# MALWARE ANALYSIS WITH SIDE-CHANNEL INFORMATION ON ANDROID SMARTPHONES

by

Erhan Davarcı

B.S., Electrical Electronics Engineering, Boğaziçi University, 2012

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical Electronics Engineering
Boğaziçi University
2015

MALWARE ANALYSIS WITH SIDE-CHANNEL INFORMATION ON ANDROID SMARTPHONES

APPROVED BY:

Prof. Emin Anarım .....
(Thesis Supervisor)

Prof. Mehmet Akar .....

Prof. Fatih Alagöz .....

DATE OF APPROVAL: 25.12.2015

# ACKNOWLEDGEMENTS

Firstly, I would like to thank my advisor, Professor Emin Anarım, for his guidance, great support and patience. Actually, he is not only advisor to me. His stimulating suggestions and encouragement helped me in all the time of studying on this thesis.

I want to thank the committee members of my thesis, Professor Fatih Alagöz and Professor Mehmet Akar, for their help and comments on this thesis.

I am thankful to İmran Ergüler and Betül Soysal for their advice and collaborations throughout the thesis.

I am also grateful to my family, especially my wife Begüm. They were always supporting and encouraging me with their best wishes.

Finally, I thank TÜBİTAK for supporting this thesis with BIDEB 2210.

# ABSTRACT

## MALWARE ANALYSIS WITH SIDE-CHANNEL INFORMATION ON ANDROID SMARTPHONES

In recent years, smartphones have become inseparable components of people's daily life. However, the great popularity of smartphones causes some security problems. For example, users may store their contact lists, passwords and other credentials in their smartphones. In addition to various features, smartphones also have various motion sensors like accelerometer and gyroscope. Therefore, number of malwares focused on smartphones goes up while capabilities of smartphones increase. It is well-known fact that Android has gained great popularity in smartphone technology and it is leader of smartphone market. This unluckily leads attackers to the Android platform. There are various malwares targeting Android smartphones like premium-SMS malwares. Since users intensely use touchscreen of their smartphones, attackers are highly interested in touchscreen nowadays. Furthermore, attackers use motion sensors as a side-channel information to obtain user's private data. There are several studies in the literature which show how motion sensors can be used as a side-channel. In this thesis, we firstly investigate malwares targeting Android smartphones, analyse their behavior and motivations. Then, we show that accelerometer sensor data can be used to infer user's input on touchscreen. We handle available studies in this area and propose new feature set for PIN inference by using accelerometer data. We considerably reduce the number of feature vectors while slightly improve the accuracy rate in PIN inference. Another important contribution of the thesis is to determine user's age interval by using accelerometer sensor as a side-channel. We show that by analyzing user's tap event on the touchscreen we can determine whether the user is child or adult. This is actually our great contribution to the literature because this information may give attackers extra avenues for their malicious acitivities.

# ÖZET

## ANDROİD TABANLI AKILLI TELEFONLARDA YAN KANAL BİLGİLERİ İLE KÖTÜ AMAÇLI YAZILIM ANALİZİ

Akıllı telefonlar son yıllarda insan hayatının ayrılmaz bir bileşeni haline gelmiştir. Fakat akıllı telefonların artan yaygınlığı bazı güvenlik problemlerine neden olabilmektedir. Örneğin, kullanıcılar rehber, şifre ve diğer kimlik bilgilerini telefonlarında saklayabilirler. Çeşitli özelliklere ek olarak, akıllı telefonlar ivmeölçer ve jiroskop gibi birçok hareket sensörüne sahiptir. Bundan dolayı, akıllı telefonların yetenekleri arttıkça akıllı telefonlara yönelik kötü amaçlı yazılımlar da artmaktadır. Android işletim sisteminin büyük bir rağbet gördüğü ve akıllı telefon pazarının lideri olduğu bilinen bir gerçektir. Bu durum maalesef saldırganların Android platformuna yönelmesine neden olmaktadır. Android platformunu hedef alan birçok kötü amaçlı yazılım vardır. Kullanıcılar yoğunlukla telefonun dokunmatik ekranıyla etkileşim içinde olduklarından, saldırganlar son zamanlarda ağırlıklı olarak dokunmatik ekranlarla ilgilenmektedirler. Ayrıca, saldırganlar akıllı telefonlardaki hareket sensörlerini yan kanal bilgi kaynağı olarak kullanıp kullanıcının özel bilgilerini elde edebilmektedirler. Literatürde hareket sensörlerinin yan kanal olarak nasıl kullanıldığını gösteren birçok çalışma mevcuttur. Bu tezde, ilk olarak Android akıllı telefonları hedef alan kötü amaçlı yazılımların davranışları incelenmiştir. Daha sonra, ivmeölçer verileri kullanarak kullanıcıların PIN kodu girişlerinin nasıl sezinlenebileceği gösterilmiştir. PIN kodu sezimi için yeni bir öznitelik kümesi önerilmiş ve daha yüksek bir başarı oranı elde edilmiştir. Tezin diğer bir önemli katkısı ise ivmeölçer verileri kullanılarak kullanıcının yaş aralığının tespit edilebileceğinin gösterilmesidir. Tezin bu kısmında, kullanıcının ekrana dokunuşları analiz edilerek kullanıcının çocuk veya yetişkin olduğu yüksek bir oranda tespit edilmiştir.

# TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . x
LIST OF SYMBOLS . . . xi
LIST OF ACRONYMS/ABBREVIATIONS . . . xii
1. INTRODUCTION . . . 1
2. OVERVIEW OF ANDROID STRUCTURE AND MALWARES . . . 4
2.1. Smart Devices . . . 4
2.2. Android Architecture and Security Overview . . . 5
2.3. Android Malware Analysis . . . 8
2.3.1. Malware Types . . . 8
2.3.2. History of Mobile Malwares . . . 10
2.3.3. Mobile Malware Motivations and Behaviors . . . 12
2.3.4. Analysis of Android Premium SMS Malware . . . 14
3. THEORETICAL KNOWLEDGE . . . 19
3.1. Dimensionality Reduction . . . 19
3.1.1. Principle Component Analysis . . . 19
3.2. Classification . . . 22
3.2.1. Logistic Regression . . . 22
3.2.2. K-Nearest Neighbor . . . 23
3.2.3. Decision Tree . . . 25
3.3. Cross Validation . . . 26
4. MOTION SENSOR-BASED SIDE CHANNELS IN ANDROID . . . 28
4.1. Motion Sensors of Android Smartphones . . . 28
4.2. Related Works on Sensor-based Side-Channels . . . 29
5. PROPOSED METHOD FOR PIN INFERENCE . . . 36
5.1. Application Development . . . 36

5.2. Data Acquisition . . . . . 38
5.3. Feature Extraction and Classification . . . . . 38
5.4. Proposed New Feature Set . . . . . 41
6. PROPOSED METHOD FOR USER AGE INTERVAL IDENTIFICATION . 48
6.1. Data Acquisition . . . . . 48
6.2. Feature Extraction and Classification . . . . . 49
7. CONCLUSION AND FUTURE WORK . . . . . 56
REFERENCES . . . . . 59

# LIST OF FIGURES

Figure 1.1. Worldwide Smartphone OS Market Share. . . . . 2

Figure 2.1. Popular Smartphone Applications. . . . . 6

Figure 2.2. Android Architecture. . . . . 7

Figure 2.3. Android Permission Screen Example. . . . . 7

Figure 2.4. Threat Families and Variants by Platform. . . . . 11

Figure 2.5. Top 20 Requested Permissions by Malicious and Benign Applications. 16

Figure 2.6. Sample SMS Application Layout. . . . . 17

Figure 2.7. Security Enhancement for Sending SMS to Short Number. . . . . 17

Figure 3.1. Example of the Logit Transformation and Logistic Regression Function. . . . . 22

Figure 3.2. Example of Decision Tree Construction. . . . . 26

Figure 3.3. Basic Scheme of Cross Validation. . . . . 27

Figure 4.1. Smartphone Orientation Angles. . . . . 30

Figure 4.2. Typical Patterns of Pitch and Roll Angles When Different Digit Keys are Pressed. . . . . 31

Figure 4.3. Distribution of Inferred Keys in Touchlogger. . . . . 32

Figure 4.4. Detection Results of Tap Position Inference in Taplogger. . . . . 33

Figure 5.1. The User Interface of Developed Sensor Application. . . . . 37

Figure 5.2. Example Sensor Data for 1234 PIN Entry. . . . . 40

Figure 5.3. 3, 10 and 20-Degree Polynomial Fit Examples. . . . . 45

Figure 6.1. Example AccSum for Child User. . . . . 50

Figure 6.2. The Fingerprint of Tap Event. . . . . 50

Figure 6.3. Example AccSum for Adult User. . . . . 51

Figure 6.4. Example of 9-degree Polynomial Fit to Tap Data. . . . . 53

Figure 6.5. Example IFFT of Tap Data. . . . . 54

Figure 6.6. Example iDCT of Tap Data. . . . . 55

# LIST OF TABLES

Table 2.1. Classification of 46 Pieces of Malware by Behavior. . . . . 12

Table 4.1. Android Sensors. . . . . 29

Table 5.1. Feature Set Used in PIN Inference. . . . . 39

Table 5.2. The Effects of FFT Size on Accuracy Rate of PIN Inference. . . . 43

Table 5.3. The Effects of Data Dimensions on Accuracy Rate of PIN Inference. 43

Table 5.4. The Effects of Polynomial Fit on Accuracy Rate of PIN Inference. 44

Table 5.5. The Effects of Normalization on Accuracy Rate of PIN Inference. . 44

Table 5.6. The Effects of DCT on Accuracy Rate of PIN Inference. . . . . 46

Table 5.7. Accuracy Results of PIN Inference for 3 Classification Methods. . . 46

Table 6.1. The Effects of 6 Features on Accuracy Rate. . . . . 53

Table 6.2. The Effects of iFFT and iDCT on Accuracy Rate. . . . . 54

Table 6.3. Accuracy Results for 3 Classification Methods. . . . . 55

# LIST OF SYMBOLS

| | |
|---|---|
| $C$ | Covariance matrix |
| $F$ | External force |
| $I$ | Identity matrix |
| $m$ | Size of fast fourier transform |
| | |
| $\alpha$ | Azimuth angle |
| $\beta$ | Pitch angle |
| $\gamma$ | Roll angle |
| $\lambda$ | Eigenvalue of matrix |

# LIST OF ACRONYMS/ABBREVIATIONS

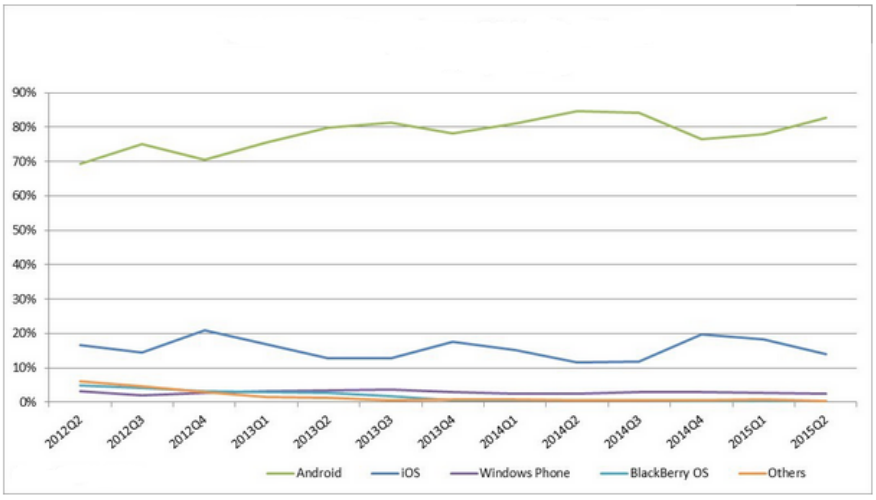
| | |
|---|---|
| API | Application Programming Interface |
| DCT | Discrete Cosine Transform |
| DVM | Dalvik Virtual Machine |
| FFT | Fast Fourier Transform |
| GPS | Global Positioning System |
| IDC | International Data Corporation |
| IDCT | Inverse Discrete Cosine Transform |
| IFFT | Inverse Fast Fourier Transform |
| IMEI | The International Mobile Station Equipment Identity |
| IOS | iPhone Operating System |
| IPC | Inter-Process Communication |
| MLE | Maximum Likelihood Estimation |
| MMS | Multimedia Messaging Service |
| NFC | Near Field Communication |
| OS | Operating System |
| PCA | Principle Component Analysis |
| SDK | Software Development Kit |
| SEO | Search Engine Optimization |
| SMS | Short Message Service |
| SVD | Singular Value Decomposition |
| SVM | Support Vector Machine |
| WEKA | Waikato Environment for Knowledge Analysis |

# 1. INTRODUCTION

In recent years, many innovations in various technological areas including mobile communication have occurred. Undoubtedly, smartphones are one of the milestones of recent years. Actually, as the new face of technology arises, people's needs become more sophisticated from day to day. They need more speed, quality and mobility. These features are combined in one solution which is small enough to be carried in pocket. In parallel with growing speed of technological advancement, smartphones have become inseparable components of people's daily life.

On the other hand, these small devices have a lot of private user data. Therefore, the growing popularity of smartphones leads to some privacy and security problems. For example, users store contact lists, pictures and videos from their surroundings. People also use their smartphones for online payment so they may want to store their credentials in smartphones. In addition, recent smartphones have various motion sensors such as accelerometer, gyroscope and orientation sensor. These sensors are quite practical and used for different purposes like gaming or user interaction. Unfortunately, number of malware targeting smartphones goes up as capabilities of smartphones increase. New technologies embedded in smartphones like GPS and NFC are also essential source for privacy and security problems.

