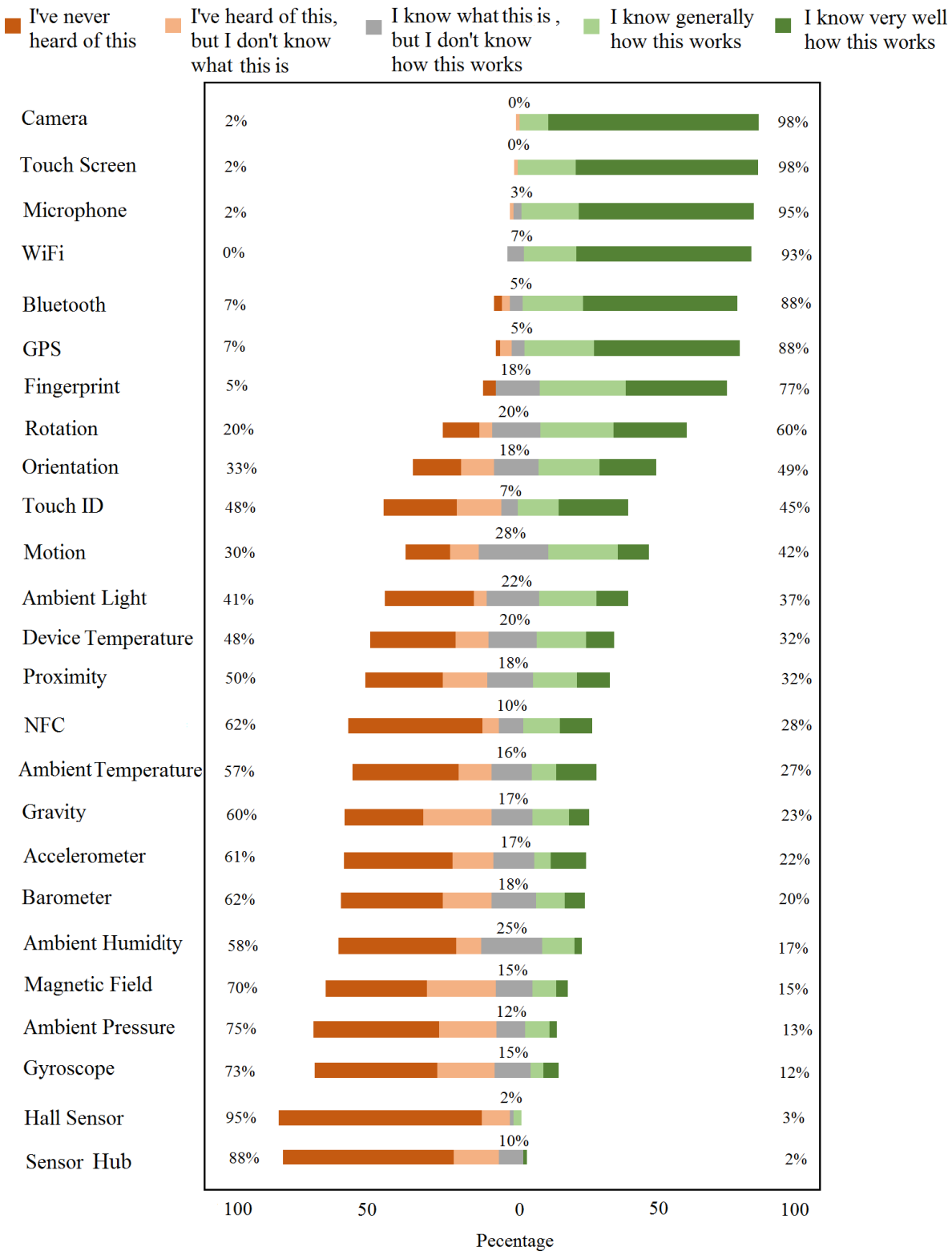


Figure 5.3: Level of self-declared knowledge about different mobile sensors

### 5.4.3 Findings

Fig. 5.3 summarizes the results of this study in the form of a stacked chart. This figure shows the level of self-declared knowledge about different mobile sensors. The question was: "To what extent do you know each sensor on a mobile device?" Sensors are ordered based on the aggregate percentage of participants declaring they know generally or very well how each sensor works. This aggregate percentage is shown on the right hand side. In the case of equal aggregate percentage, the sensor with a bigger share on being known very well by the participants is shown earlier. Our participants were generally surprised to hear about some sensors and impressed by the variety. As one may expect, newer sensors tend to be less known to the users in comparison to older ones. In particular, our participants were generally not familiar with ambient sensors. Although some of our participants knew the ambient sensors in other contexts (e.g. thermostats used at home), they could not recognise them in the context of a mobile device.

Low-level hardware sensors such as accelerometer and gyroscope seem to be less known to the users in comparison with high-level software ones such as motion, orientation, and rotation. We suspect that this is partly due to the fact that the high-level sensors are named after their functionalities and can be more immediately related to user activities.

We also noticed that a few of the participants knew some of the low-level sensors by name but they could not link them to their functionality. For example, one of our participants who knew almost all of the listed sensors (except hall sensor and sensor hub) stated that: "When I want to buy a mobile [phone], I do a lot of search, that is why I have heard of all of these sensors. But, I know that I do not use them (like accelerometer and gyroscope)".

On the other hand, as the functionalities of mobile devices grow, vendors quite naturally turn to promote the software capabilities of their products, instead of introducing the hardware. For example, many mobile devices are recognised for their gesture recognition features by the users, however the same users might not know how these devices provide such a feature. For instance, one of the participants commented on a feature on her smartphone called "Smart Stay"[5] as follows: "I have another sensor on my phone: Smart Stay. I know how it works, but I don't know which sensors it uses".

[5]samsung.com/us/support/answer/ANS00035658/234302/SCH-R950TSAUSC

## 5.5 User studies on risk perception of mobile sensors

In this section, we study the participants' risk perception of mobile sensors. There have been several studies on risk perception addressing different aspects of mobile technology. Some works discuss the risks that users perceive on smartphone authentication methods such as PINs and patterns [56], TouchID and Android face unlock [35], and implicit authentication [64]. Other works focus on the privacy risks of certain sensors such as GPS [18]. In [94], Raji et al. show users' concerns (on disclosure of selected behaviours and contexts) about a specific sensor-enabled device called AutoSense[6]. To the best of our knowledge, the research presented in this work is the first that studies the user risk perception for a comprehensive list of mobile sensors (25 in total). We limit our study to the level of perceived risks users associate with their PINs being discovered by each sensor. The reasons we chose PINs are that first, finding one's PIN is a clear and intuitive security risk, and second, we can put the perceived risk levels in context with respect to the actual risk levels for a number of sensors as described in Table 4.18.

For this study, we divide our 60 participants into two groups, and studied the two group separately using two different approaches: within-subject and between-subject. In the within-subject study, we interviewed 30 participants for all parts of the study. In contrast, in the between-subject study, we interviewed a new group of 30 participants, and we later compared the results with the previous group. By these two approaches, we aim to measure differences (after informing users on descriptions of sensors) within a participant and between participants, respectively.

### 5.5.1 Study one: within-subject

In this approach, we asked 30 participants to rate the level of risk they perceive for each sensor in regards to revealing their PINs in two phases. In phase one, we gave the same sensor list (randomized for each user). We described a specific scenario in which a game app which has access to all these sensors is open in the background and the user is working on his online banking app, entering a PIN. We used a self-rated questionnaire with five-point scale answers following the same terminology as used in [94]: "Not concerned", "A little concerned", "Moderately concerned", "Concerned", and "Extremely concerned". During this phase, we asked the users to rely on the information that they already had about each sensor (see Appendix D for details).

[6]sites.google.com/site/autosenseproject/

In the second phase, first we provided the participants with a short description of each sensor and let them know that they can ask further questions until they feel confident that they understand the functionality of all sensors. Participants could use a dictionary on their device to look at the words that were less familiar to them. Afterwards, we asked the participants to fill in another copy of the same questionnaire on risk perceptions (details in Appendix D). Participants could keep the sensor description paper during this phase to refer to it in the case they forgot the description of certain sensors.

### 5.5.2 Study two: between-subject

In this study, first we gave the description of the sensors to our second group of 30 participants and similar to previous study we gave them enough time to familiarize themselves with the sensors and to ask as many questions as they wanted until they felt confident about each sensor. Then, we presented the participants with the questionnaire on risk perceptions (details in Appendix D). Similar to our previous study, participants could keep the sensor description paper while filling in this questionnaire.

### 5.5.3 Intuitive risk perception

The results of our within-subject study are presented in Fig. 5.4. This stacked chart presents the users' perceived risk for different mobile sensors for the same group of users before (top bars) and after (bottom bars) being presented with descriptions of sensors. The results of our between-subject study are presented in Fig. 5.5. Note that this figure represents the risk perception of group one of our participants before knowing the sensors descriptions, and group two of participants after knowing the sensors descriptions. For both figures, the question was: "To what extent are you concerned about each sensor's risk to your PIN?", sensors are ordered based on the aggregate percentage of participants declaring they are either concerned or extremely concerned about each sensor before seeing the descriptions. This aggregate percentage is the first value presented on the right hand side. In the case of equal aggregate percentage, the sensor with a bigger share on being perceived extremely concerned by the participants is shown earlier.

We make the following observations from the results of the experiment.