Android is one of the operating system platform used in smartphones which has gained great popularity in smartphone revolution. According to data from International Data Corporation (IDC) Android has market share of 82.8% in 2015 Q2 [1]. This unluckily leads attackers to the Android Platform. Third-party applications installed via non-official markets are especially principal sources of security and privacy issues. Due to the open-market model of Android platform, there are a number of third-party application markets which also contribute to the increase of malicious applications for this platform.




| Period | Android | iOS | Windows Phone | BlackBerry OS | Others |
|---|---|---|---|---|---|
| 2015Q2 | 82.8% | 13.9% | 2.6% | 0.3% | 0.4% |
| 2014Q2 | 84.8% | 11.6% | 2.5% | 0.5% | 0.7% |
| 2013Q2 | 79.8% | 12.9% | 3.4% | 2.8% | 1.2% |
| 2012Q2 | 69.3% | 16.6% | 3.1% | 4.9% | 6.1% |

Figure 1.1. Worldwide Smartphone OS Market Share.

There are different types of malware targeting smartphones. While some of them try to send SMS to premium-rate numbers or make phone calls without user awareness, some malwares use motion sensors' data in order to obtain user's password. Since users intensely interact with their smartphones via touchscreen, touchscreen is focused by attackers nowadays. Therefore, there are several studies that investigated motion sensors' data on smartphones as a side-channel information source. These studies try to show how attackers use this side-channel information to steal user's private data.

In this thesis, we firstly focused on malwares targeting Android smartphones. Then, we spent time on motion sensor-based side-channels in Android smartphones. In this regard, we handled the study [2] in which the authors use the accelerometer sensor to learn user's password or pattern required to unlock smartphone. Actually, there are other similar studies dealing with the same issue, but in this study only

accelerometer sensor is used as a side-channel and it has large data in training set, so this study is more realistic compared to other studies. We implemented methods described in the study and showed that there are some unnecessary features used in analysis. Therefore, we proposed new feature set and finally obtained slightly higher success rate with only 33 features compared to 774 features in [2].

In addition, we used accelerometer sensor as a side-channel for different purpose and tried to determine user age interval by analyzing user's taps on touchscreen. We analysed accelerometer data, extracted features for user's tap fingerprints, and tried to determine whether the user is child or adult, to the best of our knowledge, this kind of analysis has not been studied yet. For user age interval identification, we obtained 87.1% accuracy rate.

The organization of this thesis is as follows: In the next chapter, overview of Android security structure, malware types, motion sensors on smartphones are investigated. Then, some theoretical knowledge is given. In the fourth chapter, accelerometer side-channels are analysed in detail. New proposed feature set for PIN inference is given in the fifth chapter. Then, child and adult user discrimination is studied. Conclusion and future works are given in the last chapter.

# 2. OVERVIEW OF ANDROID STRUCTURE AND MALWARES

## 2.1. Smart Devices

It is true that smartphones are of great importance in today's world. Smartphones are hand-hold devices that perform complex computing, information processing, and provide mobility and connectivity almost everywhere. In [3], the definition of smartphone is described as follows: A Smartphone is a mobile hand-held phone that uses an operating system supporting native third-party applications and includes multiple communication interfaces for providing connectivity. Actually, smartphone looks like a classic computer such that both have CPU, storage, operating system etc. However, there are also significant differences between them. First of all, since smartphones are hand-hold size, they have limited space so their structure must be very compact. While smartphones provide mobility, they are very dependent on their battery life. These smart devices in our pocket also support various technologies like Wi-Fi, Bluetooth, 3G. These features make smartphones different from classic computers. Like classic computers, smartphones also require an operating system. Although, there are various smartphone operating systems, we shorty mention about three major operating systems used in smartphones.

**Google Android**, Android is an operating system used for smart devices like smartphones and tablets. It is built on Linux Kernel. Android is open-source platform founded 2003 and purchased by Google in 2005. HTC Dream is the first smartphone to use Android in October 2008 [4]. Android platform can run native and third-party applications via `Google Play` or other application markets. It is well-known fact that Android became dominant of smartphone market.

**Apple iPhone**, iOS is the operating system runs on apple smart devices like iPad, iPhone, and iPod. Apple released Software Development Kit (SDK) for iOS in

2008. The iOS SDK can be used for developing, testing, installing new applications. The design of Apple iPhone contributes its market share, iPhone is also very rich in terms of smartphone applications [5]. Since applications development for iOS are restricted to non-official libraries, iOS applications are generally most secure ones. Therefore, there are little studies on iOS malwares.

**Windows Mobile**, Windows Mobile is operating system for smart devices developed by Microsoft. Although Microsoft joined smartphone market a little late, Microsoft Mobile has some different features compared to Android and iOS. It has a new user interface, it deeply integrated with social networks and Microsoft services. Interoperability with other Microsoft services is the main advantage of Windows Mobile but it has some disadvantages in terms of its security structure.

People use smartphones for both business and pleasure. According to the survey [6], most popular smartphone applications are shown in Figure 2.1. In this survey, participants were contacted twice a day over the course of one week and asked how they had used their phone in the preceding hour. Only those users who completed 10 or more surveys over the course of the study period are included in this analysis. As it can be seen, basic features such as text messaging and voice call are still on the top of the list.

## 2.2. Android Architecture and Security Overview

Android is owned by Google Inc. but Open Handset Alliance also contributes to its development. Since Android is an open-source project, there are many studies focused on it. Android is built on a Linux Kernel. It is designed according to five layers namely, Linux Kernel, Libraries, Android Run Time, Application Framework and Applications. During the operations these layers highly interact with each other and these interactions managed by Inter-Process Communication (IPC) mechanism called Binder. Dalvik Virtual Machine (DVM), is a kind of Java Virtual Machine designed for Android. DVM enables every applications to run its own process and this is called as sandboxing.

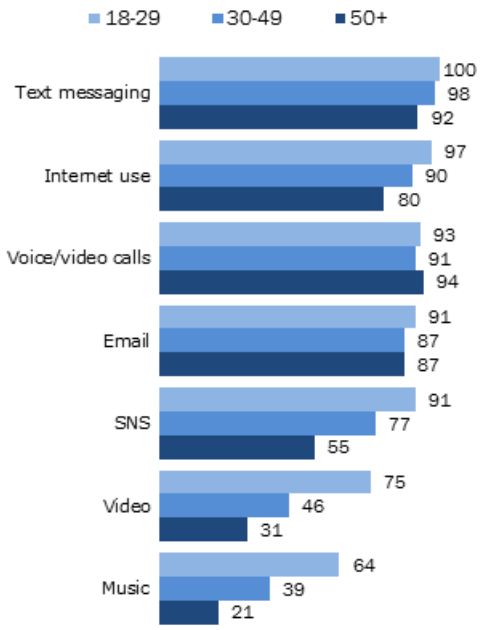



Figure 2.1. Popular Smartphone Applications.

Linux Kernel contains all the essential hardware devices. In addition, it acts as an abstraction layer between the device hardware and software stack. Libraries contain Android's native libraries and enable the device to handle different type of data. Android Runtime consists of Dalvik Virtual machine and Core Java libraries. Application Framework contains blocks such as activity manager, content providers, and telephony manager. It manages fundamental operations like resource management and voice call. Applications are at the tap layer in the Android structure. Applications like browser, games written by developers can be installed on this layer.

In addition to the practical features of Android, it also has some security-related features. Each application runs on its own Dalvik Virtual Machine with limited resources so applications are isolated from each other. Furthermore, every application developed for Android platform must have a manifest file. This file includes essential information about the application such as java package used, application components and especially the list of permissions required by the application. By looking permissions required by the application, users can have knowledge about whether the application

Figure 2.2. Android Architecture.

contains a malicious content or not. Example of application permission screen is shown in Figure 2.3. For instance, if a basic calculator application requires SMS send permission, users can think that this application may have malicious purposes. However, most users display unconscious behavior and ignore the details of permission list.

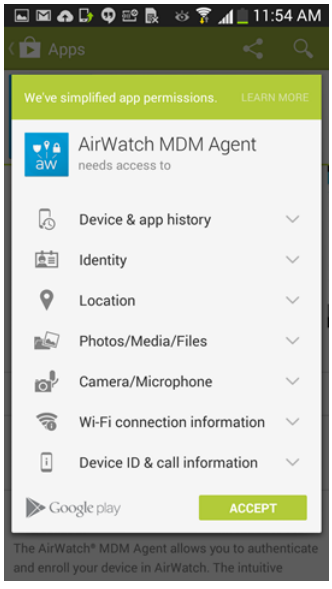



Figure 2.3. Android Permission Screen Example.

Google has also forces users to use `Google Play` store to download applications. This is because, `Google Play`, previously known as `Android Market`, has its own security mechanism. It scans application before upload it to the market, if any malicious content is recognized the application is blacklisted. Although, this is a good effort against malware, malware detection rate is not perfect, some applications with malicious contents can pass the security mechanism of the market. Moreover, this security mechanism is only valid for Google's official store, users who download applications via other markets are always under the risk of malware.

## 2.3. Android Malware Analysis

### 2.3.1. Malware Types

Malware is a combination of the two words malicious and software, and mobile malware is malicious software that is especially created to attack mobile phones or smartphones. The main goals of mobile malwares are gaining profit, leaking personal information and harming smartphone. There are different types of malwares, popular ones can be summarized as follows:

**A Virus** a malicious software which is hidden into an executable file and when the infected program is executed it replicates by inserting copies of itself into another program. Therefore, those codes infected by the virus may infect the new program in turn. This action is called as a self-replication, and it is one of the main characteristics of malware. Stealing private information and hard disk space, corrupting data, logging user's input, displaying political messages on the screen are examples of harmful operations performed by viruses. Viruses usually replicate themselves by user interaction like opening a link that sent by unknown sender or installing a program from unknown source [7].

**A Worm** is a self-replicating piece of code that spreads via networks and usually does not require human interaction to propagate [8]. Although worm is quite similar to virus, worm does not need to copy itself into an existing program. Therefore its

self-replication process can be proceeded standalone without dependent to executable code. Furthermore, while viruses usually corrupt or modify files on a device, worms always harm the network. Since worm can replicate itself without user interaction, it may send out millions of copies of itself over the network and cause huge disruptive effect.

**A Trojan Horse** is a program that appears to have some useful of benign purpose, but really masks some hidden malicious functionality [8]. At the first glance, the Trojan horse will appear as a normal or useful program but will actually be harmful once it is installed or runned on the device. Users who receive a Trojan horse will be tricked to open or run it because Trojan horse appears like legitimate software or files from a legitimate source. For example, an attacker may alter the malicious code name on a system so the Trojan horse will appear to belong to that machine. Since Trojan horses can mask itself as some normal process, it is hard to detect them by users [9].

**A Spyware** can be defined as any software that monitors user behavior, or gathers information about the user without awereness of the user [10]. Spyware can store personal information, capture keystrokes or take screen shots. It can also cause financial loss. Spyware usually appears as a part of pre-installed software but it sometimes comes via e-mail as an attachment or hyperlink.

**An Adware** or advertising-supported software is a kind of software embedded in the application and automatically show advertisements. These advertisements may appear in the user interface or installation process of the application. The main purpose is to supply revenue for owner of applications. Actually, adware is an alternative way offered to users who do not want to pay for software. Some software developers offer their product as sponsored freeware, Adware, and advertisements are disappeared if users pay to register.

### 2.3.2. History of Mobile Malwares

First attempts of mobile malwares began in 2004. Cabir was the first mobile worm in the world designed to infect Symbian based Nokia Series 60. Due to the Cabir, the word "Caribe" appears on the screen of infected phones. This worm uses Bluetooth to spread itself. Skull is another malware in that year, it overwrites the program files on the mobile device and causes them to stop [11]. By 2005, mobile malwares started to evolve to information theft. There were malwares that copies device's address book or contact list and send via MMS. Then, mobile malware was extended to other platforms like Windows Mobile. Attackers started to interested in malwares that use MMS or SMS services due to their economic aspects. From 2007 to 2009, the number of malwares sending premium-rate SMS messages without user's knowledge increased.

In 2010, radical changes on mobile malware occurred. Malwares extended from individuals or small groups to large scale worldwide basis. Malware developers noticed that they can benefit from malwares to make a lot of money easily. For example, Chinese SexySpace virus was designed to send SMS messages to all phone numbers in the contact list and directed users to download pornographic content. Zeus Zitmo is another malware designed to forward text messages related to bank account operations to attackers.

By the way, two new platforms, iPhone iOS and Android started to dominate smartphone market in 2010. Geinimi was one of the first malwares designed for Android in that year. Geinimi used the infected phone as a botnet and communicated with remote server in order to control the phone. In 2011, explosion of Android continued rapidly so malware developers noticed its potential. The number of mobile malwares increased drastically and they became more complex. For instance, DroidKungFu emerged with unique characteristics including an exploit to root of the phone. The NickSpy Trojan was recording phone conversations, text messages, call data, GPS coordinates and photos, uploading them to a remote server [12].

The increasing growth of Android threats continued in 2012. SMS Trojans were dominant behavior among all other malwares.

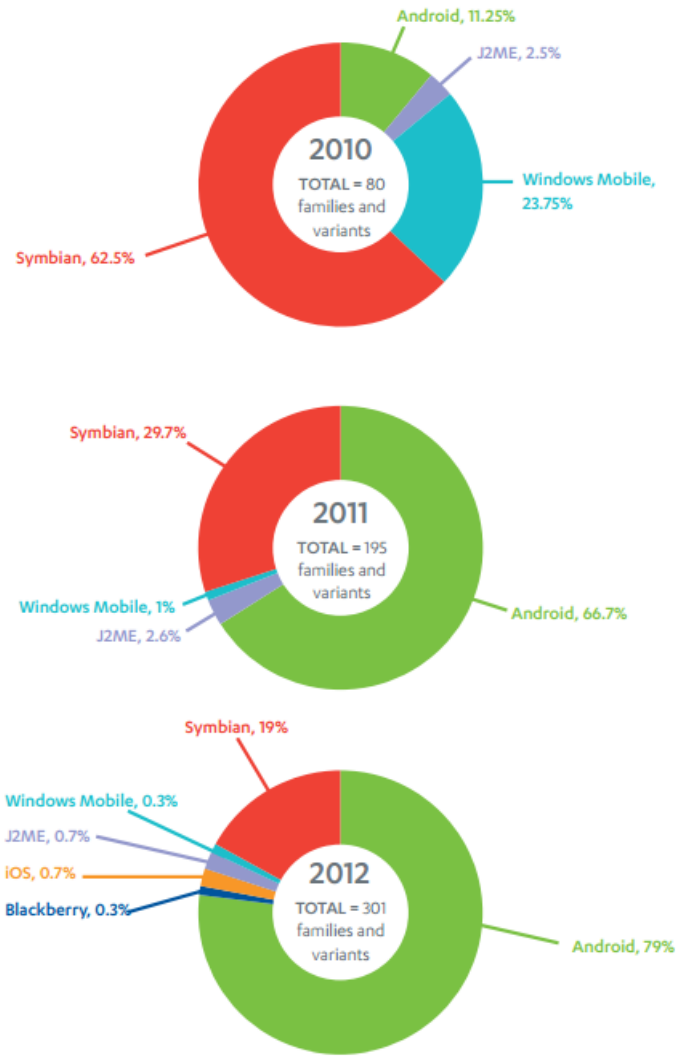



Figure 2.4. Threat Families and Variants by Platform.

2013 is starting year of new malwares for Android. FakeDefend was the first ransomware for Android mobile phones. It locks the phone and requires the victim to pay a high antivirus subscription fee in order to retrieve the contents of the device. Chuli, introduced same year, was considered the first targeted attack on the Android platform. Cybercriminals behind the attack stolen the e-mail account of an activist at the World Uyghur Conference, held March 11-13, 2013 in Geneva, to target the accounts of other Tibetan Human Rights activists and advocates. The e-mails sent from the

hacked account included Chuli as an attachment. By doing so, attackers obtained data such as incoming SMS and phone contacts, location information, and recordings of phone calls. The captured information was then sent to a remote server [13].

### 2.3.3. Mobile Malware Motivations and Behaviors

In [14], authors categorize known iOS, Symbian, and Android malwares by the motivations indicated by their behavior. They present a table showing an overview of the behavioral classification of 46 pieces of malware by behavior.

Table 2.1. Classification of 46 Pieces of Malware by Behavior.

| | |
|---|---|
| Exfiltrates user information | 28 |
| Premium calls or SMS | 24 |
| Sends SMS advertisement spam | 8 |
| Novelty and amusement | 6 |
| Exfiltrates user credentials | 4 |
| Search engine optimization | 1 |
| Ransom | 1 |

According to the study [14], mobile malwares have following malicious motivations:

(i) Novelty and Amusement

This kind of malwares are intended to amuse the author like changing the wallpaper of phone or sending anti-religion text messages from phones. The number of this type of malwares is not high and it is expected that amusement-driven mobile malware will decrease and become replaced by profit-driven malware.

(ii) Obtaining User Information

It is possible to obtain user information like user's location, list of contacts, browser and download history via applications querying mobile APIs. This kind of data can be used by malware writers to sell them for revenue. The number of malware with this incentive is high since it is related with money. Hackers

can choose to sell users' location or browsing histories to advertising companies, contact lists to scammers, spammers or phishers to make money. In addition, some applications may store user-specific data on the phone so malware could scan these applications and related files to obtain more information about users.

(iii) Stealing User Credentials

People enter various passwords for banking, shopping, e-mail accounts while using their smartphone. For instance, banks benefit from smartphones for two-factor authentication. Users may also save their passwords and credit cards information on their phones. These all factors make smartphones target for credential theft. Therefore, it is expected to increase the number of this type of malware in the feature.

(iv) Premium-Rate Calls and SMSs

Legitimate premium-rate phone calls and SMS messages deliver valuable content, such as stock quotes or adult services. These services are generally very expensive, they cost several dollars per minute or per message and their cost is added to the user's phone bill. Therefore, these services are focused by attackers in order to make a profit. For instance, Geinimi was designed to send premium SMS messages to the numbers determined by remote commands. In addition, recent studies show that mobile malwares related to premium-rate calls and SMS are one of the dominant malware types.

(v) SMS Spam

SMS spam is used as an alternative way for advertising and phishing links. Since sending SMS spam is illegal in most countries, commercial spammers use malicious programs to send SMS spams. Malicious codes can be embedded into popular applications so user might not notice the outgoing SMS messages. Users realize the situation when their monthly phone bills arrive.

(vi) Search Engine Optimization

Search Engine Optimization (SEO) is mainly affecting web page ranking in search engine's results. Search engines sort web pages according to how relevant each web page is to a given search term. Relevance means users' clicks according to search result. For example, you search a term via search engine then click on one web site, that web site will rise in the results. Therefore, malware can be used to increase a web site's ranking in search engine results. This malware sends web requests to the search engine for the target search term. The malware then fraudulently clicks on the search result that corresponds to the target web site. As a result, the web site's rank for that search term will increase. For example, ADRD was developed to boost the Baidu search result ranking of a Chinese web site [15].

(vii) Ransom

A Dutch worm locked iPhone screens and demanded 5 euros to unlock the screens of infected phones [14]. In addition, malware can be used for blackmail. A malicious program may steal user's browser history, bookmarks on the phone and publish them on the Internet with name of the user. Then, attackers may demand some money from the user to delete this data.

### 2.3.4. Analysis of Android Premium SMS Malware

Since, SMS malwares are highly profitable for attackers, they are one of the main types of malwares for smartphones. Therefore, we spent some time on SMS malware analysis. In [16], it is stated that on the Android, there is a permission-guarded function $sendTextMessage$ that allows sending an SMS message in the background without user's awareness. There are some examples of this type of malware in different countries. For example, FakePlayer malware sends SMS message "798657" to multiple premium-rate numbers in Russia or GGTracker automatically signs up the user to premium services in US without user's awareness. The researchers state that, in some premium SMS malwares, premium-rate numbers are embedded in the malicious code. However, in others premium-rate numbers are dynamically controlled remotely and

determined at runtime. The second types are more dangerous because you cannot determine a malicious behavior by simply searching premium-rate numbers in the code.

In addition, there are extra confirmation policies for subscribing to premium-rate services in some countries like China. Users must reply to incoming SMS messages sent from service provider in order to active their service subscription. SMS malwares reply these activation messages by themselves without awareness of the user. They can also filter the billing-related messages from premium-rate SMS services.

In [16], the authors also compared top permissions requested by malicious and benign applications, results shown in Figure 2.5. Actually, these permissions are declared in application manifest file. They observed that requiring some permissions like $READ_SMS$ or $WRITE_SMS$ by irrelevant applications can be a sign for malicious behavior. They concluded that there are 790 samples (62.7%) in the dataset that request the $READ_SMS$ permission, while less than 33 benign applications (or 2.6%) request this permission. These results are consistent with the fact that 28 malware families in their dataset (or 45.3% of the samples) have the SMS-related malicious functionality. Furthermore, they realized that malicious applications usually request more permissions compared to benign ones.

Furthermore, malicious application can install a $BroadcastReceiver$ for incoming messages such that as the messages arrive, they are received by the malicious application and processed. For instance, the malware sends SMS to subscribe to a premium-rate services. Then it listens the SMSs coming from this service and replies this SMS to approve the subscription. Finally, it removes this received SMS from inbox. HippoSMS is example of this kind of malware [17].

In [18], another typical behavior of SMS malwares is stated. They usually prefer certain time slots to perform their malicious activity for example while the user is sleeping. Actually, this behavior can be used against SMS malwares. In [19], the authors state that sending an SMS message is strongly related with the user activity. In a normal usage, an SMS is sent after the user has typing his message, which requires

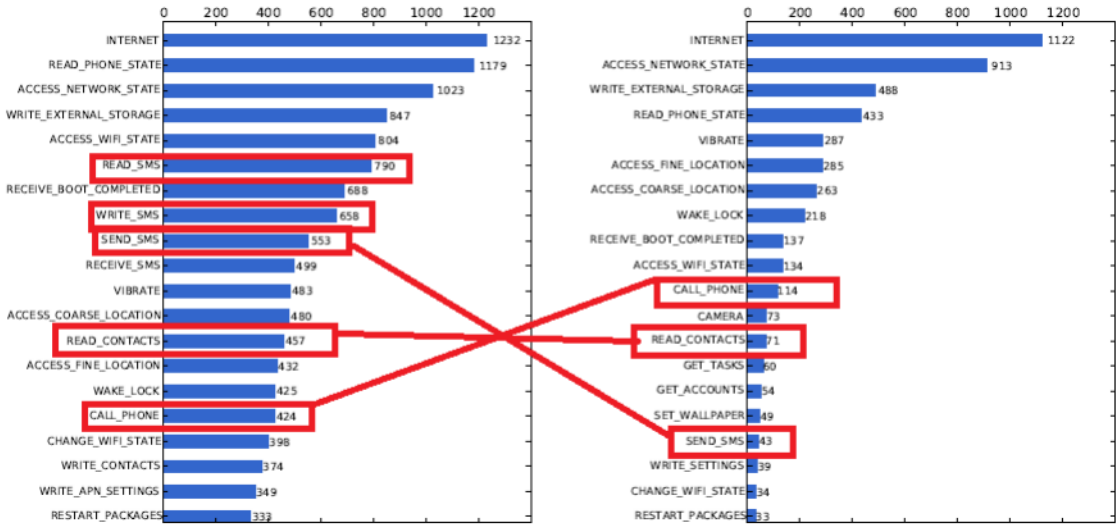



Figure 2.5. Top 20 Requested Permissions by Malicious and Benign Applications.

an active interaction from the user. Therefore, if one application sends SMS messages when the user is idle, it can be considered as malware. They describe MADAM, a Multi-level Anomaly Detector for Android Malware. As an example, to detect sent SMSs, MADAM exploits the Android system log file (LogCat), which contains the output of a low level function that is called each time an SMS is being sent. It also monitors user's activity so it can detect SMSs sent when the user is idle.

On the other hand, Android develops a countermeasure to deal with this kind of premium SMS malwares. Beginning with Android 4.4, only one application, namely default SMS application, can receive the new $SMS_DELIVER_ACTION$ intent, which is broadcasts of the system when a new SMS message arrives. Also, only the application chosen as a default SMS application is able to write to the SMS Provider. Other application can only realize if a new SMS arrives.

In order to better understand the SMS sending concept, we developed a simple SMS application by using Android SDK. This was useful for us to practice manifest file, user interface and other capabilities of Android SDK because we do not have any application development experience before.

Figure 2.6. Sample SMS Application Layout.

However, during the analysis of the Android tutorial, we learned a new security enhancements coming with Android version 4.2 or higher. The default Android operating system asks the approval of the user when an application tries to send an SMS to short numbers which are usually used by premium services. Users can choose whether to allow the application to send its message or block it.

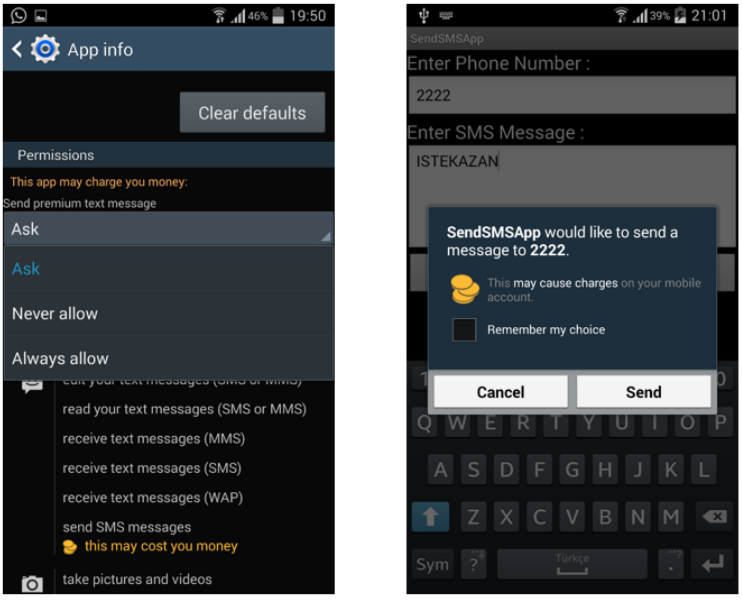



Figure 2.7. Security Enhancement for Sending SMS to Short Number.

Although, there is no clear information in the literature showing that how this security enhancement is bypassed, studies in 2014 and 2015 show that these types of malwares are still active and researchers still study on detecting premium SMS malware activities [20–22]. However, it is stated that bypassing this security precaution requires complex Android code design at kernel level. Therefore, we try to study on alternative ways to deal with this precaution. Also, we investigate the vulnerabilities of the platform as a white hacker. For example, we study on how we combine malware analysis and side-channel information on smartphones.

# 3. THEORETICAL KNOWLEDGE

## 3.1. Dimensionality Reduction

The number of instances and attributes in training dataset may increase the complexity of classification algorithm. Therefore, we should decrease dimensions of training data to save space and time for classification. Actually, some classifiers have their own dimension reduction capability and use whichever features are necessary or irrelevant. However, performing dimension reduction before classification is more suitable because if we can represent data with fewer features, we obtain a better idea about next processes. Also, by ignoring irrelevant features we can save the cost of processing them.

Actually, there are two main methods for dimension reduction namely, feature selection and feature extraction. In feature selection, we select subset of available features which gives most information about our data. On the other hand, we try to find the set of features which are combinations of available features in feature extraction. Principle Component Analysis (PCA) is most widely used method for feature extraction [23]. Therefore, we used PCA in our study for dimension reduction.

### 3.1.1. Principle Component Analysis

PCA is a technique that transforms a number of correlated features into smaller number of features called principle components. PCA uses complex mathematical principles especially linear algebra fundamentals. Its main goal is to reduce the dimension of data with minimum loss of information. PCA first takes the whole data and computes covariance matrix of the whole data set. Then, it computes eigenvectors and corresponding eigenvalues. Eigenvectors are sorted by eigenvalues and eigenvectors with higher eigenvalues are chosen to form new matrix. PCA uses this new matrix to transform data features into new subspace.

A summary of the steps in PCA is as follows:

(i) Center the data.
(ii) Obtain the covariance matrix.
(iii) Compute the eigenvectors and eigenvalues from the covariance matrix.
(iv) Sort eigenvectors in descending order of eigenvalues and choose the eigenvectors that correspond to the largest eigenvalues.
(v) Construct the projection matrix from the selected eigenvectors.
(vi) Transform the original data via projection matrix.