**Touch Screen.** Although our participants rated touch screen as one of the most risky sensors in relation to a PIN discovery scenario, still about half of our participants were either moderately concerned, a little concerned, or not concerned at all. Through

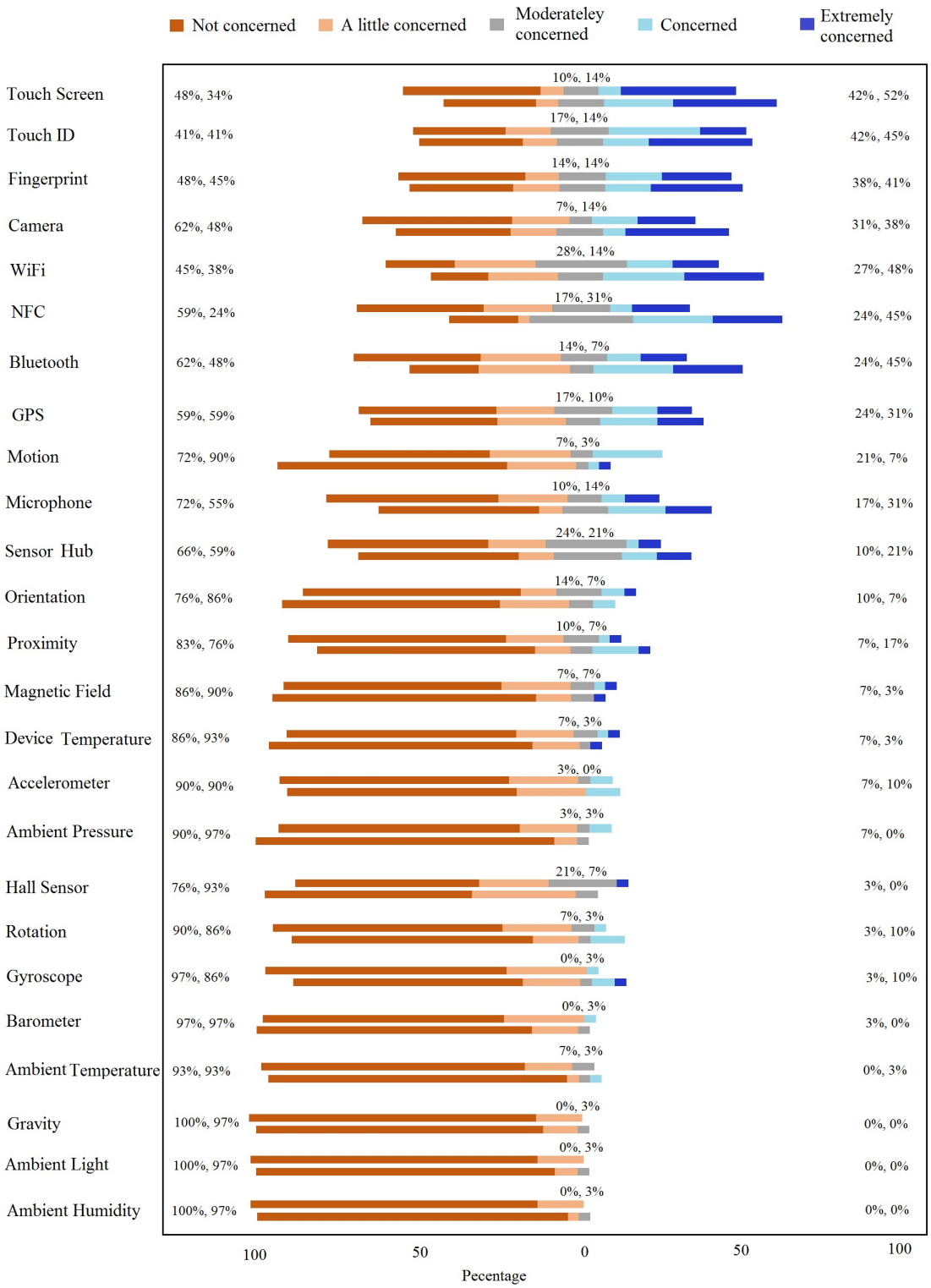


Figure 5.4: Users' perceived risk for different mobile sensors for within-subject approach

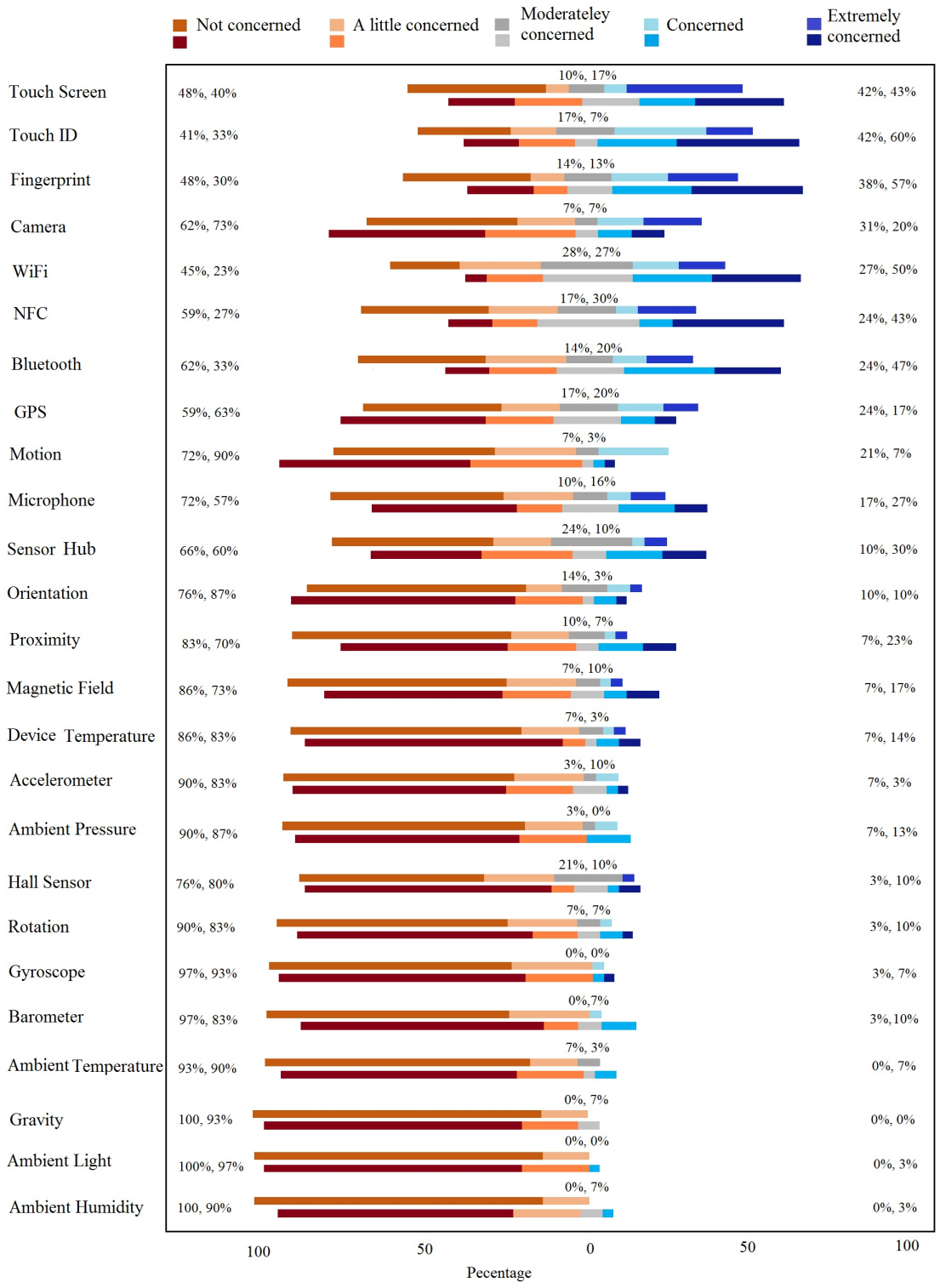


Figure 5.5: Users' perceived risk for different mobile sensors for between-subject approach

our conversations with the users, we received some interesting comments, e.g. "Why any of these sensors should be dangerous on an app while I have officially installed it from a legal place such as Google Play?", and "As long as the app with these sensors is in the background, I have no concern at all". It seems that a more general risk model in relation to mobile devices is affecting the users' perception in regard to the presented PIN discovery threat. This fact can be a topic of research on its own, and is out of the scope of this work.

**Communicational Sensors.** One category of the sensors which users are relatively more concerned about includes WiFi, Bluetooth and NFC. For example one of the participants commented that: "I am not concerned with physical [motion, orientation, accelerometer, etc.]/ environmental [light, pressure, etc.] sensors, but network ones. Hackers might be able to transfer my information and PIN". These sensors appearing more risky to the users is understandable since we asked them to what extent they were concerned about *each sensor* in regard to the PIN discovery.

**Identity-related Sensors.** Another category which has been rated more risky than others contains those sensors which can capture something related to the user's identity i.e. fingerprint, TouchID, GPS, camera, and microphone. Despite that we described a PIN-related scenario, our participants were still concerned about these sensors. This was also pointed out by a few participants through the comments. For example a user stated: "..., however, GPS might reveal the location along with the user input PIN that has a risk to reveal who (and where) that PIN belongs to. Also the fingerprint/TouchID might recognize and record the biometrics with the user's PIN". Some of these sensors such as GPS, fingerprint, and TouchID, however, can not cause the disclosure of PINs on their own. Hence, the concern does not entirely match the actual risk. Similar to the discussion on touch screen, we believe that a more general risk model on mobile technology influences the users to perceive risk on specific threats such as the one we presented to them.

**Environmental Sensors.** The level of concern on ambient sensors (humidity, light, pressure, and temperature) is generally low and stays low after the users are provided with the description of the sensors (see Fig. 5.4). In many cases, our users expressed that they were concerned about these sensors simply because they did not know them: "[now that I know these sensors,] I am quite certain that movement/environmental sensors would not affect the security of personal id/passwords etc.". In fact, researchers have reported that it is possible to infer the user's PIN using the ambient light sensor data [100], although, to our knowledge, exploits of other environmental sensors have not been reported in the literature.

**Movement Sensors.** On the sensors related to the movement and the position of the phone (accelerometer, gyroscope, motion, orientation, and rotation), the users display varying levels of the risk perceptions. In some cases they are slightly more concerned, but in others they are less concerned once they know the functionality. Some of our users stated that since they did not know these sensors, they were not concerned at all, but others were more concerned when they were faced with new sensors. Overall, knowing, or not knowing these sensors has not affected the perceived risk level significantly, and they were rated generally low in both cases.

**Motion and Orientation Sensors.** The sensors which we used in our attack, namely orientation, rotation, and motion, have not been generally scored high for their risk in revealing PINs. Users do not seem to be able to relate the risk of these sensors to the disclosure of their PINs, despite that they seem to have an average general understanding about how they work. On hardware sensors such as accelerometer and gyroscope, the risk perception seems to be even lower. A few comments include: "In my everyday life, I don't even think about these [movement] sensors and their security. There is nothing on the news about their risk", and "I have never been thinking about these [movement] sensors and I have not heard about their risk". On the other hand, some of the participants expressed more concerns for sensors that they were familiar with, as one wrote, "You always hear about privacy stuff for example on Facebook when you put your location or pictures". Similarly, it seems that having a previous risk model is a factor that might explain the correlation between the user's knowledge and their perceived risk.

## 5.6 General knowledge vs. risk perception

Figs. 5.3 and 5.4 suggest that there may be a correlation between the relative level of knowledge users have about sensors and the relative level of risk they perceive from them.

We confirm our observation of correlation using Spearman's rank-order correlation measure [57]. As it can be seen in Table 5.4, we present the Spearman's correlation between the comparative knowledge and the perceived risk about different sensors for different participants' dataset: group one before being presented with the sensor descriptions, group one after sensor description, group two after sensor descriptions, and finally groups one and two after being presented with the sensor descriptions.