First of all, lets have $D_{n\times d}$ dataset where $n$ is dimensions and $d$ is data samples. We should calculate mean of the values in each dimension and then center our data set to obtain adjusted data set $A$:

$$A_{n\times d} = D_{n\times d} - M_{n\times d}$$

where $M$ is mean matrix.

Then, lets calculate the covariance matrix of $A$:

$$C = \frac{1}{n-1} A A^T$$

where $C$ is $n \times n$ matrix.

The next step is calculating eigenvectors and eigenvalues of $C$ by solving the following equation.

$$Cu = \lambda u$$

where $\lambda$ is an eigenvalue and $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ is an eigenvector.

Then, if some eigenvalues are 0 or very small, we can ignore those eigenvalues and the corresponding eigenvectors so dimensionality is reduced.

We now transform our data set to the new basis:

$E = P^T A$ where

- $E$ is the transformed data set.
- $P^T$ is the transpose of the $P$ matrix containing the eigenvectors.
- $A$ is the centered data set.

It is clear that the dimensions of the new data set, $E$, are less than the data set $A$ because of ignoring some eigenvectors.

Note that by using Singular Value Decomposition (SVD), we can decompose matrix $A$ as follows:

$$A = USV^T$$

where $U$ and $V$ are ortogonal matrices, and $S$ is a diagonal matrix consists of singular values. Then, multiplying both sides by $A^T$ on the right results as follows:

$$\begin{aligned} AA^T &= USV^T(USV^T)^T \\ AA^T &= USV^T(VSU^T) \end{aligned}$$

$$V^TV = I$$

$$AA^T = US^2U^T$$

Therefore, if we know $U$ and $S$, we can find the eigenvectors and eigenvalues of $AA^T$. This is because, columns of $U$ contain the eigenvectors of $AA^T$ and the

eigenvalues of $AA^T$ are the squares of the singular values in $S$.

In PCA we basically try to find eigenvalues and eigenvectors of the covariance matrix, $C$. We know that $C = (AA^T)/(n-1)$, and thus finding the eigenvalues and eigenvectors of $C$ is the same as finding the eigenvalues and eigenvectors of $AA^T$.

Thus, there is no need to compute covariance matrix $C$ if we use SVD. SVD gives the eigenvectors and eigenvalues that we need for PCA.

## 3.2. Classification

In this section, we briefly mention about the general concepts of some classification models we used on our data.

### 3.2.1. Logistic Regression

Linear regression is way of measuring the relationship between two variables. On the other hand, logistic regression is a kind of classification model. It is based on the logistic function which is always between 0 and 1. Logistic regression fits an S-shaped curve to the data. Examples of logit transformation and logistic regression function are shown in Figure 3.1 [24].

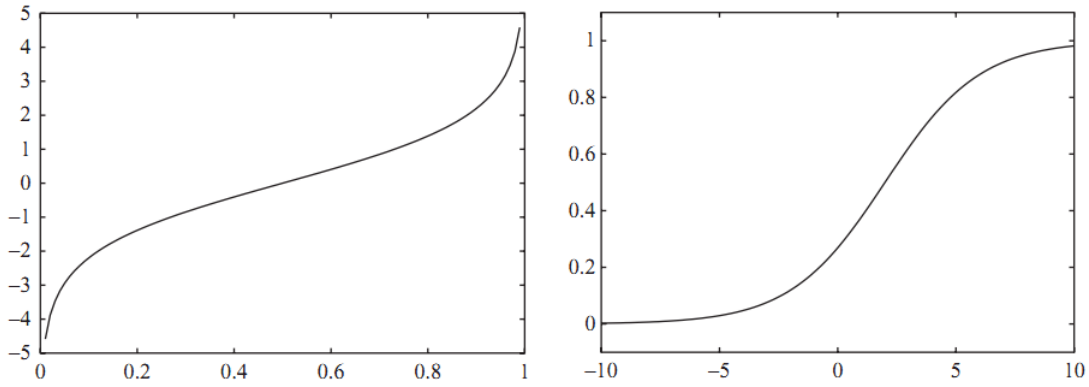



Figure 3.1. Example of the Logit Transformation and Logistic Regression Function.

First of all, logistic regression ensures that the predicted values are always between 0 and 1. In addition, the predicted values correspond to the probability of $Y$ belonging to one class. Let's assume two class problem where $Y$ is 0 or 1.

A regression is performed with a transformed values of $Y$ via logit function.

$$Logit(Y) = \ln(odds) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

where $\beta_0$ is intercept or constant term of regression, $\beta_1$ and $\beta_2$ are coefficients. Furthermore, *odds* means the odds of $Y$ being equal to 1. *Odds* can be defined as the probability of belonging to one class divided by the probability of belonging to the other class.

$$odds = \frac{p}{1-p} \implies p = \frac{odds}{odds+1}.$$

Then, we can take the exponent of both sides and divide both sides by $(odds+1)$

$$e^{\ln(odds)} = odds = e^{(\beta_0+\beta_1 x_1+\beta_2 x_2)},$$

$$p = \frac{odds}{odds+1} = \frac{e^{(\beta_0+\beta_1 x_1+\beta_2 x_2)}}{e^{(\beta_0+\beta_1 x_1+\beta_2 x_2)}+1}$$

where $p$ is the probability of $Y$ belongs to a class, i.e. $Y = 1$.

Maximum Likelihood Estimation (MLE) technique is usually used to estimate coefficients of regression.

### 3.2.2. K-Nearest Neighbor

K-Nearest Neighbor algorithm is a non-parametric method used for classification and regression. K-NN is a type of instance-based learning, or lazy learning. It is one of the fundamental of all machine learning algorithms [25].

K-NN actually stores all available cases and classifies a new case according to similarity measure like distance functions. Popular distance functions used in k-NN are Euclidean, Manhattan and Minkowski distance.

$$\text{Euclidean:} \qquad \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2},$$

$$\text{Manhattan:} \qquad \sum_{i=1}^{k}|x_i - y_i|,$$

$$\text{Minkowski:} \qquad \left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}.$$

K-NN computes $k$ nearest neighbors of a given case according to one of distance functions, then classifies a case by a majority vote of its neighbors. If $k = 1$, then the case is assigned to the class of its nearest neighbor. Obviously, it is useful to choose $k$ to be an odd number because this avoids tied votes.

However, all of $k$ neighbors have the same contribution to classification purpose in classical k-NN. Therefore, it can be useful to assign weights to the neighbors so that the closest neighbors contribute more to the average than distant ones. For instance, we should give each neighbor a weight of $1/d$, where $d$ is the distance to the neighbor.

Choosing the optimal value for $k$ is critical in k-NN algorithm. A large $k$ is generally more precise because it reduces the overall noise but there is no guarantee. On the other hand, cross validation can be used to determine the optimal value of $k$ by using an independent dataset to validate different $k$ values.

A disadvantage of the k-NN algorithm is that it is sensitive to the local structure of data. For example, variables in data set may have different measurement scales such that one variable is between 100 and 1000, the other is between 0 and 1 so the first one will have higher effect on the distance. To avoid this, dataset should be standardized before applying k-NN.

In addition, before applying k-NN it can be helpful to use one of the dimensionality reduction techniques like PCA. PCA reduces calculation time for k-NN by removing redundant variables. PCA also avoids the effects of the curse of dimensionality. The curse of dimensionality in the k-NN means that the distance is unhelpful in high dimensions because all vectors are almost equidistant to the search query vector.

### 3.2.3. Decision Tree

Decision tree is one of the popular nonparametric classification methods. In decision tree, features are ordered according to their importance by information gain method and the most important one is selected as a root node. The other features are mapped to tree nodes. New branches are created for threshold values of corresponding feature and training instances are divided into these branches by considering their values. Then, algorithm continues to run recursively until all of the leaf nodes have the same labeled instances.

Example of decision tree construction is shown in Figure 3.2. Decision nodes are oval nodes and leaf nodes are rectangular ones. Note that successive splits are orthogonal to each other.

In decision tree, constructing the smallest tree is the main purpose. Most decision tree algorithms use Entropy and Information Gain to construct a decision tree. Entropy is the measure of uncertainty. While constructing a tree, the data is splitting into branches that contain instances with similar values, in other words homogenous instances. Entropy is used to calculate the homogeneity of data. If given data is completely homogeneous the entropy is zero while the data is uniformly partitioned the entropy is one [26]. For a two class problem, entropy can be defined as:

$$Entropy: \quad f(p, 1-p) = -p \cdot \log_2 p - (1-p) \cdot \log_2(1-p).$$

While splitting data into branches, we should find an attribute which has the highest information gain. The highest information gain corresponds to minimum entropy.

Therefore, splitting the instances results in maximum decrease in entropy. Finally, an attribute with the largest information gain is selected as the decision node, a branch with zero entropy is a leaf node. We also continue to split branches with entropy more than zero.

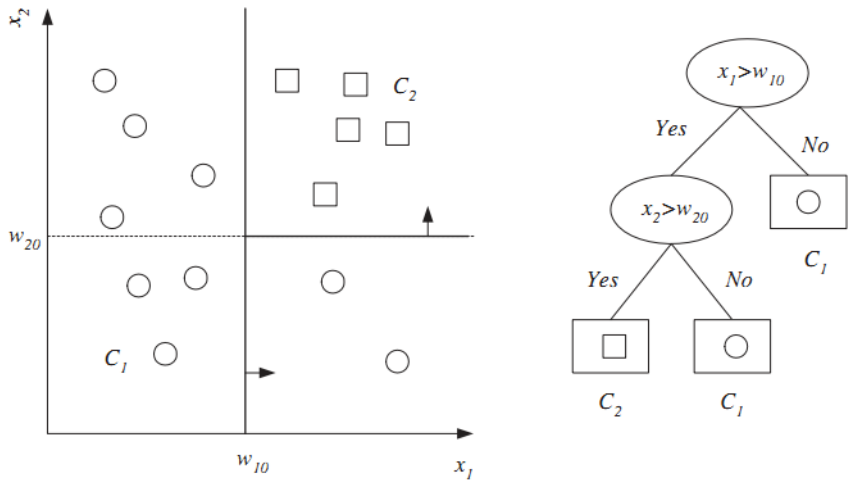



Figure 3.2. Example of Decision Tree Construction.

## 3.3. Cross Validation

Cross validation is a statistical method for evaluating and comparing classification algorithms by dividing data into two parts. One part is used to train a model and the other is used to validate this model. N-fold cross validation is the basic form of cross validation. Other types of cross validation are special cases of n-fold cross validation or involve repeated rounds of n-fold cross validation [27].

Firstly, data set is equally divided into $n$ parts in this technique. Then, cross validation runs such that it uses $n-1$ parts of data for training and other part for testing. In the next step, it uses another part for testing and remaining $n-1$ parts for training. This process continues until all parts are used for testing and training purpose. Therefore, this runs $n$ times and gives $n$ prediction accuracies. Then, these

$n$ accuracies are averaged to obtain final success rate of cross validation.

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i$$

where $E_i$ is the accuracy for ith fold and $E$ is the final accuracy.

The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once. In general, 5 or 10 fold cross validation is used.

Figure 3.3. Basic Scheme of Cross Validation.

# 4. MOTION SENSOR-BASED SIDE CHANNELS IN ANDROID

## 4.1. Motion Sensors of Android Smartphones

Android smartphones are equipped with various sensors that measure motions and different environmental conditions. There are three main categories of sensors in Android platform namely, motion sensors, position sensors and environmental sensors. Some of these sensors are hardware-based and others are software-based. Hardware-based sensors are physical components embedded into device such as accelerometer and gyroscope. They obtain their data by directly measuring specific environmental properties. Software-based sensors are not physical components, they derive their data from one or more of the hardware-based sensors so they are sometimes called virtual sensors. The linear acceleration and the gravity sensor are examples of these type of sensors.

Table 4.1 lists these sensors and their descriptions. Motion sensors monitor the reaction of direct user input. All motion sensors give their data as a multi-dimensional array. For example, the accelerometer returns acceleration force data for three coordinates, and the gyroscope gives rate of rotation data for three coordinates.

In studies [2, 28, 29], motion sensors such as accelerometer, gyroscope, orientation sensor are used to obtain private data or infer user's input. Similarly, in our study we are interested in motion sensors, especially accelerometer sensor. Actually, besides side-channels motion sensors can be used for different tasks. For example, in [30–32] motion sensors are used as a biometric identifier for authentication purpose.

Table 4.1. Android Sensors.

| **Sensor** | **Description** |
|---|---|
| Accelerometer | Measures the acceleration force on all three dimensions |
| Gyroscope | Measures the rotation rate of device around all three dimensions |
| Ambient Temperature | Measures the room temperature |
| Gravity | Measures the gravity force on all three dimensions |
| Linear Acceleration | Measures the acceleration force on all dimensions, excluding the gravity force |
| Light | Measures the ambient light level |
| Magnetic Field | Measures the geomagnetic field for all three dimensions |
| Orientation | Measures degrees of rotation around all three dimensions |
| Temperature | Measures the device temperature |
| Pressure | Measures the air pressure |
| Proximity | Measures the proximity of an object to the device screen |
| Relative Humidity | Measures the relative surrounding humidity |
| Rotation Vector | Measures the device orientation |

## 4.2. Related Works on Sensor-based Side-Channels

Sensor-based smartphone side-channels are described as information leakage due to sensor readings and these side-channels are mainly caused by touchscreen interaction of users [33]. In this section, we give analysis of studies about motion sensor-based side-channels.

In [28], the authors described a new side-channel, motion on touch screen smartphones due to soft keyboard interaction. They developed an application called Touchlogger and obtained inputs entered via soft keyboard using device orientation data. They obtained device orientation features and correctly inferred more than 70% of the keys typed on the soft keyboard.

The device orientation data is obtained from orientation events which are mainly derived from the accelerometer sensor output. Firstly, they ignored the azimuth angle ($\alpha$) because rotation caused by typing mainly affects pitch ($\beta$) and roll ($\gamma$) angles. Since the average orientation angle represents the initial position of the device, the average value is eliminated and orientation data is normalized in the next step. Finally, the Touchlogger application listens the sensor data and calculates Peak-to-Average values

to understand whether any tap event is occurred and then records the start and end of the tap events. This is because the Peak-to-Average ratios of the ($\beta$) and ($\gamma$) angles are much larger during typing and can be used to determine tap events.