For each participants' dataset, the sensors are separately ranked based on the level that the users are familiar with them, similar to Figure 5.3. Accordingly, the levels

| Participants' dataset | Status | Spearman's correlation |
|---|---|---|
| Group 1 | Before sensor desc. | 0.61 |
| Group 1 | After sensor desc. | 0.61 |
| Group 2 | After sensor desc. | 0.48 |
| Groups 1 and 2 | After sensor desc. | 0.58 |

Table 5.4: Spearman's correlation between the comparative knowledge and the perceived risk about different sensors

of concern are ranked too. The Spearman's correlation equation has been applied on these ranks for each group separately.

For example, the Spearman's correlation between the comparative knowledge (median: "I know what this is, but I don't know how this works", IQR[7]: "I've never heard of this" – "I know very well how this works") and the perceived risk about different sensors for group one (median: "Not concerned", IQR: "Not concerned" – "A little concerned") before knowing the sensor descriptions is $r = 0.61$ ($p < 0.05$).

As it can be seen, these results support that the more the users know about these sensors, the more concern they express about the risk of the sensors revealing PINs. We acknowledge that other methods of ranking the results, e.g. using median, produce slightly different final rankings. However, given the high confidence level of the above test, we expect the correlation to be supported if other methods of ranking are used.

Assuming that customer demand drives better security designs, the above correlation may explain why sensors that are newer to the market have not been considered as OS resources and consequently have not been subject to similar strict access control policies.

## 5.7 Perceived risk vs. the actual risk

We are specifically interested in the users' relative risk perception of sensors in revealing their PINs in comparison to the actual relative risk level of these sensors. As mentioned before, we only study the level of perceived risks users associate with their PINs being discovered by each sensor. Users might have different levels of concerns for different sensors if asked for other types of risks such as revealing touch actions, physical activities, and phone call timing to a third party and via an installed (game) app without their consent. As discussed earlier, we chose the risk to PINs since it is

[7]interquartile range

a clear and intuitive security risk, and we can put the perceived risk levels in context with respect to the actual risk levels for a number of sensors as presented in Section 4.8.5.

In Chapter 4, we listed the results reported in the literature in Table 4.18 for the following sensors: light, camera, microphone, gyroscope, motion, and orientation. Fig. 5.4 shows that users generally have expressed more concern about sensors such as camera and microphone than accelerometer, gyroscope, orientation, and motion. This does not match the actual risk levels since the latter sensors allow PIN recovery with higher accuracy as we have shown in Section 4.8.5. When asked after filling the questionnaire, most participants could not come up with realistic attack scenarios using camera and microphone. For microphone, some users thought they might say the PIN out loud. For camera, a few of our participants thought face recognition might be used to recover the PIN, hence they rated camera's risk to their PINs high. One user thought the camera might capture the reflection of the entered PIN in her glasses.

Among our participants, one mentioned but described doubt about motion, orientation, accelerometer, and gyroscope being able to record the shakes of the mobile phone while entering a PIN after they saw the sensor descriptions: "I feel those positional sensors might be able to reveal something about my activities, for example if I open my banking app or enter my PIN. But it is extremely hard for different users, and when working with different hands and positions". This participant expressed only "a little concern" about them, stating that: "..., and by little concern, I mean extremely little concern". One of our participants was completely familiar with these attacks and in fact had read some related papers. This user was "extremely concerned". Other users who rated these sensors risky in general, said they were generally concerned about different sensors. One commented: "I can not think of any particular situation in which these sensors can steal my PIN, but the hackers can do everything these days."

## 5.8 Possible solutions

In this section, we discuss the current academic and industrial countermeasures to mitigate sensor-based attacks.

### 5.8.1 Academic approach

Different solutions to address the in-app access attacks have been suggested in the literature: e.g. restricting the sensor to one app, reducing the sampling rate, temporal pause of the sensor on sensitive entries such as keyboard, rearranging keyboard for password entrance, asking for explicit permission from the user, ranking apps based on their similarities to malware, and obfuscating anomalies in sensor data [17, 21, 33, 84, 85, 88, 92, 99, 100, 115]. However, after many years of research on showing the serious security risks of sensors such as accelerometer and gyroscope, none of the major mobile platforms have revised their in-app access policy.

We believe that the risks of unmanaged sensors on mobile phones, specially through JavaScript code, are not known very well yet. More specifically, many OS-/app level solutions such as asking for permissions at the installation time, or malware detection approaches would not work in the context of a web attack. In our previous work [82], we suggested to apply the same security policies as those for camera, microphone, and GPS for the motion and orientation sensors. Our suggestion was to set a multi-layer access control system on the OS and browser levels. However, the usability and effectiveness of this solution are arguable. First, asking too many permissions from the user for different sensors might not be usable. Furthermore, for some basic use cases such as gesture recognition to clear a web form, or adjusting the screen from portrait to landscape, it might not make sense to ask for user permission for every website. Second, with the increase of the number of sensors accessible through mobile browsers, this approach might not be effective due to the classic problem of sidestepping the security procedure by users when it is too much of a burden [24]. As stated by one of our participants: "I don't mind these sensors being risky anyway. I don't even review the permission list. I have no other choice to be able to use the app". Moreover, as we have shown in Section 5.3, users generally do not understand the implications of these sensors on discovering their PINs for example, even though they know how these sensors work. Hence, such an approach might not be effective in practice.

### 5.8.2 Industrial approach

**Native apps.** As mentioned in Chapter 1, having access to newer sensors such as ambient light, accelerometer, and gyroscope is unrestricted via native apps. Android does not require developers to declare any permission for these sensors when using them in apps [6]. Accordingly, when users install an app using these sensors, they

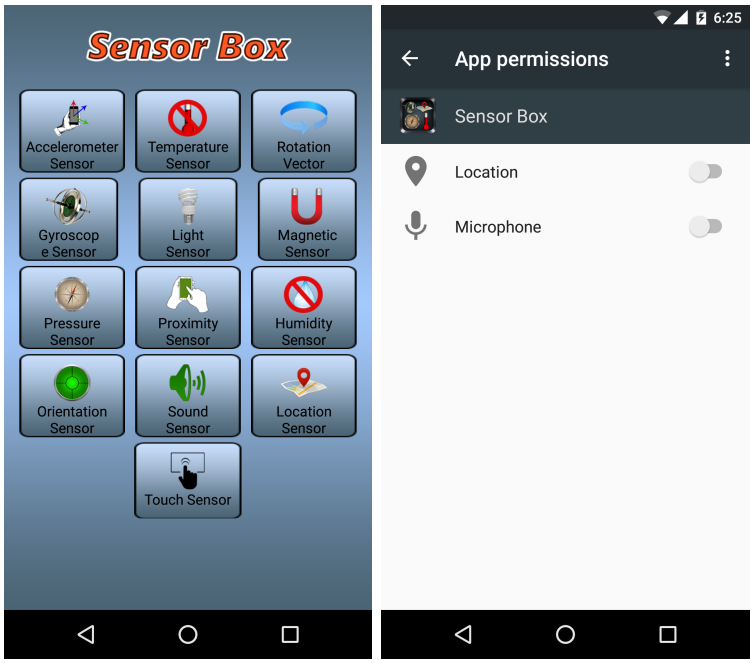


Figure 5.6: Sensor Box, an Android app and its permissions

are not required to grant any permissions, and are not even aware of the activation of these sensors. For example, Fig. 5.6, shows Sensor Box[8], an app installed from Google Play, which allows the users to visually test their mobile sensors. As it can be seen, apart from GPS and microphone, no permissions are required for other sensors. Depending on the app, these sensors can be active even when the app is running in the background.

**W3C Device Orientation Event Specification.** There is no Security and Privacy section in the latest official W3C Working Draft Document on Device Orientation Event [112]. However, at the time of writing this thesis, a new version of the W3C specification is being drafted, which includes a new section on security and privacy issues related to mobile sensors[9], as suggested by us in [82]. The authors working on the revision of the W3C specification point out the problem of fingerprinting mobile devices [21], and touch action recovery [81, 82] through these sensors, and suggest the following mitigations:

- "Do not fire events when the page where they were registered on is not visible or has been backgrounded."
- "Fire events only on the top-level browsing context or same-origin nested iframes."

[8] play.google.com/store/apps/details?id=imoblife.androidsensorbox
[9] w3c.github.io/deviceorientation/spec-source-orientation.html

- "Limit the frequency of events (typically 60 Hz seems to be sufficient)."

We believe that these measures may be too restrictive in blocking useful functionalities. For example, imagine a user consciously running a web program in the browser to monitor his daily physical activities such as walking and running. This program needs to continue to have access to the motion and orientation sensor data when the user is working on another tab or minimizes the browser. One might argue that such a program should be available as an app instead, hence the use case is not valid. However, it is expected that the boundary between installed apps and embedded JavaScript programs in the browser will gradually diminish [28].

**Mobile browsers.** As we showed in [82], browsers and mobile operating systems behave differently on providing access to sensors. Some allow access only on the active webpage and any embedded iframes (although with different origins), some allow access to other tabs, when browser is minimized, or even when the phone is locked. Hence, there is not a consistent approach across all browsers and mobile platforms. Reducing the frequency rate has been applied to all well-known browsers at the moment [82]. For instance, Chrome reduced the sensor readings from 200 Hz to 60 Hz due to security concerns[10]. However, our attack shows that security risks are still present even at lower frequencies. iOS and Android limit the maximum frequency rate of some sensors such as Gyroscope to 100 Hz and 200 Hz, respectively. It is expected that these frequencies will increase on mobile OSs in the near future and in-browser access is no exception. In fact, current mobile gyroscopes support much higher sampling frequencies, e.g. up to 800 Hz by STMicroelectronics (on Apple products), and up to 8000 Hz by InvenSense (on the Google Nexus range) [84]. With higher frequencies available, attacks such as ours can perform better in the future if adequate security countermeasures are not applied.

As discussed in previous chapter, following our report of the issue to Mozilla, starting from version 46 (released in April 2016), Firefox restricts JavaScript access to motion and orientation sensors to only top-level documents and same-origin iframes[11]. In the latest Apple Security Updates for iOS 9.3 (released in March 2016), Safari took a similar countermeasure by "suspending the availability of this [motion and orientation] data when the web view is hidden"[12]. However, we believe the implemented countermeasures should only serve as a temporary fix rather than the ultimate

[10] bugs.chromium.org/p/chromium/issues/detail?id=421691
[11] mozilla.org/en-US/security/advisories/mfsa2016-43/
[12] support.apple.com/en-gb/HT206166

solution. In particular, we are concerned that it has the drawback of prohibiting potentially useful web applications in the future. For example, a web page running a fitness program has a legitimate reason to access the motion sensors even when the web page view is hidden. However, this is no longer possible in the new versions of Firefox and Safari. Our concern is confirmed by members in the Google Chromium team[13], who also believe that the issue remains unresolved.

## 5.9 Discussions

As we explained in section 5.3.2, there exist around 25 different sensors on mobile platforms. They include communicational sensors such as WiFi, environmental sensors such as ambient light, movement sensors such as motion and orientation, and biometric sensors such as Fingerprint. Here we specifically discuss biometric sensors since they are highly related to the individuals' identity.