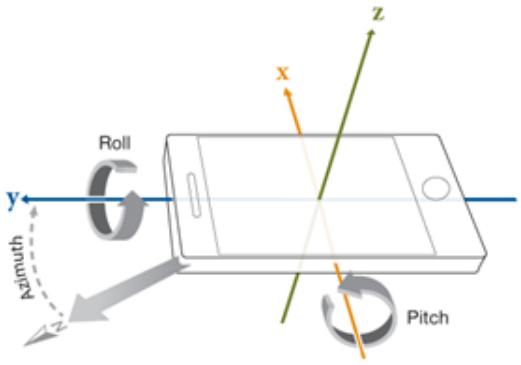



Figure 4.1. Smartphone Orientation Angles.

The main goal in this study is to understand which number is entered. To determine the entered number, they plotted the change of the pitch and roll angles as shown in Figure 4.2.

Figure 4.2 shows the paths followed by pitch and roll angles, and each path consists of two main lobes. They observed that the lobes of the patterns produced by the same key point to similar directions, while the angles of the lobes vary for different keys. Therefore, they extracted two features for each pattern. The probability distributions of these features are assumed to have Gaussian distribution. The mean and standard deviations of the feature vectors are calculated from the collected data for each number entry. Their first evaluation shows an accuracy rate of 50.6%

Then, they added two more pair of features related to upper and lower lobe. This feature increase results in accuracy rate of 71.5%.

The inference results on each key are shown in Figure 4.3. They showed that 321 out of 449 keystrokes were correctly inferred. Digits 1 and 9 have the highest inference accuracy so they concluded that physical proximity decreases inference accuracy. In

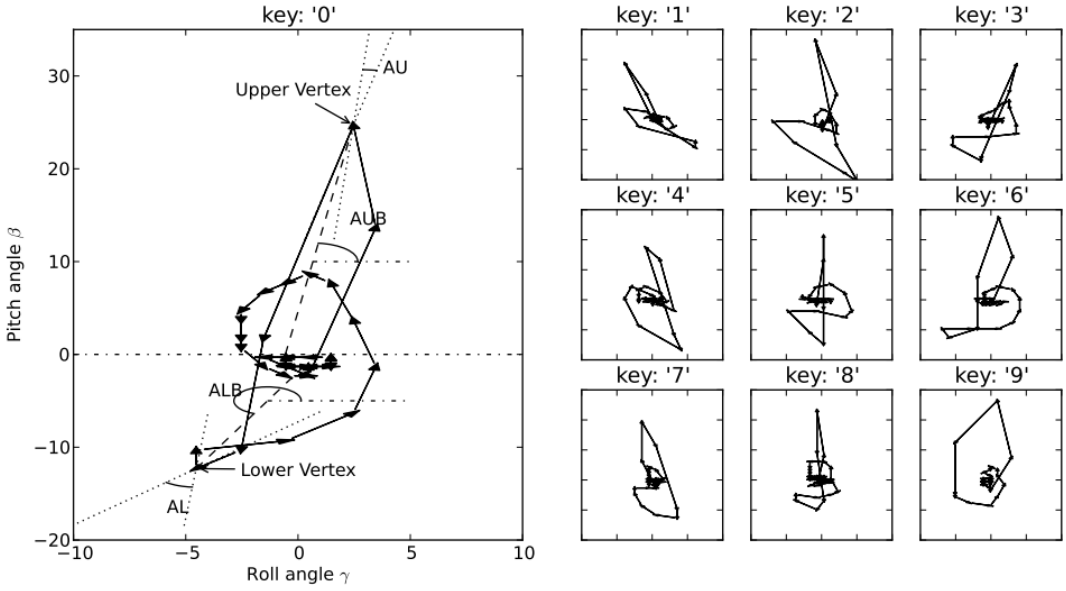



Figure 4.2. Typical Patterns of Pitch and Roll Angles When Different Digit Keys are Pressed.

the testing phase, keystrokes were collected from the same user on different days so this actually cannot reflect real scenario. They should test your models on larger set of users.

Another study that monitors motion and gesture changes of smartphones in order to infer tap events and entered number is given in [29]. The authors focused on correlations between the tap events and the motion change of smartphone. Actually, the study has two phases, namely tap event detection and tap position inference. Finally, an application called as Taplogger was developed. Taplogger has ability to detect automatically the numbers entered during phone call and during screen unlocking. To realize these scenarios, the application listens the broadcast receiver for understanding that phone call occurred and listens device state changes to understand that screen unlock occurred.

In the first phase, the authors tried to detect tap events by using the accelerometer readings. To detect the tap events, the unique change pattern of external force on the

| Actual Key | Inferred Key | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | 64% | - | 6% | 10% | - | 12% | - | - | 8% | - |
| 1 | - | 86.3% | - | - | 13.7% | - | - | - | - | - |
| 2 | 8.3% | 4.2% | 68.8% | 4.2% | - | 2.1% | 3.1% | 4.2% | 6.2% | - |
| 3 | 18% | - | - | 70% | - | - | 6% | - | - | 6% |
| 4 | - | 10% | 8% | - | 72% | 2% | - | 8% | - | - |
| 5 | 8% | 4% | 4% | 8% | - | 60% | - | 4% | 12% | - |
| 6 | - | - | 1.9% | 7.5% | - | 1.9% | 77.4% | - | - | 11.3% |
| 7 | 2% | - | 4% | - | 16% | 14% | - | 56% | 8% | - |
| 8 | - | - | 10% | - | - | 15% | - | - | 75% | - |
| 9 | - | - | - | 3.8% | - | 3.8% | 11.5% | - | - | 80.8% |

Figure 4.3. Distribution of Inferred Keys in Touchlogger.

smartphone during tap events is utilized. To measure the change of the external force, they calculated 2-norm of acceleration vector by the following formula:

$$SqSum = |A|^2 = A_x^2 + A_y^2 + A_z^2.$$

In the training phase, a user holds the smartphone with her left hand and taps the touchscreen using the forefinger of her right hand. The motions of smartphone during one tap event are observed. The authors observed that the unique $\searrow\nearrow$ pattern of SqSum in the *Action_Down* phase is the key to tap event detection. They also described time duration of tap event as a SigWin parameter. Then, they extracted five features from each tap event within SigWin.

In logging mode, Taplogger cannot receive touch event information from the touchscreen. They tried to detect tap event by the help of features obtained in training phase. For example, Taplogger monitors the accelerometer data and tries to find peak value in SigWin called as checkpoint. According to this value, other 4 features are calculated. Then, it checks if the extracted features are within the intervals determined in training phase. If calculated features are in the determined intervals, Taplogger states that there is a tap event.

To test Taplogger, the tester students enter approximately 150 numbers while Taplogger runs in the background. Actually, detection rate is vary from one user to another. They test Taplogger on three users. While, detection rate is 99.3% for one user, it is 70.4% for another user.

For tap position inference, Taplogger also uses orientation sensor data. The azimuth angle is ignored and the numbers are detected using the change of roll and pitch angles. The authors determined new features according to gesture change of smartphone. Similar to tap detection, they tested tap position inference accuracy of Taplogger, results are shown in Figure 4.4.

| Layout of Number Pad | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
| * | 0 | # |

| Coverage rate with Top 1 ranked label | | |
|---|---|---|
| 0.2759 | 0.4643 | 0.5185 |
| 0.4138 | 0.1200 | 0.3333 |
| 0.2069 | 0.1250 | 0.2500 |
| 0.4348 | 0.3462 | 0.8750 |

| Coverage rate with Top 1 & 2 ranked labels | | |
|---|---|---|
| 0.7931 | 0.75 | 0.7037 |
| 0.6897 | 0.4400 | 0.6061 |
| 0.4483 | 0.2917 | 0.6250 |
| 0.6087 | 0.4615 | 0.9583 |

| Coverage rate with Top 1 & 2 & 3 ranked labels | | |
|---|---|---|
| 0.9310 | 0.8214 | 0.9259 |
| 0.8621 | 0.7200 | 0.9091 |
| 0.6897 | 0.5833 | 0.8333 |
| 0.6522 | 0.6154 | 0.9583 |

| Coverage rate with Top 1 & 2 & 3 & 4 ranked labels | | |
|---|---|---|
| 0.9310 | 0.9286 | 0.9259 |
| 0.9655 | 0.8400 | 0.9394 |
| 0.8966 | 0.6250 | 1.0 |
| 0.8261 | 0.7692 | 1.0 |

Figure 4.4. Detection Results of Tap Position Inference in Taplogger.

In another study [2], the authors used accelerometer data to predict user input as required to unlock smartphone such as password or pattern. They collected data from 24 users in controlled and uncontrolled settings, then they extracted features for accelerometer readings and analysed them by using different techniques. In their experiments, they randomly selected 50 PINs and patterns, their model can predict the PIN entered 43% and pattern drawn 73% of the time within 5 guesses.

In data collection phase, they had 24 students. They randomly chose 50 patterns from 389,112 possible patterns and 50 PINs from 10,000 possible PINs. They used 24 students, 12 students entered all PINs a total of 5 times and other 12 students entered all patterns a total of 5 times. During the experiment, they recorded accelerometer measurements and timing of touch events.

To analysis accelerometer measurements, they extracted 774 different features. These features include standard statistics, polynomial fit techniques and Fast Fourier Transform (FFT). Firstly, they used three normalization techniques for accelerometer data namely, mean, linear and quadratic normalization. The first set of features consists of standard statistics like the root mean square, mean, standard deviation, variance, maximum and minimum. The second set of feature is obtained by fitting 3-degree polynomial to accelerometer data in each dimension. They also computed standard statistics for fitted curves. Therefore, coefficients and statistics of polynomial curves are the second set of features. They used FFT for the third set of features.

Totally, they extracted 86 features for each dimension and for each normalization so $3x3x86 = 774$ features were obtained.

In machine learning classification they modeled two scenarios. In the first part, they assumed that the attacker has sufficient samples for unknown input. They runned multi-class logistic regression model on feature vectors, to do this they used `LIBLINEAR` implementation. The model can predict patterns with 40% accuracy, PINs with 18% accuracy. The model performs better when the number of prediction increases. For instance, the pattern can be predicted with 73% accuracy and PIN can be predicted with 43% accuracy on the fifth prediction.

In the second part, they used Hidden Markov Models, the conditional probabilities of single digit or swipes are taken into account. In this case, patterns can be detected with 26% accuracy and PINs 40% accuracy at 20 guesses but this is also for single user on single device. When they train all users, accuracy rates greatly depressed.

In addition to above studies, there are also other ones. For instance, in [34] the authors proposed ACCessory and they showed that accelerometer measurements can be used to infer character passwords. Their results are based on four users and 2700 keystrokes, they showed that 6 of the 99 passwords can be detected correctly in a median 4.5 trials but 59 of 99 passwords are cracked in approximately $2^{15}$ median trials. They stated that a brute force attack on a 6 character password takes approximately $2^{28}$ trials on average. Therefore, they concluded that accelerometer data can be used to reduce the search area for text entered on smartphones.

The authors try to infer user input by considering timings between two keystrokes in [35]. In [36], the accelerometer data and magnetometer data are used together to develop model for detecting tap position on smartphone touchscreen.

# 5. PROPOSED METHOD FOR PIN INFERENCE

In this thesis, we mainly focused on accelerometer sensor-based side-channels. Studies show that accelerometer readings can be used to infer users' input such as PIN required to unlock smartphones. For this purpose, we handle the study [2] in detail. We take this study as a refence point because, it uses only accelerometer sensor to infer user inputs. Furthermore, they have large data in training set, so their experiment is more realistic compared to other studies, which use data from only 3 or 4 users. In [2], the authors use only accelerometer measurements to infer PIN or pattern entered by user. Actually, we implement the main part of the study where it is assumed that the attacker has sufficient samples of the unknown inputs. Firstly, we developed an application that records accelerometer sensor data during users enter PIN via soft keyboard. Then, we collected data from 12 users. We extracted the same features in the reference study and calculated success rate for PIN inference by using machine learning classification. During the implementation, we noticed that the number of features is very high and some features have no or very small contribution to success rate. Therefore, we proposed the new feature set and showed how this new set affects the success rate of the study.

## 5.1. Application Development

In order to collect data, we developed an Android application by using Android Studio 1.1.0. This application logs accelerometer data in x, y and z directions while users enter numbers via a soft number keyboard. The user interface of developed application is shown in Figure 5.1. Firstly, user enters his age and presses the button `"Kayda Başla"`. Actually, user age is not necessary for PIN inference but it is used for another study in this thesis. Then, application starts to record the accelerometer data to the buffer and when `"Kaydı Bitir"` button is pressed, the application stops recording and saves the recorded data to the internal memory of the smartphone. Then, this data is transferred to PC via e-mail or USB for further analysis.

Application saves data in .txt format and txt file includes 5 columns. The first column is the timestamp information that is the time in nanosecond at which the sensor event occurred. The next 3 columns are accelerometer readings in x, y and z directions, respectively. The last column gives information about touch events like $Action_Down$, $Action_Move$, $Action_Up$. We define three touch states in application. When user touches the screen this is $Action_Down$ and $touch_state = 1$. $Action_Up$ means user removes his finger from touchscreen i.e, touching has finished, in this case $touch_state = -1$. Between $Action_Down$ and $Action_Up$, there is $Action_Move$ phase and $touch_state = 2$. By looking touch state information we determine tap locations in accelerometer readings.

Figure 5.1. The User Interface of Developed Sensor Application.

In the analysis of sensor readings, the sampling rate of the sensor data is critical. The sampling rate depends on the smartphone used and sensor delay assigned in application code. Actually, different sampling frequencies can be assigned to read data in Android. Android offers four levels namely sensor delay $UI$, $Normal$, $Game$ and $Fastest$. They can be chosen according to purpose of the process. For example, $UI$ rate is suitable for the user interface, $Normal$ rate is for screen orientation changes, $Game$ is used for games and $Fastest$ gets sensor data as fast as possible. In the first steps, our sensor application has sampling rate of 7 Hz, which is very low to process

accelerometer data. When we increase the sampling rate, our application has buffering problem at the beginning and it crashes in a short time. Therefore, we modify our code and finally obtain sampling rate of 100 Hz in smartphones like Samsung Galaxy S3 and Galaxy S4. Application development phase actually was very time consuming for us because we do not have any experince about Android Studio and available applications do not have enough sampling rate or touch event information. Because of this, we learned basics of application development and Android sensor events and finally developed our own application for data collection.

## 5.2. Data Acquisition

In [2], 50 PINs are used in experiment but we determine 10 PINs to facilitate data collection and implement techniques in [2] with these 10 PINs. There are 12 volunteer users whose age between 22 and 30 in our experiment. Note that 10 PINs are determined randomly and they are not users' real PINs. While collecting the data, the users hold the smartphone with their left hands and touch to the screen with their right forefingers. We use Samsung Galaxy S3 and S4 smartphones during experiment and they have 100 Hz sampling rate.

*PINs: 1234, 1470, 1937, 2469, 2580, 3574, 3578, 3690, 5555, 7896*

Each user enters 10 PINs o total of 5 times while sitting. We ignore accelerometer data if the user does not enter the PIN correctly. When one user starts to enter one PIN, instead of stopping application after each PIN entry, he starts the application, enters a PIN five times in series and stops the application. Therefore, the application saves the readings into log file which consist of five successive PIN entries.. Finally, we have 60 accelerometer readings for each PIN.

## 5.3. Feature Extraction and Classification

Before starting feature extraction, we make data obtained from users ready to process on Matlab. We have 60 txt files and each file consists of five entries of the same

PIN. We divide data in one txt file into five parts such that each part corresponds to one PIN entry. Since we also record touch events when logging accelerometer data, we know start and end timing of each PIN entry. For one PIN entry, we also take accelerometer data 50 ms before start of the first digit and 50 ms after the last digit of the PIN. Since our sampling rate is 100 Hz, 50 ms data corresponds to 5 samples.

First of all, we use same features in the reference study. Therefore, we extract 86 features for three dimensions and three normalization techniques so $3 \times 3 \times 86 = 774$ features are obtained totally. Table 5.1 shows the list of 86 features. For classification of features, we use multi-class logistic regression model by using LIBLINEAR implementation [37].

Table 5.1. Feature Set Used in PIN Inference.

| **Feature** | **# of Features** | **Description** |
|---|---|---|
| Stats | 6 | Root mean square, mean, standard deviation, variance, max and min |
| 3D-Poly-Coefs | 4 | Coefficients of 3-degree polynomial fit |
| 3D-Poly-Stats | 6 | Stats for polynomial reconstruction curve |
| iFFT-Poly | 35 | The inverse FFT of polynomial reconstruction curve |
| iFFT-Acc | 35 | The inverse FFT of accelerometer data |

`LIBLINEAR` is a simple and practical open source library which is used for large-scale linear classification. It supports linear Support Vector Machines (SVM) and Logistic Regression (LR). Experiments show that `LIBLINEAR` is very efficient especially on large sparse data sets. For instance, it can train a text classification problem with 600,000 samples within several seconds. It also provides probability outputs for Logistic Regression [37].

Logistic regression model determines a discriminating line in feature space for each possible class such that it separates members of one class from members of other classes in the best way. To do this, model learns a weighted sum of feature set. In prediction phase, logistic regression gives predicted PIN class, and this prediction is successful if predicted PIN matches to actual PIN. Model also outputs probabilities for each class in prediction.

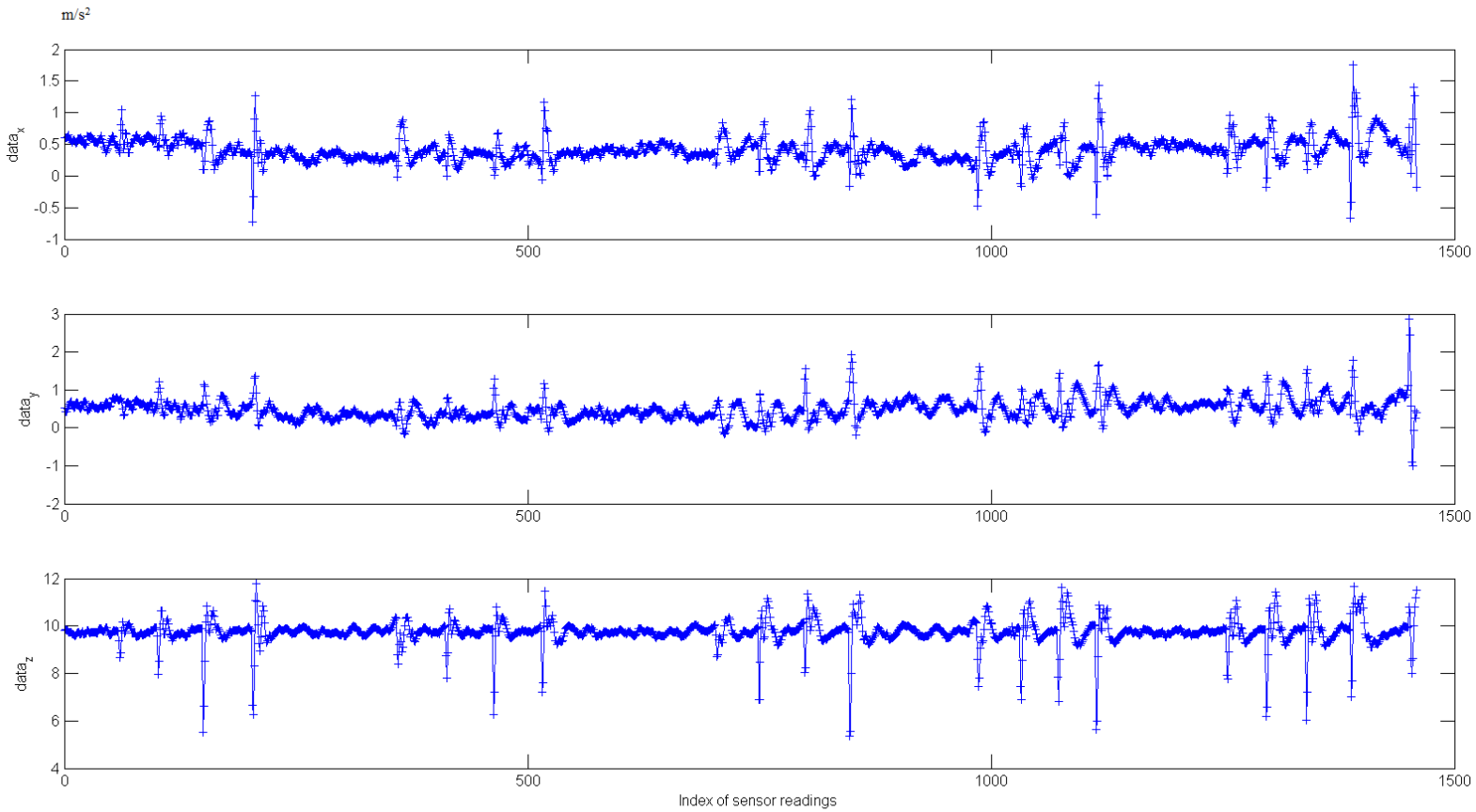



Figure 5.2. Example Sensor Data for 1234 PIN Entry.

For testing the success rate of logistic regression model, we use 5-fold cross validation. Firstly, data set is randomly divided into 5 parts in this technique. Then, cross validation runs such that it uses four parts of data for training and other part for testing. In the next step, it uses another part for testing and remaining part for training. This process continues until all parts are used for testing and training purpose. Therefore, this runs five time and gives five prediction accuracies. Then, these five accuracies are averaged to obtain final success rate of cross validation.

We obtain 40.2% prediction accuracy for 10 PINs by using logistic regression and 5-cross validation. In the study [2], the prediction accuracy was 18% on the first guess for 50 PINs.

### 5.4. Proposed New Feature Set

In this section, we detaily analyse all 774 features in the study [2]. We show that some features are chosen badly so they do not contribute prediction accuracy. Therefore, we proposed new feature set for PIN inference.

First of all, we analyse FFT size used in feature extraction. They state that there is a large variance in the number of samples for each accelerometer measurement. Therefore, they compute $real(\mathcal{F}_m^{-1}(\mathcal{F}_m(A^d)))$.

Note that the Discrete Fourier Transform can be defined by the following formula:

$$F[n] = \sum_{k=0}^{N-1} f[k] e^{-j2\frac{2\pi}{N}nk} \quad n = 0, \ldots, N-1,$$

and the inverse DFT can be expressed as:

$$f[k] = \frac{1}{N} \sum_{n=0}^{N-1} F[n] e^{+j2\frac{2\pi}{N}nk}.$$

However, we can specify DFT length such that it is not equal to the length of the input vector. In this case, one of the following operation occurs: zero-padding, truncating, or modulo-$m$ data wrapping where $m$ is DFT length. When the input length, $n$, is greater than the DFT length $m$, we may see magnitude increases in our inverse DFT output. This is because of the fact that modulo-$m$ data wrapping occurs to preserve all available input samples in inverse transform.

By setting DFT length $m$ smaller than input length $n$, we can maintain general shape of the original signal but set the time domain signal to $m$ samples. This is useful if we have various time domain signals and there is a variance in their length.

By using this fact, they first encode the signal using $m = 35$ complex frequency basis, then reconstruct the original signal. This operation preserves general shape of the signal but normalize the signal to $m$ samples in time domain and also eliminates noisy high frequency components. They test various $m$ values and stated that small $m$ values cannot retain shape of original signal while larger value of $m$ preserves too much variance due to zero padding so 35 is optimum value for $m$.

However, we also test various $m$ values and consider their effects on prediction accuracy, results are shown in Table 5.2. We find that accuracy is maximized when $m = 40$. We think that this is because since we have sampling rate of 100 Hz, there are more samples for each PIN entry in our experiment. Sampling rate in [2] was about 76 Hz.

Secondly, we analyse data on x, y, z dimensions and their effects on prediction accuracy. Since we multiple the extracted features by 3 for three dimensions, elimination of any dimension could decrease the number of features considerably. We noticed that data on z dimension contributes less to prediction accuracy compared to data on x and y dimensions. Actually, this does not mean that there is no data on z dimension, there is data but this data cannot cause discrimination about PIN entries. We investigated that data on z axis strongly depends on the force of typing finger and resistance of holding hand but these factors generally cause shift of the smartphone. On the other

Table 5.2. The Effects of FFT Size on Accuracy Rate of PIN Inference.

| **FFT Size** | **# of Features** | **Accuracy Rate** |
|---|---|---|
| $m = 10$ | 324 | 35.0% |
| $m = 20$ | 504 | 36.8% |
| $m = 25$ | 594 | 38.2% |
| $m = 30$ | 684 | 39.2% |
| $m = 35$ | 720 | 40.2% |
| **m=40** | **864** | **41.2%** |
| $m = 45$ | 954 | 36.2% |
| $m = 50$ | 1044 | 39.3% |
| $m = 60$ | 1224 | 39.4% |
| $m = 80$ | 1584 | 35.3% |
| $m = 100$ | 1944 | 36.3% |
| $m = 120$ | 2034 | 34.0% |

hand, typing different numbers on the soft keyboard causes different rotations of the smartphone. Therefore, we can ignore z dimension and use data on x and y dimensions. Remember that the authors also ignore the azimuth angle for pin inference in [28, 29]. Ignoring data on z axis causes 288 decrease in the number of features. We also test different combinations when $m = 40$, the results are in Table 5.3.

Table 5.3. The Effects of Data Dimensions on Accuracy Rate of PIN Inference.

| **Data Dimensions** | **# of Features** | **Accuracy Rate** |
|---|---|---|
| x,y,z dimensions | 864 | 41.2% |
| **x,y dimensions** | **576** | **40.2%** |
| y,z dimensions | 576 | 36.2% |
| x,z dimensions | 576 | 35.5% |
| x dimension | 258 | 34.5% |
| y dimension | 258 | 29.3% |
| z dimension | 258 | 27.9% |
| $\|A\|^2 = A_x^2 + A_y^2 + A_z^2$ | 258 | 28.6% |

They use polynomial fitting technique to obtain the second set of features. They fit 3-degree polynomial to accelerometer data in each dimension. Then, four coefficients of polynomial are used as a feature. They also compute standard statistics for fitted curves, so six features are also added for each dimension and each normalization. However, we realize that 3-degree polynomial cannot fit to accelerometer data goodly.

Therefore, we also test higher degree polynomial fits and calculate their accuracy rates. Results show that it is unnecessary to use featured derived from poly-fit techniques. The authors also use inverse Discrete Fourier Transform (DFT) of the polynomial fit curve as a feature. Therefore, ignoring parameters from polynomial fitting causes reduction of 300 features.

Table 5.4. The Effects of Polynomial Fit on Accuracy Rate of PIN Inference.

| **Polynomial Fit Features** | **# of Features** | **Accuracy Rate** |
|---|---|---|
| 3-Degree Polynomial Fit | 576 | 40.2% |
| 10-Degree Polynomial Fit | 618 | 40.3% |
| 20-Degree Polynomial Fit | 678 | 40.6% |
| **No Poylnomial Fit** | **276** | **40.1%** |

The authors also apply three different normalization techniques namely, mean, linear and quadratic normalization. We apply each normalization technique separately and monitor changes in accuracy rate. We see that there is no need to apply three normalization techniques together and quadratic normalization gives better results. Finally, we also remove statistical parameters on feature set and this causes accuracy rate to decrease from 40.1% to 38.9%. Now, feature set contains only 80 features, iFFT of data with quadratic normalization. While the number of features decreases from 774 to 80, we have success rate of 38,4% instead of 40,2%. The final feature set includes data on x and y dimensions, quadratic normalization and iFFT of FFT with $m = 40$.