After decades of working on password, it seems that people still cannot remember strong passwords. Biometrics have been offered to users as an effective authentication mechanism. Examples include TouchID and Fingerprint sensors on iOS and Android devices respectively. But the biometric-based authentication is not limited to mobile devices only. For example, when paying with iPhone contactlessly, you need to rest your finger on TouchID and hold your iPhone in close proximity to the contactless reader until the task is finished. Furthermore, since many banks have already moved their services to mobile platforms, they benefit from the biometrics sensors available on mobile devices, say for implementing 2-factor authentication. As an example, in addition to user name and passwords, HSBC authenticates their customers through TouchID[14] and voice ID[15]. Another example is *Smile to Pay* facial recognition app[16] where deep learning is applied to overcome the difficulty of face authentication when the face photo is not in the normal form. Recently Yahoo has also introduced its ear-based smartphone identification system[17].

On the other hand, our findings show that mobile users are relatively concerned with identity-related or biometric sensors. However, we discussed that these sensors are not necessarily the most risky ones to PINs in practice. As we mentioned earlier, we believe that this might be the influence of a more general risk model that the users

[13] bugs.chromium.org/p/chromium/issues/detail?id=523320
[14] us.hsbc.com/1/2/home/personal-banking/pib/mobile/touchid
[15] hsbc.co.uk/1/2/contact-and-support/banking-made-easy/voice-id
[16] brandchannel.com/2015/03/16/alibaba-demos-smile-to-pay-facial-recognition-app/
[17] bbc.co.uk/news/technology-32498222

have on mobile technology. We believe that this is an important research topic and requires further studies.

## 5.10 Limitations

We consider this work a pilot study that explores user risk perception on a comprehensive list of mobile sensors. We envisage the following future work to address these limitations and expand this work:

- *More Participants*: We performed our user studies on a set of users who were recruited from a wide range of backgrounds. Yet the number of the participants is limited. A larger set of participants will improve the confidence in the results. With a large and diverse set of participants, we can also study the effect of demographic factors on perceived risk.

- *Other Risks*: We studied the perceived risk on PINs as a serious and immediate risk to users' security. The study can be expanded by studying users' risk perception on other issues such as attackers discovering phone call timing, physical activities, or shopping habits.

- *Other Types of Access*: When interviewing our participants, we presented them with a scenario involving a game app which is installed on their smartphone. This only covers the in-app access to sensors. However, people might express different risk levels for other types of access, e.g. in-browser access. This needs further investigation.

- *Issues with Training Users.* We decided to provide our participants with a short description of each sensor's functionality (details in Appendix D, part 3). Furthermore, the participants were given the chance to ask as many questions as they wanted to fully understand the functionality of each sensor. This might not be the most effective way to inform users about sensors since some descriptions might seem too technical (and hence not fully understandable) to some users. How to inform users in an effective way is a complex topic of research which can be explored in the future. In fact, the need of teaching users about mobile sensors has already been felt by Android. As reported, Android Pay prepares to show new users where to find NFC on their phones[18].

[18] androidpolice.com/2016/08/12/android-pay-v1-5-prepares-to-show-new-users-where-to-find-nfc-on-their-phones-and-might-be-experimenting-with-the-secure-element-again-apk-teardown/

## 5.11 Summary

In this chapter we showed that users do not generally perceive a high risk about such sensors being able to steal their PINs. Furthermore, we showed that people are not even generally knowledgeable about these sensors on mobile devices. Accordingly, we discussed the complexity of designing a usable and secure solution to prevent the proposed attacks. Hence, designing a general mechanism for secure and usable sensor data management remains a crucial open problem for future research.

Many of the suggested academic solutions either have not been applied by the industry as a practical solution, or have failed. Given the results in our user studies, designing a practical solution for this problem does not seem to be straightforward. A combination of different approaches might help researchers devise a usable and secure solution. Having control on granting access *before* opening a website and *during* working with it, in combination with a smart notification feature in the browser would probably achieve a balance between security and usability. Users should also have control on reviewing, updating and deleting these data, if stored by the website or shared with a third party *afterwards*. Solutions such as Taintroid [46], a tracking app for monitoring sources of sensitive data on a mobile which has been applied for GPS in [18] could be helpful. After all, it seems that an extensive study is required toward designing a permission framework which is usable and secure at the same time. Such research is a very important usable security and privacy topic to be explored further in the future.

# Chapter 6

# Conclusion

In this chapter, we summarise the research work presented in this thesis, and discuss open research problems in the field, motivating a number of ongoing research efforts.

## 6.1 Thesis summary

In this thesis, we have studied the use of mobile sensors in security applications, investigating different security and privacy angles of sensors: designing a secure app for contactless payment, attacking user's contactless payment privacy by a malicious app, attacking users sensitive information such as touch actions and PINs through JavaScript code, and inspecting users' understanding and concern about mobile sensors.

By studying each angle, we contributed to the field of mobile sensors in different ways. First, in Chapter 2, we showed that it is promising to apply mobile sensors for security purposes such as secure contactless payment. Next in Chapter 3, we proposed an attack on contactless payments by the use of mobile NFC, and demonstrated that different types of in-app based attacks are possible through mobile sensors. In Chapter 4, we described multiple side channel attacks to steal the user's private information via JavaScript code, in contrast to native apps which require installation. We implemented multiple attacks on a wide range of sensitive information about users including their 4-digit PINs. During this work we informed the mobile industry about this vulnerability, and consequently we contributed toward fixing this issue, as described in Chapter 4. And finally in Chapter 5, we designed and performed multiple user studies in order to study the human aspects of mobile sensors. Our aim was to find out to what extent are mobile users familiar with, and concerned about these sensors while working with their mobile devices on a daily basis. We interviewed multiple user groups from different backgrounds, and concluded that people are not generally aware of these sensors on mobile devices, and hence their perceived risks levels do not match the actual risks of these sensors.

As mentioned in several parts of this thesis, our research aim was tackling real-world problems. To accomplish this, we were continuously in contact with the industry to have a practical perspective on the topic. Through this valuable experience, we found it challenging to make real-world impact based on academic research. For example, when we reported the vulnerability that we found in Chapter 4 to W3C and mobile browser vendors, it took us several rounds of communications with different technical and non-technical team members via different channels for each vendor to

| Vendor | Connection means | Date of our first msg | Date of fix release | No. of msgs | No. of people in the loop |
|---|---|---|---|---|---|
| W3C | Mailing lists and Github | 10 Aug 2015 | 26 Feb 2016 | 20 | 15 |
| Google Chrome | Chromium bugs | 21 Aug 2015 | Never | 29 | 10 |
| Firefox | Bugzilla | 24 Aug 2015 | 26 Apr 2016 | 62 | 23 |
| Apple | Private emails | 26 Aug 2015 | 9 Mar 2016 | 10 | 3 |

Table 6.1: A summary on our communication with W3C and mobile browser vendors

make an actual impact (Table 6.1). We learned a few lessons during this process which we reflect them here. First, when reporting a bug which requires programming to fix it, provide some attack code as well as suggested solutions to the vendor. Second, provide free versions of your technical reports and papers to the community. Third, find the correct channel to contact each community and vendor. As you can see in Table 6.1, different standard organisations and mobile browser vendors have their own preferred way of communication. Finally, try to stay in the loop of the fix development and make contribution toward it. As you can see in the Table 6.1, this is a time-consuming process; e.g. fixing the reported issue in the case of W3C has taken over a year.

## 6.2 Future work

As future work, we would like to conduct an in-depth study of all sensors available on mobile devices. So far, only a few sensors including the ambient light [100], accelerometer and gyroscope [85, 115] have been shown to leak sensitive user information. In Chapter 5, and in [80–83], we have identified a list of 25 sensors available in modern Android/iOS devices. It remains to be investigated whether other sensors may also cause the information leakage. Furthermore, it is yet unclear how the information leakage relates to the sensor sampling rate. With better understanding of sensors, we wish to investigate if we can build new security mechanisms. On the other hand, we are interested in more user studies related to sensors. We believe that in order to manage these sensors on mobile platforms properly, we need to involve mobile users as co-designers of such a systems. Accordingly, this system needs to be smart enough to keep a balance between automatic decisions about sensor permissions and user prompts. Such a system would probably include an artificial intelligence (AI) unit to handle such a complex problem.

**Extended study on sensors.** We will extend the study on information leakage from a few selected sensors [80–83, 99, 100] to the full range. As an example, certain apps (e.g. games) tend to require intensive computation and hence cause the device temperature to rise. Hence, by measuring device's temperature and analysing the patterns of changes, it might be possible to infer what kind of apps are running on the device.

**Sensor rate correlation.** We will investigate the correlation between information leakage and sensor sampling rates. This is a topic that the industrial community is particularly interested in. The sampling rate for the in-app access to sensors is usually limited to 200 Hz, at which it has been shown that an app can reliably infer the user inputs on the touchscreen based on the measurements of motion sensors [115]. Cautioned by this attack, browser vendors reduce the in-browser sensor access rate by a factor of 3 to 5. But in [80–83], we demonstrate that, at this reduced rate, it is still possible to infer the user's 4-digital PIN with high accuracy. Hence, it will be worthwhile to investigate whether there exist *safe-zone* rates, where the information leakage is minimized (i.e., our attack in [80–83] stops working) while the normal use of the application (e.g. orientation of screen) is unaffected.

**Useful apps.** Besides studying the "weaknesses" of sensors, we also plan to investigate the "strengths" of sensors. The aim is to find ways to make good uses of sensors, instead of simply containing their bad use. This builds on our previous work [79] that utilizes accelerometers to prevent attacks in NFC payments. We will investigate other novel applications of sensors, e.g. to perform secure pairing of two devices based on their correlated sensor readings after physical tapping.

**Understanding mobile users.** We aim to develop an in-depth understanding of user perceptions of the risks and benefits associated with data generated by mobile sensors, and how they drive behaviour. Based on our previous work [80, 83], our hypothesis is that users are not aware of the data generated by many sensors, and how that data might be used to undermine their security and privacy. Our aim is to study perceptions and behaviour 'in context' (how the device is used in users' daily tasks), over a longer period of time, and the response to a number of interventions (different UI configurations, but also feedback, individual and group reflection exercises). The resulting insights will be used to develop designs that support users' security and privacy goals in the context of their daily tasks, and also to create a map of mental models of security and privacy risks associated with the data generated by mobile devices in general, and sensors in particular.

**AI-based sensor management system.** The specific design of an intelligent system will be informed by the understanding of users as well as the understanding of sensors as we mentioned earlier, but we anticipate an important role will be played by an artificial intelligence (AI) unit. The AI unit will be aware of the context of the application in which sensors are used and assess the risk of information leakage accordingly. We envisage that the unit works autonomously most of the time without requiring user intervention. But we also anticipate that the user studies may demonstrate that the user wants to have a sense of control in configuring the AI unit (e.g. reviewing collected data, logging earlier risk assessment decisions, and updating and deleting data). This builds on research of existing automated risk assessment techniques for mobile apps such as natural language processing (NLP) of the permissions and descriptions of the apps [93]. Another critical component of such a system is the user interface (UI), the design of which needs further user studies for an effective design.

# Appendix A

# TTP Usability Experiment

Participant Identification Number: ……

Name of Researcher: **Maryam Mehrnezhad**

| | |
|---|---|
| 1. I confirm that I have read and understand the study description for this study. I have had the opportunity to consider the information, ask questions and have had these answered satisfactorily. | ☐ |
| 2. I understand that my participation is voluntary and that I am free to withdraw at any time, without giving any reason. | ☐ |
| 3. I understand that any information given by me will be anonymized and may be used in future reports, articles or presentations by the research team. | ☐ |
| 4. I agree to take part in the above study. | ☐ |

| _____ | _____ | _____ |
|---|---|---|
| Name of Participant (optional) | Date | Signature |
| _____ | _____ | _____ |
| Researcher | Date | Signature |

**Study description:** *Contactless payment is a new PIN-less approach for payment using the Near Field Communication (NFC) technology. If you own an NFC-enabled card and wish to make a payment, you just need to hold the card in front of an NFC reader and wait for the confirmation. Using an NFC-enabled mobile device gives you the same functionality of a contactless card.*

*In this study, you will be asked to perform two experiments. In the first experiment, you will experience the conventional process of making a contactless payment using a mobile phone. In the second experiment, you will be asked to follow a new protocol called "Tap-Tap and Pay" to make an NFC payment. The "Tap-Tap and Pay" protocol is designed to make the NFC payment more secure, preventing man-in-the-middle attacks. However, we want to hear from you about your opinions on usability of the new protocol. At the end of the two experiments, we appreciate if you can help our research by filling in a questionnaire.*

Participant Identification Number: ……..

1. Have you ever paid by a contactless card? **Yes** ☐ **No** ☐

| | | **Strongly agree** | **Agree** | **Neutral** | **Disagree** | **Strongly disagree** |
|---|---|---|---|---|---|---|
| 2. | NFC payment in Experiment 1 is convenient | ☐ | ☐ | ☐ | ☐ | ☐ |
| 3. | NFC payment in Experiment 1 is fast | ☐ | ☐ | ☐ | ☐ | ☐ |
| 4. | I feel NFC payment in Experiment 1 is secure | ☐ | ☐ | ☐ | ☐ | ☐ |

5. Do you have any comments about the **usability** of making an NFC payment in Experiment 1?

6. Do you have any comments about the **security** of making an NFC payment in Experiment 1?

Participant Identification Number: ……..

| | | Strongly agree | Agree | Neutral | Disagree | Strongly disagree |
|---|---|---|---|---|---|---|
| 7. | NFC payment in Experiment 2 is convenient | ☐ | ☐ | ☐ | ☐ | ☐ |
| 8. | NFC payment in Experiment 2 is fast | ☐ | ☐ | ☐ | ☐ | ☐ |
| 9. | I feel NFC payment in Experiment 2 is secure | ☐ | ☐ | ☐ | ☐ | ☐ |

10. Do you have any comments about the **usability** of making an NFC payment in Experiment 2?

11. Do you have any comments about the security of making an NFC payment in Experiment 2?

**CONTACT:** Thank you for your participation in this study. If you have further questions about the study, please contact Maryam Mehrnezhad (m.mehrnezhad@ncl.ac.uk). In addition, if you have any concerns about any aspect of the study, you may contact Dr. Feng Hao, Room 1106, Claremont Tower, School of Computing Science, Newcastle University, Email: feng.hao@ncl.ac.uk , Telephone: +44 191 208 6384 ; Fax: +44 191 208 8788

# Appendix B

# Help Document for Sensor Data Collection Process

# *Reading sensor data for 4-digit PINs using JavaScript*

*Author: Maryam Mehrnezhad (m.mehrnezhad@ncl.ac.uk), Apr 2017*

In this help file, we describe the details of our JavaScript code used for reading sensor data (motion and orientation) for 4-digit PINs in a project conducted in Newcastle University, UK. The outcome of this project is published in [1-4]. Our JavaScript code is publicly available on *Github* via this link: https://github.com/maryammjd/Reading-sensor-data-for-fifty-4digit-PINs. This code asks the user to enter fifty 4-digit PINs, each 5 times, and saves the PINs along with their sensor measurements (motion and orientation) in an *m-lab* database. A sample dataset for 10 users is also publicly available via the project's *Github* page. In case of any further questions, please contact the authors.

## JavaScript code (client, server, and db)

We setup an account in *mlab.com* and created a deployment (database) named *sensordata*. In this deployment, we created a collection named *sensor*. This collection is in charge of saving json (JavaScript Object Notation) data received by the server as documents. We defined our json structure in our JavaScript code in *Node.js* to include three elements: *type* (status, or sensor type, or time), *data* (value), *ts* (time value). Note that the time in the *type* element is when the data is read on the mobile device, versus the *ts* element is when each record is inserted in the database.

In our JavaScript code (*app.js*), we connect to *mongoDB* and handle the sensor data via the *socket.io* API. All user interactions (beginning PIN entry, entering PINs, and finishing), alongside with the sensor measurements (motion and orientation), are sent to the database by the server. We run the server on a local computer through *node.js* cmd. Once the index page is opened on the phone, the data collection starts.

In our *index.html* file in the client side, we call the *numPad.js* script which presents the users with a GUI where fifty 4-digit PINs are shown (each repeated 5 times). The user needs to enter them in a textbox as shown in Fig. 1. As it can be seen, the number of PINs entered (out of 50) and the number of counts (out of 5) are also shown to users. On each digit entry, our JavaScript code sends a new record (Key Down Key Up) to our database using the *onkeydown* event. Our *numPad.js* file includes two event listeners on the window object which fire on device motion and device orientation DOM events (called *devicemotion* and *deviceorientation*). We have hard-coded the fifty 4-digit PINs in this file. These semi-random PINs are created by using a Matlab code.

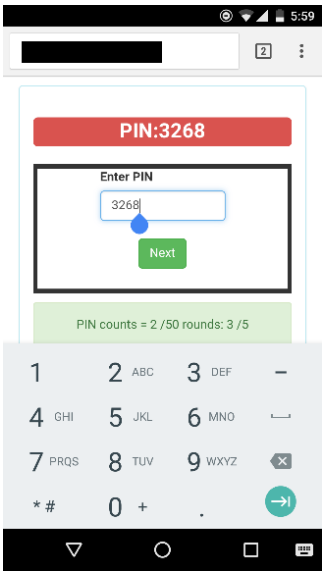


Figure 1: GUI for PIN entry

This data is arrived and inserted to our *MongoDB* database as shown in Fig 2.

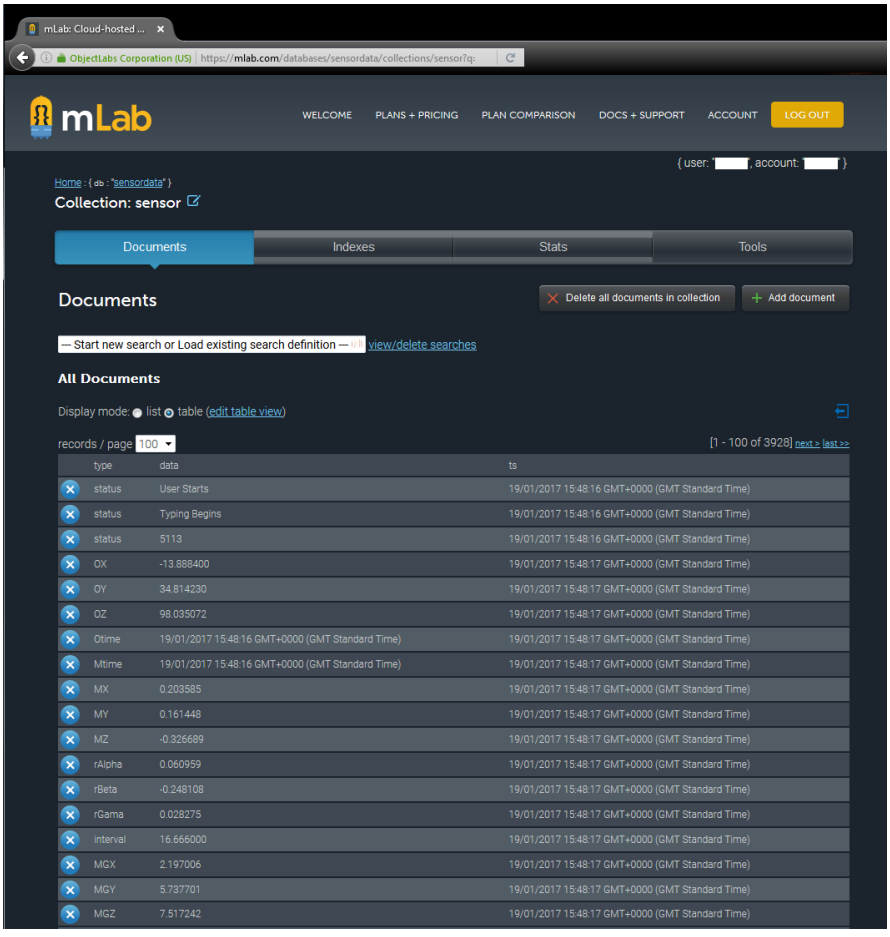


Figure 2: M-lab database

As it can be seen, the *type* element can include either the status of the data, the type of the data, or the time that it has been collected from the mobile device. The order of the values

for a sample data collection for fifty 4-digit PINs (each PIN 5 times) from a user is saved as presented in the bellow set:

| |
|---|
| {User Starts, |
| {{Typing Begins, |
| 5113 (First shown PIN), |
| a series of Orientation and Motion Data, |
| Key Down, Key Up (when the first digit is clicked), |
| a series of Orientation and Motion Data for the first digit, |
| Key Down, Key Up (when the second digit is clicked), |
| a series of Orientation and Motion Data for the second digit, |
| … (the same for the third and fourth digits), |
| Key Down, Key Up (to show the end of the 4-digit PIN entry), |
| 5113 (First typed PIN which could be different from the shown PIN due to user error), |
| Typing Ends}, |
| … (the previous process for the first PIN for another 4 times)}, |
| … (the previous process for another 49 PINs), |
| User Finishes}. |

## Data Exportation

After we collected data for each user, we exported the data to an *Excel* file on a local computer for further processing in *Matlab*. Next, we deleted all the documents in the related collection in *mlab.com* for the next user data collection. We used the following command through *MongoDb* cmd for exportation (the username and password are set on the time of creating the *sensordata* development):

```
mongoexport -h ds033818.mlab.com:33818 -d sensordata -c sensor -u <username> -p <password> -o sensor.csv --csv -f "type","data","ts"
```

Since the browser leverage a wrapper API to provide the motion and orientation sensor readings through JavaScript, similar to the Android sensor manager API, the same reading to native apps is provided here (except for the sampling rate). This means that these sensor readings are provided *onSensorChanged* event (with lower frequency). While analysing our measurements, we noticed that the resolutions of the orientation data and the motion data are different. Due to this technical issue and for simplicity while working with this data in *Matlab*, when converting data from Excel files to text files, we created two different text files (User<no.>Motion and User<no.>Orientation) for motion and orientation separately. We repeated the same process for each user using the *Sort & Filter* feature in *Excel* as shown in the Fig. 3. As can be seen, we only include the records that we need and we exclude the unnecessary ones (e.g. interval and times).