Table 5.5. The Effects of Normalization on Accuracy Rate of PIN Inference.

| **Normalization** | **# of Features** | **Accuracy Rate** |
|---|---|---|
| Mean, linear and quadratic normalization | 240 | 38.9% |
| Mean normalization | 80 | 36.3% |
| Linear normalization | 80 | 36.9% |
| **Quadratic normalization** | **80** | **38.4%** |

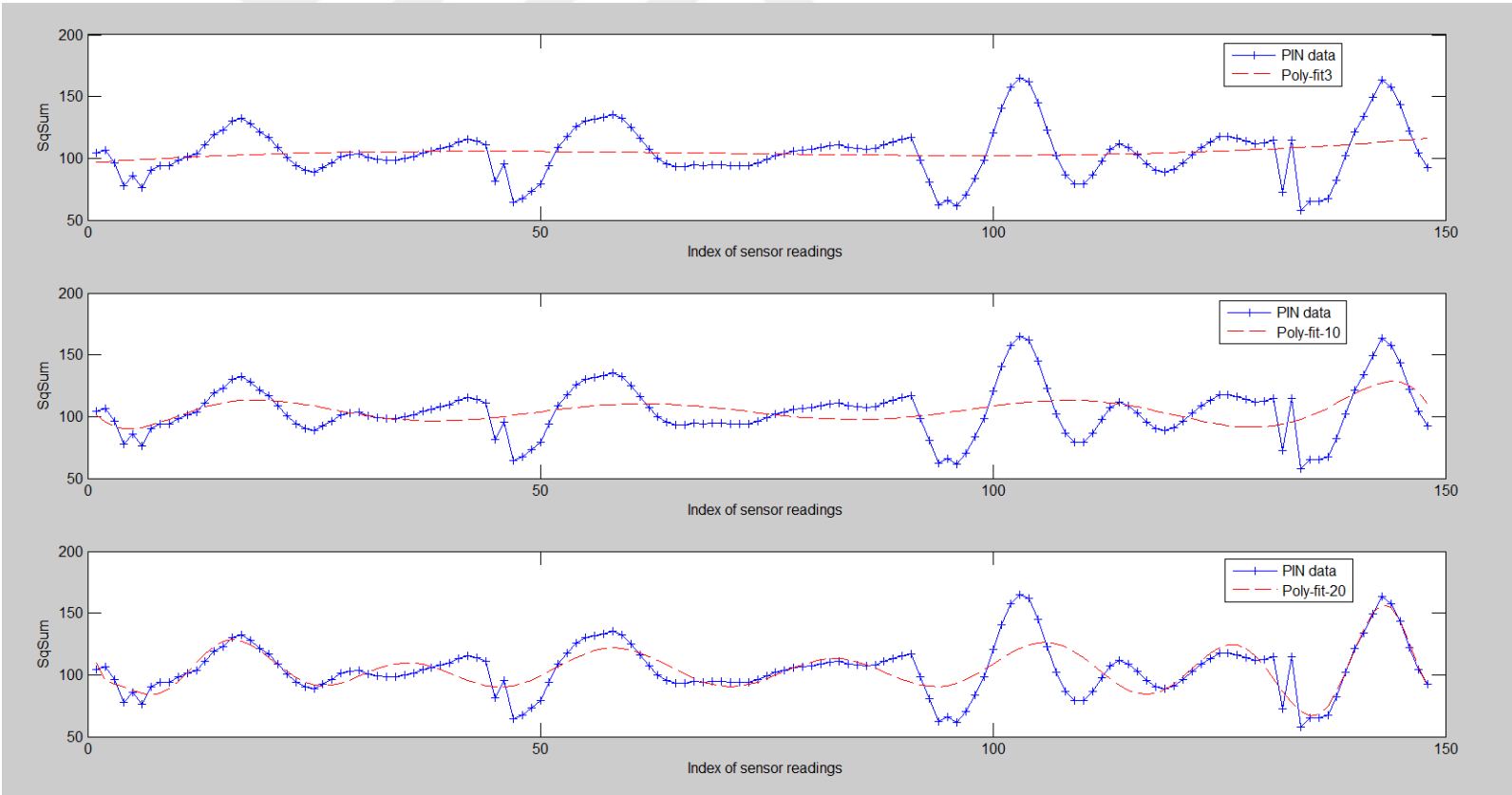



Figure 5.3. 3, 10 and 20-Degree Polynomial Fit Examples.

Then, we apply Discrete Cosine Transform (DCT) to our last feature set. We again test different transform size, and conclude that DCT gives better results than FFT. We obtain accuracy rate of 42,7% with 50 features. Actually, FFT and DCT perform similar operations. They both decompose a discrete time signals into sum of basis functions. The difference between is that the FFT uses complex exponential functions as a basis while the DCT uses real-valued cosine functions.As we can see, DCT can approximate signal well with fewer coefficients. In other words, DCT representation of original signal tends to have more of its energy concentrated in fewer coefficients compared to FFT representation. Actually, we expect this result because the DCT is intensively used in lossy data compression applications like JPEG image format.

Table 5.6. The Effects of DCT on Accuracy Rate of PIN Inference.

| **Discrete Cosine Transform** | **# of Features** | **Accuracy Rate** |
|---|---|---|
| $transformsize = 20$ | 40 | 40.1% |
| **transform size=25** | **50** | **42.7%** |
| $transformsize = 30$ | 60 | 42.4% |
| $transformsize = 35$ | 70 | 43.0% |
| $transformsize = 40$ | 80 | 42.0% |
| $transformsize = 45$ | 90 | 41.5% |

Table 5.7. Accuracy Results of PIN Inference for 3 Classification Methods.

| **Dimension Reduction** | **Classification Method** | **Accuracy Rate** |
|---|---|---|
| none, 50 features | Logistic Regression | 43.4% |
| none, 50 features | k-NN with k=1 | 45.6% |
| none, 50 features | RandomForest Tree | 46.9% |
| | | |
| PCA, 33 features | Logistic Regression | 41.9% |
| **PCA, 33 features** | **k-NN with k=1** | **42.2%** |
| PCA, 33 features | RandomForest Tree | 42.2% |

In addition to `LIBLINEAR`, we also use `WEKA` (Waikato Environment for Knowledge Analysis) machine learning tool for feature selection and classification. `WEKA` consists of various machine learning algorithms such as pre-processing, classification, regression and clustering. `WEKA` has a graphical interface namely Explorer. All algorithms can

be run via Explorer easily. We apply different classification algorithms to our data set with 50 features. Accuracy results of three applied classification algorithms are shown in Table 5.7. RandomForest Tree classification gives the best accuracy rate of 46.9%.

We also use Principal Component Analysis (PCA) for our data in pre-processing step in order to reduce the number of features. Nearest Neighbor algoritm gives the best accuracy rate with PCA, 44.2%.

# 6. PROPOSED METHOD FOR USER AGE INTERVAL IDENTIFICATION

In the fourth chapter, we mention about how motion sensors of smartphones can be used as a side-channel. There are various studies that try to infer user's input on touchscreen by using different sensors. For example, the authors use accelerometer and orientation data to detect user input in [28, 29]. In [35], timing between two keystrokes is used to infer user's input. The accelerometer and magnetometer data are used together to detect tap position on smartphone touchscreen in [36]. In the previous chapter we discuss one of these studies, and try to infer user's PIN entry by using accelerometer sensor readings. However, we use accelerometer sensor data for different goal in this chapter. Actually, we analyse tap fingerprints in accelerometer sensor data and extract features for tap events. Then, we see that these features can give information about users' age interval so we try to determine whether the user is child or adult by looking features we extracted. To the best of our knowledge, this kind of analysis has not been studied yet. We know that this information can be combined with malicious activities such that if attackers know that the user touching the screen is child, they obtain extra information for their malicious activities.

## 6.1. Data Acquisition

For data acquisition, we use the same application in the previous chapter. For PIN inference, there are 12 volunteer users whose age between 22 and 30 in our experiment. However, we expand our user group for this study. We take data from 50 child users whose age between 3 and 11, and 100 adult users whose age between 12 to 51. We went to a kindergarten to take data from child users several times, also took data from children in our families. We took data from our friends, colleagues and families for adult user data. Data collection phase, especially for child users, was very time consuming for us because it was so difficult to show children how they use application properly.

While collecting the data, the users hold the smartphone with their left hands and touch to the screen randomly with their right forefingers while sitting. We use Samsung Galaxy S3, S4 and other smartphones during experiment and they have 100 Hz sampling rate. Firstly, user enters his age and presses the button `"Kayda Başla"`. User taps different positions on the screen approximately one minute. Then, he presses the button `"Kaydı Bitir"` to stop the application. The application stops recording accelerometer data and saves the recorded data to the internal memory of the smartphone.

## 6.2. Feature Extraction and Classification

We import txt files to our PC and make data obtained from users ready to process on Matlab. Each txt file consists of timestamp, accelerometer readings in x, y, z dimensions and touch event information. By the help of touch events, we know tap locations in accelerometer readings. While we analyse accelerometer readings, we notice that there is unique pattern in tap locations and this is fingerprint of tap events. Actually, this change in accelerometer data is due to the external force on the touchscreen. Therefore, we use AccSum term to measure the change of external force in similar way to [29, 36]. AccSum is 2-norm of acceleration vector and represented by the following formula:

$$AccSum = |A|^2 = A_x^2 + A_y^2 + A_z^2$$

where $A_x$, $A_y$, $A_z$ are acceleration values in x, y, z dimensions, respectively. AccSum is directly proportional to $|F|^2$. AccSum is zero when smartphone is dropping freely and is equal to the square of gravity when smartphone is face up in hand. Example AccSum values for child and adult user are shown in Figure 6.1 and 6.3, respectively.

The fingerprint of tap event is shown in Figure 6.2 . The motions of smartphone during the tap event can be expressed as three consecutive phases: *Action_Down*, *Action_Hold* and *Action_Up*. When the user taps on the touchscreen, the smartphone will move downward and this is called as *Action_Down* phase. When the *Action_Down*

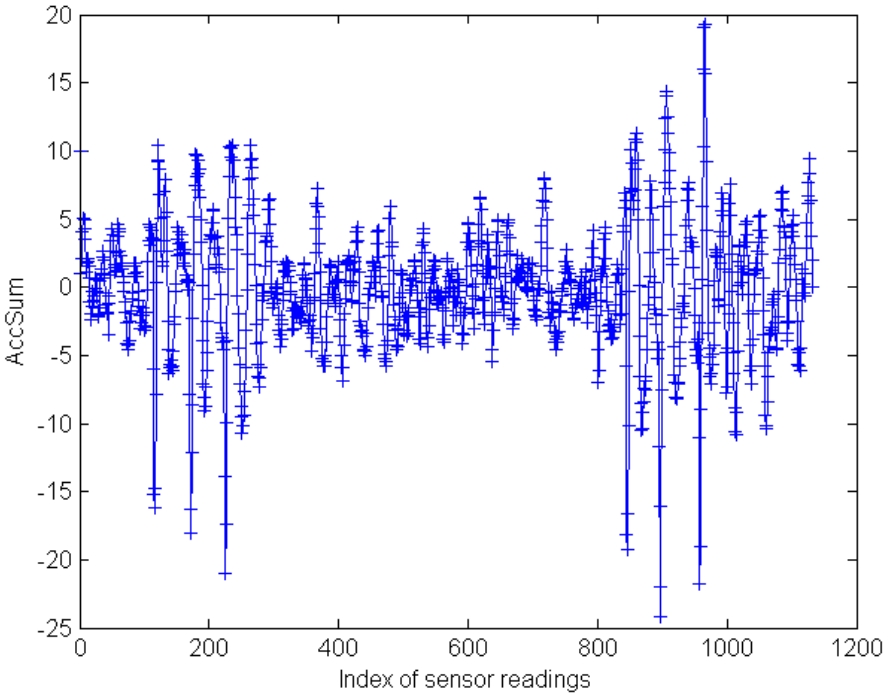



Figure 6.1. Example AccSum for Child User.

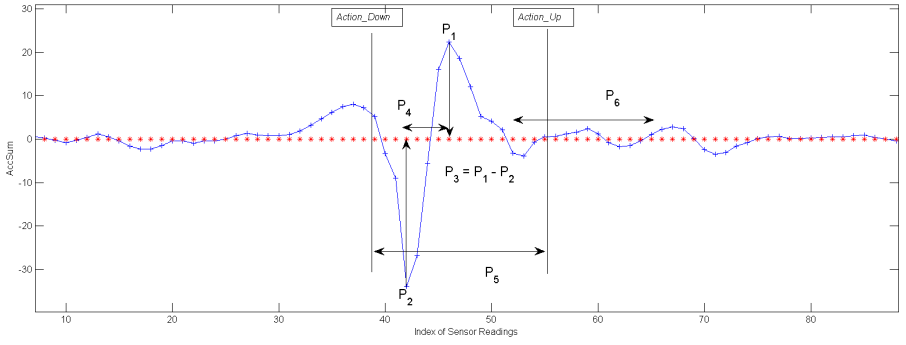



Figure 6.2. The Fingerprint of Tap Event.

is over; the smartphone will stop, this step is called *Action_Hold* phase. Then, the user lifts his finger and the hand holding the smartphone will cause the smartphone back to its beginning position, *Action_Up* phase.

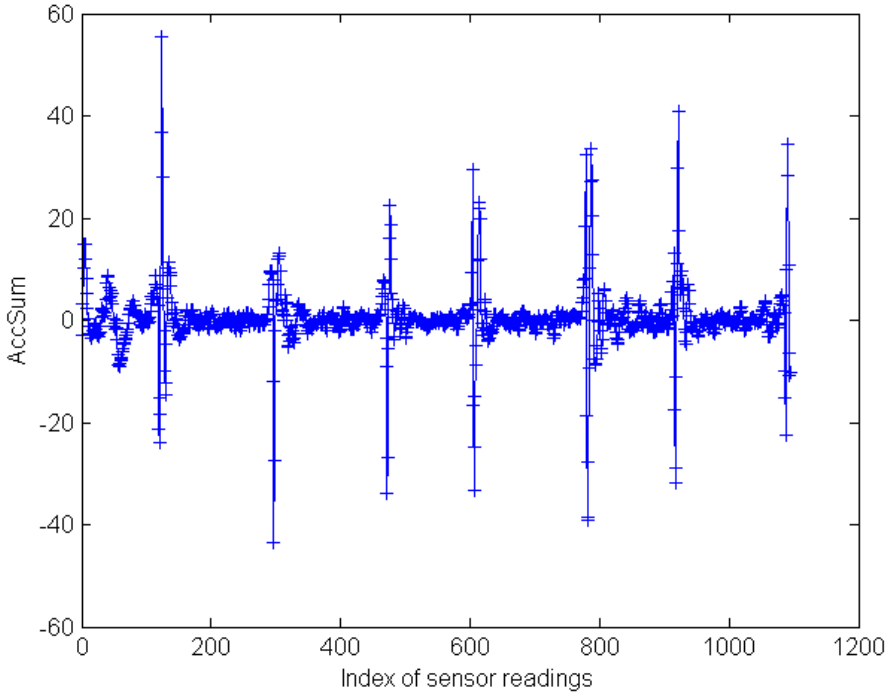



Figure 6.3. Example AccSum for Adult User.