When the text files were created for all users, we imported them to *Matlab* and performed further analysis on them as explained in our papers [1-4].

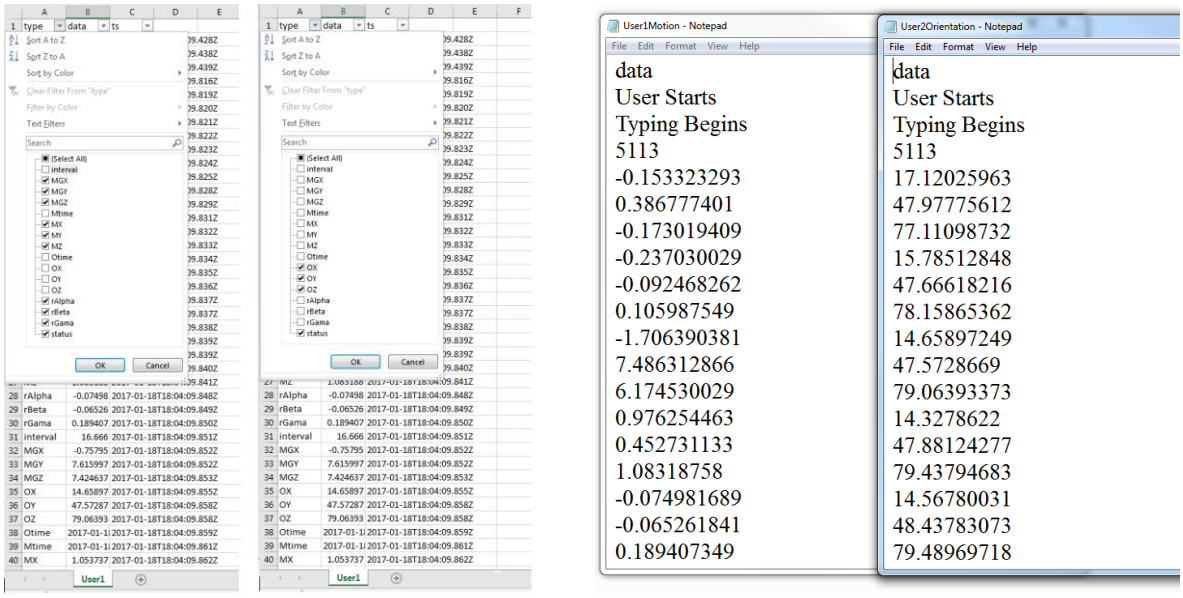


Figure 3: Converting excel files to text files

## References

[1] M. Mehrnezhad, E. Toreini, S. Shahandashti, and F. Hao, "TouchSignatures: Identification of User Touch Actions based on Mobile Sensors via JavaScript", In the Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS 2015, Singapore, Apr 14-17, 2015, ACM, P 673-673.

[2] M. Mehrnezhad, E. Toreini, S. Shahandashti, and F. Hao, "TouchSignatures: Identification of User Touch Actions and PINs based on Mobile Sensors via JavaScript", Journal of Information Security and Applications, Elsevier, V 26, Feb 2016, P 23-38.

[3] M. Mehrnezhad, E. Toreini, S. Shahandashti, F Hao, "Stealing PINs via Mobile Sensors: Actual Risk versus User Perception", The 1st European Workshop on Usable Security, EuroUSEC 2016, Workshop at the Privacy Enhancing Technologies Symposium (PETS 2016), Jul 18, 2016, Germany.

[4] M. Mehrnezhad, E. Toreini, S. Shahandashti, F Hao, "Stealing PINs via Mobile Sensors: Actual Risk versus User Perception", International Journal of Information Security, Springer, April 2017, Pages 1-23.

# Appendix C

# Touch Action Study Guide

Dear volunteer,

Thanks for your participation in this experiment. This study will ask you to perform some quick tasks working with mobile phones. The collected data will be anonymously used for research purposes only. You will receive a £10 Amazon voucher as a thank you for your contribution to this research.

This experiment has 2 sections:

- Sections 1: working with a webpage opened by **Chrome** on an **iPhone 5** using **only one hand** for holding the phone and the same hand for touching the screen while sitting on a chair.
- Sections 2: working with a webpage opened by **Chrome** on an **iPhone 5** using **two hands**; one for holding the phone and the second for touching the phone while sitting on a chair.

You will perform a simple touch action in each step and repeat each action 5 times. After that, please wait for 3 seconds until you start the next touch action. Meanwhile, you will be notified of the progress via the information box and alert box. The touch actions are as blow:

- Touch Action 1: **One time click** on any part of the box that you like
- Touch Action 3: **Scroll down only once**
- Touch Action 4: **Scroll up only once**
- Touch Action 5: **Scroll right only once**
- Touch Action 6: **Scroll left only once**
- Touch Action 7: **Zoom in only once**– for this action you can use two hands even in the *one-hand mode.*
- Touch Action 8: **Zoom out only once**– for this action you can use two hands even in the *one-hand mode*
- Touch action 9: **Hold** a word in order to copy it and when it is chosen release it

A trial setup is provided for you to practice as many times as you wish before performing the real tests. Please remain seated while doing this experiment.

| Please leave any extra comments here…. |
| --- |
| |

# Appendix D

# Interview Description of Mobile Sensors User Study

Hi. Thanks very much for contributing to our study. In this interview, we will ask you to fill in a few questionnaires about mobile sensors such as GPS, camera, light, motion and orientation. You are encouraged to think out loud as you go through, and please feel free to provide any comments during the interview. There is no right or wrong answer, and our purpose is to evaluate the mobile sensors, not you. Everything about this interview is anonymous. Please provide some information about yourself in Table D.1.

| | |
|---|---|
| Age | |
| Gender | |
| Profession/ background (optional) | |
| 1st language (optional) | |
| Mobile device | |
| Duration of owning a smartphone/tablet | |

Table D.1: Demography table

## Part One

A list of multiple mobile sensors is presented below. To what extent do you know each sensor on a mobile device? Please rate them in the table (Table D.2 was used).

## Part Two

Imagine that you own a smartphone which is equipped with all these sensors. Consider this scenario: you have opened a game app which can have access to all mobile sensors. You leave the game app open in the background, and open your banking app which requires you to enter your PIN.

Do you think any of these sensors can help the game app discover your entered PIN? To what extent are you concerned about each sensor's risk to your PIN? Please rate them in the table (Table D.3 was used). In this section, please only rely on the knowledge you already have about the sensors, and if you do not know some of them, describe your feeling of security about them.

## Part Three

Let us explain each sensor here:

- GPS: identifies the real-world geographic location.

| Sensor | I've never heard of this | I've heard of this but I don't know what this is | I know what this is but I don't know how this works | I know generally how this works | I know very well how this works |
|---|---|---|---|---|---|
| Bluetooth | | | | | |
| Gyroscope | | | | | |
| GPS | | | | | |
| Sensor Hub | | | | | |
| Ambient Temperature | | | | | |
| Accelerometer | | | | | |
| Magnetic Field | | | | | |
| Motion | | | | | |
| Fingerprint | | | | | |
| Orientation | | | | | |
| Proximity | | | | | |
| Ambient Pressure | | | | | |
| Hall Sensor | | | | | |
| Rotation | | | | | |
| Touch Screen | | | | | |
| Camera | | | | | |
| TouchID | | | | | |
| Barometer | | | | | |
| Gravity | | | | | |
| Microphone | | | | | |
| Ambient Humidity | | | | | |
| WiFi | | | | | |
| Ambient Light | | | | | |
| NFC | | | | | |
| Device Temperature | | | | | |

Table D.2: Sensor familiarity form used for part one

- Camera, Microphone: capture pictures/videos and voice, respectively.
- Fingerprint, TouchID: scans the fingerprint.
- Touch Screen: enables the user to interact directly with the display by physically touching it.

| Risk to PIN | | | | | |
|---|---|---|---|---|---|
| Sensor | Not Concerned | A little Concerned | Moderately Concerned | Concerned | Extremely Concerned |
| Bluetooth | | | | | |
| Gyroscope | | | | | |
| GPS | | | | | |
| Sensor Hub | | | | | |
| Ambient Temperature | | | | | |
| Accelerometer | | | | | |
| Magnetic Field | | | | | |
| Motion | | | | | |
| Fingerprint | | | | | |
| Orientation | | | | | |
| Proximity | | | | | |
| Ambient Pressure | | | | | |
| Hall Sensor | | | | | |
| Rotation | | | | | |
| Touch Screen | | | | | |
| Camera | | | | | |
| TouchID | | | | | |
| Barometer | | | | | |
| Gravity | | | | | |
| Microphone | | | | | |
| Ambient Humidity | | | | | |
| WiFi | | | | | |
| Ambient Light | | | | | |
| NFC | | | | | |
| Device Temperature | | | | | |

Table D.3: Sensor concern form used for parts two and three

- WiFi: is a wireless technology that allows the device to connect to a network.
- Bluetooth: is a wireless technology for exchanging data over short distances.
- NFC (Near Filed Communication): is a wireless technology for exchanging data over shorter distances (less than 10 cm) for purposes such as contacless payment.
- Proximity: measures the distance of objects from the touch screen.

- Ambient Light: measures the light level in the environment of the device.
- Ambient Pressure (Barometer), Ambient Humidity, and Ambient Temperature: measure the air pressure, humidity, and temperature in the environment of the device, respectively.
- Device Temperature: measures the temperature of the device.
- Gravity: measures the force of gravity.
- Magnetic Field: reports the ambient magnetic field intensity around the device.
- Hall sensor: produces voltage based on the magnetic field.
- Accelerometer: measures the acceleration of the device movement or vibration.
- Rotation: reports how much and in what direction the device is rotated.
- Gyroscope: estimates the rotation rate of the device.
- Motion: measures the acceleration and the rotation of the device.
- Orientation: reports the physical angle that the device is held in.
- Sensor Hub: is an activity recognition sensor and its purpose is to monitor the device's movement.

Please feel free to ask us about any of these sensors for more information.

Now that you have more knowledge about the sensors, let us describe the same scenario here again. Imagine that you own a smartphone which is equipped with all these sensors. You have opened a game app which can have access to all mobile sensors. You leave the game app open in the background, and open your banking app which requires you to enter your PIN.

Do you think any of these sensors can help the game app to discover your entered PIN? To what extent are you concerned about each sensor's risk to your PIN? Please rate them in the table (Table D.3 was used). In this part, please make sure that you know the functionality of all the sensors. If you are unsure, please have another look at the descriptions, or ask us about them.

Thanks very much for taking part in this study. Please leave any extra comment here.

An Amazon voucher and a business card are in this envelope. Please contact us if you have any questions about this interview, or are interested in the results of this study.

# Bibliography


[1] Apple & Samsung drive NFC mobile payment users to nearly 150m globally this year. Market leading report by Juniper Research, March 2014. Available online at http://www.juniperresearch.com/press/press-releases/apple-samsung-drive-nfc-mobile-payment-users.

[2] Mobile payment strategies: Remote, contactless & money transfer 2014–2018. Market leading report by Juniper Research, July 2014. Available online at http://www.juniperresearch.com/reports.php?id=726.

[3] Ahmed Al-Haiqi, Mahamod Ismail, and Rosdiadee Nordin. On the best sensor for keystrokes inference attack on android. *Procedia Technology*, 11(0):989–995, 2013.

[4] Host-based card emulation. Available online at http://developer.android.com/guide/topics/connectivity/nfc/hce.html.

[5] Sensormanager. Available online at https://developer.android.com/reference/android/hardware/SensorManager.html.

[6] Android sensors. Available at http://developer.android.com/guide/topics/sensors/sensors_overview.html.

[7] International organization for standardization, BS ISO/IEC 14443-1:2008+A1:2012 identification cards. contactless integrated circuit cards. proximity cards. physical characteristics, 2012. Available at www.bsol.bsigroup.com.

[8] International organization for standardization, BS ISO/IEC 14443-2:2010+A2:2012 identification cards. contactless integrated circuit cards. proximity cards. radio frequency power and signal interface, 2012. Available at www.bsol.bsigroup.com.

[9] International organization for standardization, BS ISO/IEC 14443-3:2011+A6:2014 identification cards. contactless integrated circuit cards. proximity cards. initialization and anticollision, 2014. Available at www.bsol.bsigroup.com.

[10] International organization for standardization, BS ISO/IEC 14443-4:2008+A4:2014 identification cards. contactless integrated circuit cards. proximity cards. transmission protocol, 2014. Available at www.bsol.bsigroup.com.

[11] Contactless Specifications for Payment Systems, Book A: Architecture and General Requirements, 2015. Available at www.emvco.com /specifications.aspx?id=21.

[12] EMV Contactless Specifications for Payment Systems, Book B: Entry Point, 2015. Available at www.emvco.com/specifications.aspx?id=21.

[13] EMV Contactless Specifications for Payment Systems, Book C2: Kernel 2 Specification, 2015. Available at www.emvco.com/specifications.aspx?id=21.

[14] EMV Contactless Specifications for Payment Systems, Book C3: Kernel 3 Specification, 2015. Available at www.emvco.com/specifications.aspx?id=21.

[15] EMV Contactless Specifications for Payment Systems, Book D: Contactless Communication Protocol, 2015. Available at www.emvco.com /specifications.aspx?id=21.

[16] International organization for standardization, BS ISO/IEC 7816-4:2013, identification cards. integrated circuit cards. organization, security and commands for interchange, 2013. Available at www.bsol.bsigroup.com.

[17] Adam J Aviv, Benjamin Sapp, Matt Blaze, and Jonathan M Smith. Practicality of accelerometer side channels on smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012.

[18] Rebecca Balebako, Jaeyeon Jung, Wei Lu, Lorrie Faith Cranor, and Carolyn Nguyen. Little brothers watching you: Raising awareness of data leaks on smartphones. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, SOUPS 2013. ACM, 2013.

[19] Daniel Bichler, Guido Stromberg, Mario Huemer, and Manuel Low. Key generation based on acceleration data of shaking processes. In *Ubiquitous Computing*, UbiComp 2007. Springer Berlin Heidelberg, 2007.

[20] Cheng Bo, Lan Zhang, Xiang-Yang Li, Qiuyuan Huang, and Yu Wang. Silentsense: Silent user identification via touch and movement behavioral biometrics. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking*, MobiCom 2013. ACM, 2013.

[21] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. Mobile device identification via sensor fingerprinting. 2014. Technical report available athttp://arxiv.org/abs/1408.1416.

[22] Joseph Bonneau, Soren Preibusch, and Ross Anderson. A birthday present every eleven wallets? the security of customer-chosen banking PINs. In *Proceedings of 16th Financial Cryptography and Data Security*, FC 2012. Springer Berlin Heidelberg, 2012.

[23] Stefan Brands and David Chaum. Distance-bounding protocols. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer Berlin Heidelberg, 1994.

[24] Cristian Bravo-Lillo, Saranga Komanduri, Lorrie Faith Cranor, Robert W. Reeder, Manya Sleeper, Julie Downs, and Stuart Schechter. Your attention please: Designing security-decision uis to make genuine risks harder to ignore. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, SOUPS 2013. ACM, 2013.

[25] E Oran Brigham, E Oran Brigham, JulioJ Rey Pastor, Rey Pastor, Tom M Tom M Apostol, MargaritaMartínez Rodríguez, Miguel RamónMargarita Rodríguez, Miguel Ramón Martínez, C HenryPENNEY Edwards, DAVID EC Henry Edwards, et al. *The fast Fourier transform and its applications*. Number 517.443. Prentice Hall, 1988.

[26] Liang Cai and Hao Chen. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX conference on Hot topics in security*, HotSec 2011. ACM, 2011.

[27] Liang Cai and Hao Chen. On the practicality of motion based keystroke inference attack. In *Proceedings of the 5th international conference on Trust and Trustworthy Computing*, TRUST 2012. Springer Berlin Heidelberg, 2012.

[28] Andre Charland and Brian Leroux. Mobile application development: Web vs. native. *Communications of the ACM*, 54(5):49–53, 2011.

[29] Ming Ki Chong and Hans Gellersen. How users associate wireless devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI 2011. ACM, 2011.

[30] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[31] Marianne Curphey. Card clash, what is it, and how to avoid ir, 2014. Available online at http://uk.creditcards.com/credit-card-news/what-is-card-clash-and-how-to-avoid-it-1372.php.

[32] 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 *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008.

[33] Anupam Das, Nikita Borisov, and Matthew Caesar. Exploring ways to mitigate sensor-based smartphone fingerprinting. 2015. Technical report available at http://arxiv.org/abs/1503.01874.

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

[35] Alexander De Luca, Alina Hang, Emanuel von Zezschwitz, and Heinrich Hussmann. I feel like i'm taking selfies all day!: Towards understanding biometric authentication on smartphones. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI 2015. ACM, 2015.

[36] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5):352–359, 2002.

[37] Saar Drimer and Steven J. Murdoch. Keep your enemies close: Distance bounding against smartcard relay attacks. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, SS 2007. USENIX Association, 2007.

[38] Planet of the phones. From the print edition by The Economist, 2015. Available online at http://www.economist.com/news/leaders/21645180-smartphone-ubiquitous-addictive-and-transformative-planet-phones.

[39] Nicole Eling, Siegfried Rasthofer, Eric Bodden, and Peter Buxmann. Investigating users' reaction to fine-grained data requests: A market experiment. In *Proceedings of Hawaii International Conference on System Sciences*, HICSS 2016. IEEE Press, 2016.

[40] Martin Emms, Budi Arief, Nicholas Little, and Aad van Moorsel. Risks of offline verify pin on contactless cards. In *Proceedings of 17th Financial Cryptography and Data Security*, FC 2013. Springer Berlin Heidelberg, 2013.

[41] Martin Emms and Aad van Moorsel. Practical attack on contactless payment cards. In *HCI2011 Workshop-Heath, Wealth and Identity Theft*, 2011.

[42] EMV Integrated Circuit Card Specifications for Payment Systems, Book 3, 2011. Available at www.emvco.com/specifications.aspx?id=223.

[43] Book 2 - Security and Key Management, 2011. Available at www.emvco.com /specifications.aspx?id=223.

[44] EMV Acquirer and Terminal Security Guidelines, 2014. Available at www.emvco.com/specifications.aspx?id=71.

[45] EMV Issuer and Application Security Guidelines, 2014. Available at www.emvco.com/specifications.aspx?id=71.

[46] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. *Transactions on Computer Systems.*, June 2014.

[47] Adrienne Porter Felt, Serge Egelman, and David Wagner. I've got 99 problems, but vibration ain't one: A survey of smartphone users' concerns. In *Proceedings*

*of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM 2012. ACM, 2012.

[48] 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*, SOUPS 2012. ACM, 2012.

[49] Lishoy Francis, Gerhard P Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical relay attack on contactless transactions by using NFC mobile phones. 2011. Technical report available at IACR Cryptology ePrint Archive.

[50] Gartner says emerging markets drove worldwide smartphone sales to 15.5 percent growth in third quarter of 2015, 2015.

[51] W3C Editor's Draft on Generic Sensor API. Available at w3c.github.io/sensors/.

[52] Location and Sensors APIs. Available at: developer.android.com/guide/topics/sensors/index.html.

[53] 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. Springer, 2012.

[54] Tzipora Halevi, Di Ma, Nitesh Saxena, and Tuo Xiang. Secure proximity detection for NFC devices based on ambient sensor data. In *17th European Symposium on Research in Computer Security*, ESORICS 2012. Springer Berlin Heidelberg, 2012.

[55] Chan Choong Wah Hao Feng. Private key generation from on-line handwritten signatures. *Information Management & Computer Security*, 10:159–164, 2002.

[56] Marian Harbach, Emanuel von Zezschwitz, Andreas Fichtner, Alexander De Luca, and Matthew Smith. It's a hard lock life: A field study of smartphone (un)locking behavior and risk perception. In *Symposium On Usable Privacy and Security*, SOUPS 2014. USENIX Association, 2014.

[57] Jan Hauke and Tomasz Kossowski. Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *QUAESTIONES GEOGRAPHICAE*, 30(2):87–93, 2011.

[58] Arik Hesseldahl. Apple iPhone 4 parts cost about $188. *Bloomberg Business*, June 2010. Available at www.bloomberg.com/bw/technology/content/jun2010/tc20100627_763714.htm.

[59] Ken Hinckley. Synchronous gestures for multiple persons and computers. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST 2003. ACM, 2003.

[60] Iulia Ion, Marc Langheinrich, Ponnurangam Kumaraguru, and Srdjan Capkun. Influence of user perception, security needs, and social factors on device pairing method choices. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, SOUPS 2010. ACM, 2010.

[61] Xing Jin, Xunchao Hu, Kailiang Ying, Wenliang Du, Heng Yin, and Gautam Nagesh Peri. Code injection attacks on html5-based mobile apps: Characterization, detection and mitigation. In *Proccedings of 21th ACM Conference on Computer and Communications Security*, CCS 2014. ACM, 2014.

[62] Ruogu Kang, Laura Dabbish, Nathaniel Fruchter, and Sara Kiesler. My data just goes everywhere: User mental models of the internet and implications for privacy and security. In *Eleventh Symposium On Usable Privacy and Security*, SOUPS 2015. USENIX Association, 2015.

[63] Eamonn J Keogh and Michael J Pazzani. Derivative dynamic time warping. In *The 1st SIAM International Conference on Data Mining*, SDM 2001. SIAM, 2001.

[64] Hassan Khan, Urs Hengartner, and Daniel Vogel. Usability and security perceptions of implicit authentication: Convenient, secure, sometimes annoying. In *Eleventh Symposium On Usable Privacy and Security*, SOUPS 2015. USENIX Association, 2015.

[65] D. Kirovski, M. Sinclair, and D. Wilson. The martini synch: Device pairing via joint quantization. In *IEEE International Symposium on Information Theory*, ISIT 2007. IEEE, 2007.

[66] Darko Kirovski, Mike Sinclair, and David Wilson. The martini synch. Technical Report MSR-TR-2007-123, Microsoft Research, September 2007.

[67] Alfred Kobsa, Rahim Sonawalla, Gene Tsudik, Ersin Uzun, and Yang Wang. Serial hook-ups: A comparative usability study of secure device pairing methods. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, SOUPS 2009. ACM, 2009.

[68] Adam Lella and Andrew Lipsman. The u.s. mobile app report, 2014. Available online at http://www.comscore.com/Insights/Presentations-and-Whitepapers/2014/The-US-Mobile-App-Report/.

[69] 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 2013. ACM, 2013.

[70] Mengyuan Li, Yan Meng, Junyi Liu, Haojin Zhu, Xiaohui Liang, Yao Liu, and Na Ruan. When csi meets public wifi: Inferring your mobile phone password via wifi signals. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS'16, pages 1068–1079, New York, NY, USA, 2016. ACM.

[71] David Lieb. All good things..., 2014. Available at blog.bu.mp/post/71781606704/all-good-things.

[72] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[73] Di Ma, Nitesh Saxena, Tuo Xiang, and Yan Zhu. Location-aware and safer cards: Enhancing RFID security and privacy via location sensing. *Dependable and Secure Computing, IEEE Transactions on*, 10(2):57–69, 2013.

[74] Geoff Marshall. Travel using contactless cards: An update from tfl, 2014. Available online at http://londonist.com/2014/07/travel-using-contactless-cards-an-update-from-tfl.

[75] Rene Mayrhofer. The candidate key protocol for generating secret shared keys from similar sensor data streams. In *Security and Privacy in Ad-hoc and Sensor Networks*. Springer Berlin Heidelberg, 2007.

[76] Rene Mayrhofer and Hans Gellersen. Shake well before use: Authentication based on accelerometer data. In *Pervasive Computing*. Springer Berlin Heidelberg, 2007.

[77] Rene Mayrhofer and Hans Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *Mobile Computing, IEEE Transactions on*, 8(6):792–806, 2009.

[78] Maryam Mehrnezhad, Mohammed Ali, Feng Hao, and Aad van Moorsel. Nfc payment spy: Privacy attacks on contactless payments using NFC-enabled mobile. In *Proccedings of Third International Conference on Research in Security Standardisation*, SSR 2016. Springer International Publishing, 2016.

[79] Maryam Mehrnezhad, Feng Hao, and Siamak Shahandashti. Tap-Tap and Pay (TTP): Preventing the mafia attack in NFC payment. In *Proccedings of Second International Conference on Research in Security Standardisation*, SSR 2015. Springer International Publishing, 2015.

[80] Maryam Mehrnezhad, Ehsan Toreini, and Feng Shahandashti, Siamakand Hao. Stealing pins via mobile sensors: Actual risk versus user perception. In *The 1st European Workshop on Usable Security*, EuroUSEC 2016, 2016.

[81] Maryam Mehrnezhad, Ehsan Toreini, Siamak Shahandashti, and Feng Hao. Touchsignatures: Identification of user touch actions based on mobile sensors via javascript. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS 2015. ACM, 2015.

[82] Maryam Mehrnezhad, Ehsan Toreini, Siamak Shahandashti, and Feng Hao. Touchsignatures: Identification of user touch actions and PINs based on mobile sensor data via javascript. *Journal of Information Security and Applications*, 26:23–38, 2016.

[83] Maryam Mehrnezhad, Ehsan Toreini, Siamak F. Shahandashti, and Feng Hao. Stealing pins via mobile sensors: actual risk versus user perception. *International Journal of Information Security*, pages 1–23, 2017.

[84] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *Proceedings of the 23rd USENIX conference on Security Symposium*, SEC 2014. ACM, 2014.

[85] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: your finger taps have fingerprints. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012.

[86] Katie Morley. Contactless cards: how to avoid paying twice, 2014. Available online at http://www.telegraph.co.uk/finance/personalfinance/money-saving-tips/11215583/Contactless-cards-how-to-avoid-paying-twice.html.

[87] Martin Fodslette Muller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525 – 533, 1993.

[88] Sashank Narain, Amirali Sanatinia, and Guevara Noubir. Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec 2014. ACM, 2014.

[89] ISO 14443, ISO 18092, Type-A, Type-B, Type-F, Felica, Calypso NFCIP, NFC-HELP!, 2009. Available online at http://www.nfc.cc/2009/01/03/iso-14443-iso-18092-type-a-type-b-type-f-felica-calypso-nfcip-nfc-help/.

[90] Smartphones: So many apps, so much time, 2015. Available online at http://www.nielsen.com/us/en/insights/news/2014/smartphones-so-many-apps-so-much-time.html.

[91] AN10927, MIFARE and handling of UIDs, 2013. Published by NXP.

[92] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: password inference using accelerometers on smartphones. In *Proceedings of the 12th Workshop on Mobile Computing Systems & Applications*. ACM, 2012.

[93] Rahul Pandita, Xusheng Xiao, Wei Yang, William Enck, and Tao Xie. Whyper: Towards automating risk assessment of mobile applications. In *Proceedings of the 22Nd USENIX Conference on Security*, SEC 2013. USENIX Association, 2013.

[94] Andrew Raij, Animikh Ghosh, Santosh Kumar, and Mani Srivastava. Privacy risks emerging from the adoption of innocuous wearable sensors in the mobile environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI 2011. ACM, 2011.

[95] Oriana Riva, Chuan Qin, Karin Strauss, and Dimitrios Lymberopoulos. Progressive authentication: deciding when to authenticate on mobile phones. In *In Proceedings of 21st USENIX Security Symposium*. ACM, 2012.

[96] Heather Saul. Oyster card users pay up to £91 more each week than people using new contactless payment, 2014. Available online at http://www.independent.co.uk/news/uk/home-news/oyster-card-users-pay-up-to-91-more-each-week-than-people-using-new-contactless-payment-9843642.htmll.

[97] Nitesh Saxena and Jonathan Voris. Still and silent: Motion detection for enhanced rfid security and privacy without changing the usage model. In *In Proccedings of 6th International Workshop Radio Frequency Identification: Security and Privacy Issues*, RFIDSec 2010. Springer Berlin Heidelberg, 2010.

[98] Babins Shrestha, Nitesh Saxena, Hien Thi Thu Truong, and N. Asokan. Drone to the rescue: Relay-resilient authentication using ambient multi-sensing. In *Financial Cryptography and Data Security: 18th International Conference*, FC 2014. Springer Berlin Heidelberg, 2014.

[99] Laurent Simon and Ross Anderson. PIN Skimmer: Inferring PINs through the camera and microphone. In *Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones Mobile Devices*, SPSM 2013, pages 67–78. ACM, 2013.

[100] Raphael Spreitzer. Pin skimming: Exploiting the ambient-light sensor in mobile devices. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones Mobile Devices*, SPSM 2014. ACM, 2014.

[101] Number of apps available in leading app stores as of july 2015, 2015. Available online at http://www.statista.com/statistics/276623/number-of-apps-available-in-leading- app-stores/.

[102] Ahren Studer, Timothy Passaro, and Lujo Bauer. Don't bump, shake on it: The exploitation of a popular accelerometer-based smart phone exchange and its secure replacement. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC 2011, pages 333–342. ACM, 2011.

[103] Vincent F. Taylor and Ivan Martinovic. A longitudinal study of app permission usage across the google play store. 2016. Technical report available at http://arxiv.org/abs/1606.01708.

[104] Why contactless cards can leave you with a losing deal, 2014. Available online at http://www.theguardian.com/money/2013/may/25/contactless-cards.

[105] Watch out for card clash. Available online at https://tfl.gov.uk/fares-and-payments/contactless/card-clash.

[106] Michael Velten, Peter Schneider, Sascha Wessel, and Claudia Eckert. User identity verification based on touchscreen interaction analysis in web contexts. In *Information Security Practice and Experience*. Springer International Publishing, 2015.

[107] Jose Vila and Ricardo J. Rodriguez. Practical experiences on NFC relay attacks with android. In *Radio Frequency Identification: 11th International Workshop*, RFIDsec 2015. Springer International Publishing, 2015.

[108] Device and sensors working group, 2016. Available online at https://www.w3.org/2009/dap/.

[109] W3C Working Draft Document on Media Capture and Streams. Available at http://w3c.github.io/mediacapture-main/getusermedia.html.

[110] W3C Geolocation API Specification. Available at dev.w3.org/geo/api/spec-source.html.

[111] W3C Working Draft Document on Ambient Light Events. Available at w3.org/TR/ambient-light/.

[112] W3C Working Draft Document on Device Orientation Event. Available at http://www.w3.org/TR/orientation-event/.

[113] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. Mole: Motion leaks through smartwatch sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom'15, pages 155–166, New York, NY, USA, 2015. ACM.

[114] Takuya Watanabe, Mitsuaki Akiyama, Tetsuya Sakai, and Tatsuya Mori. Understanding the inconsistencies between text descriptions and the use of privacy-sensitive resources of mobile apps. In *Eleventh Symposium On Usable Privacy and Security*, SOUPS 2015. USENIX Association, 2015.

[115] Zhi Xu, Kun Bai, and Sencun Zhu. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WISEC 2012. ACM, 2012.