Similar to other studies [29, 36], we firstly extract 6 features for each tap event. However, we find the base value which is starting AccSum value of sensor readings before feature extraction. Since, AccSum is highly affected from gravity force, we should eliminate this effect in the beginning. To do this, we sort AccSum values, and then take the average of ten percent samples in the middle of sorted AccSum. We subtract base value from entire AccSum in order to see effects of external force more reliably. Then, we extract the following features.

(i) $P_1$: The peak value of AccSum at the end of *Action_Down* phase.

(ii) $P_2$: The minimum reading of AccSum in *Action_Down* phase.

(iii) $P_3$: Difference between $P_1$ and $P_2$ values.

(iv) $P_4$: Difference between $P_1$ and $P_2$ in terms of sample index.

(v) $P_5$: Time difference between *Action_Down* and *Action_Up* events.

(vi) $P_6$: Variance of AccSum after *Action_Up* phase to settle the original position.

Actually, $P_1$ measures the magnitude of tapping finger, $P_2$ measures the magnitude of force from the hand holding the smartphone, $P_3$ and $P_4$ measure the change rate, $P_5$ determines tap time and $P_6$ measures the fluctuation. We analyse that for child users, magnitude of tapping finger is weaker compared to adult users. Similarly, adult users hold smartphone more powerfully so reactions due to their holding hands are more strong compared to children. Therefore, we expect lower $P_2$ values and higher $P_1$ values for adult users. In addition, our observations show that children touch on the screen longer than adult users, so we expect higher $P_4$ and $P_5$ values for children. Furthermore, we observe that after *Action_Up* phase, it takes more times for AccSum of children to return its original position. This is because children's holding hands are weak and settle time is higher compared to adults. Therefore, we add $P_6$ parameter as a sixth feature.

In classification, we use the same model in the previous chapter. Actually, we perform multi-class logistic regression by using `LIBLINEAR` implementation [37]. For testing the success rate of classification model, we use 5-fold cross validation. We obtain 81.2% accuracy rate by using logistic regression with 5-fold cross validation. In other word, we can determine a given tap whether belongs to child or adult user with 81.2% accuracy. We also test effects of 6 features on the accuracy rate, the result are show in Figure 6.1. Since $P_3$ is dependent on $P_1$ and $P_2$, it has small contribution on accuracy rate. On the other hand, we obtain 66.9% accuracy rate when we use only $P_1$ and $P_2$ parameters.

In addition to these features, we also try to extract other features. For example, we use polynomial fit techniques to accelerometer readings of tap event. We observe that polynomials with lower degree cannot fit data properly. We observe that 9-degree polynomial fit our data well. However, the accuracy rate decrease to 51.26% when we use coefficient of 9-degree polynomial together with 6 features. We think, this is due to

Table 6.1. The Effects of 6 Features on Accuracy Rate.

| **Features** | **Accuracy Rate** |
|---|---|
| $P_1$, $P_2$, $P_3$, $P_4$, $P_5$, $P_6$ | 81.2% |
| $P_1$, $P_2$, $P_4$, $P_5$, $P_6$ | 80.3% |
| $P_1$, $P_2$, $P_3$, $P_4$, $P_5$ | 75.1% |
| $P_1$, $P_2$, $P_6$ | 70.8% |
| $P_1$, $P_2$, $P_3$ | 68.2 % |
| $P_1$, $P_2$ | 66.9 % |

the fact that polynomial reduces difference between child and adult tap characterictics. Difference between child and adult tap data is mainly because of $P_1$ and $P_2$ parameters. Polynomial fit reduces this difference.

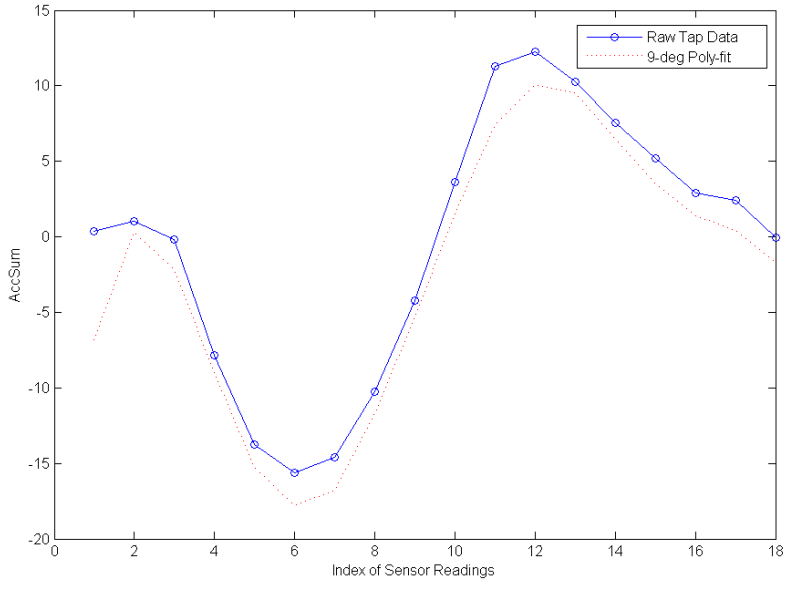



Figure 6.4. Example of 9-degree Polynomial Fit to Tap Data.

After, we apply iFFT of FFT to tap event data. By setting FFT size equal to 10, we fix all tap events to 10 samples so we can use each of these 10 values as a feature. When we use 6 features together with 10 FFT features, we obtain 83.4% accuracy rate. Therefore, we can state that adding 10 features from FFT operation slightly increases the success rate. Example of iFFT is shown in Figure 6.5.

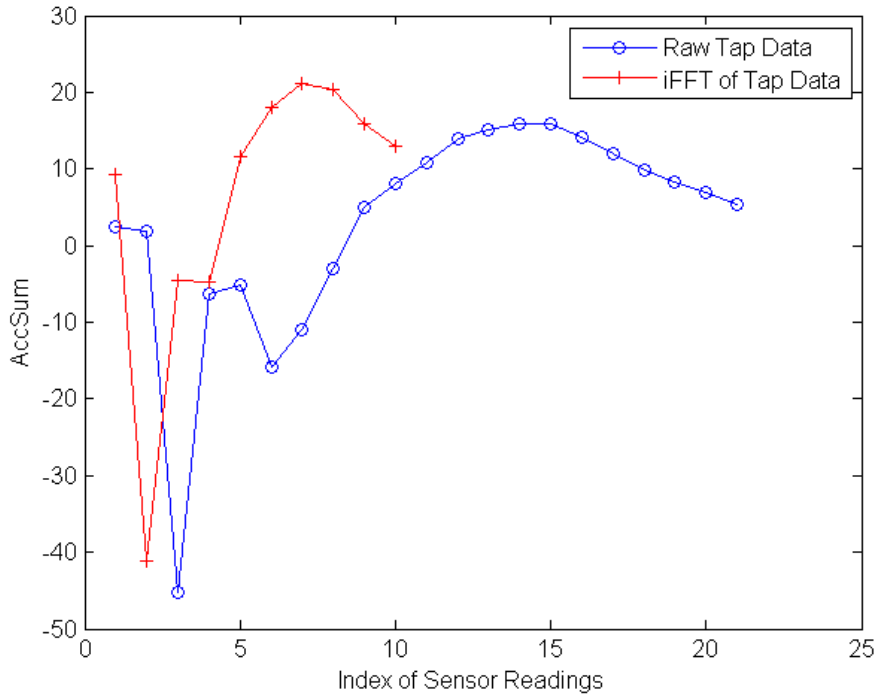



Figure 6.5. Example IFFT of Tap Data.

Instead of FFT, we also apply Discrete Cosine Transform to our data. We again set size of transformation as 10 sample. By using 6 parameters with 10 Cosine Transform parameters, we obtain accuracy rate of 83.6%. However, accuracy rate decreases to 70.0% if we use only 10 cosine transform features. Accuracy rates for different feature set are given in Table 6.2.

Table 6.2. The Effects of iFFT and iDCT on Accuracy Rate.

| **Features** | **# of Features** | **Accuracy Rate** |
|---|---|---|
| $P_1$, $P_2$, $P_3$, $P_4$, $P_5$, $P_6$ | 6 | 81.2% |
| $P_1$, $P_2$, $P_3$, $P_4$, $P_5$, $P_6$ and coefficients of 9-degree polynomial | 16 | 51.2% |
| $P_1$, $P_2$, $P_3$, $P_4$, $P_5$, $P_6$ and features of iFFT transform | 16 | 83.4% |
| $P_1$, $P_2$, $P_3$, $P_4$, $P_5$, $P_6$ and features of iDCT transform | 16 | 83.6% |
| Only iFFT transform | 10 | 69.3% |
| Only iDCT transform | 10 | 70.0% |

In addition to `LIBLINEAR`, we also use `WEKA` for classification. We run 3 different classification algorithms. We also perform PCA for our data in pre-processing step

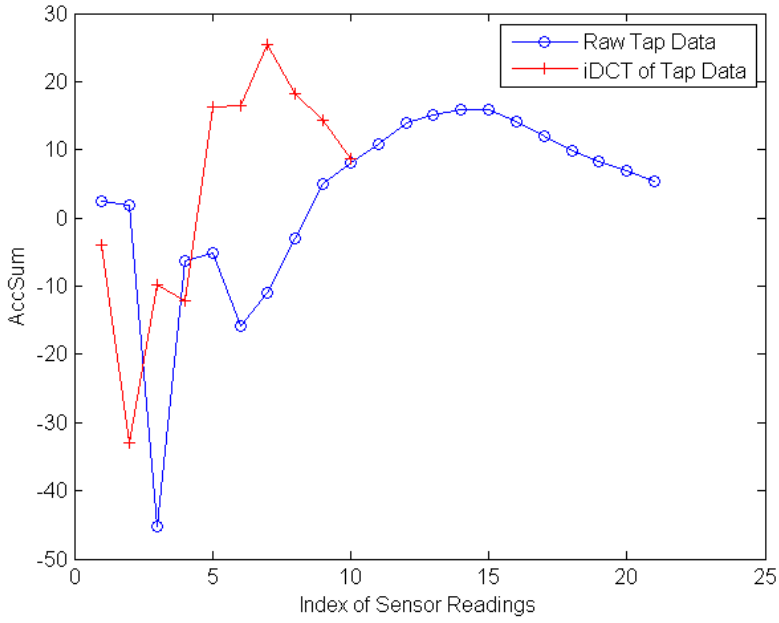



Figure 6.6. Example iDCT of Tap Data.

in order to reduce the number of features. PCA reduces the number of feature to 10 which was 16 in the beginning. Accuracy results for 3 classification models are shown in Table 6.3. As a result, we obtain 87.1% accuracy by 10 - Nearest Neighbor classification method and PCA.

Table 6.3. Accuracy Results for 3 Classification Methods.

| **Dimension Reduction** | **Classification Method** | **Accuracy Rate** |
|---|---|---|
| none, 16 features | Logistic Regression | 83.7% |
| none, 16 features | k-NN with k=10 | 86.9% |
| none, 16 features | k-NN with k=1 | 85.1% |
| none, 16 features | RandomForest Tree | 87.0% |
| | | |
| PCA, 10 features | Logistic Regression | 82.4% |
| **PCA, 10 features** | **k-NN with k=10** | **87.1%** |
| PCA, 10 features | k-NN with k=1 | 84.6% |
| PCA, 10 features | RandomForest Tree | 86.8% |

# 7. CONCLUSION AND FUTURE WORK

In this thesis, first of all, we focused on malwares targeting Android smartphones. We analysed their motivations and behaviors. Then, we spent time on motion sensor-based side-channels in Android smartphones.

We handled the study which use accelerometer sensor data to infer user's PIN entries. In order to implement the steps described in this study, we firstly developed an application that records accelerometer sensor data when users enter PIN. We used Android Studio for development environment. In data acquisition phase, we collected data from 12 volunteers. We determined 10 PINs and each volunteer entered 10 PINs 5 times while sitting. Actually, application development and data collection phases were very time consuming for us. Then we extracted the same features in our reference study, calculated success rate of PIN inference by using machine learning classification methods. We obtained 40.2% prediction accuracy for 10 PINs with 774 features. Then we observed that there are some unnecessary features among these 774 ones. We proposed new feature set by eliminating existing unnecessary features and adding new features. We also applied PCA to our final feature set and tested our accuracy rate with three different classification methods. Finally, we obtained 44.2% prediction accuracy by using only 33 features.

We used `LIBLINEAR` and `WEKA` tools for implementation of machine learning algorithms and pre-processing steps.

Secondly, we proposed the study for determining users' age interval by using accelerometer sensor as a side-channel. We observed that users' tap characteristics on the touchscreen can give information about users' age interval. For this study, we firstly collected tap samples from 50 child users and 100 adult users. This was great effort to form large data set for training phase. We investigated that there are similar studies in which data from only 3 or 4 users are used for training set. Then we extracted features from each tap events which are distinguishing between child and adult users. Then,

we finalized our feature set and tested the success rate by using different classification algorithms. We obtained 87.1% accuracy rate by using k-NN classification algorithm. To the best of our knowledge, determining whether the user is child or adult has not been studied before. Therefore, this is actually our great contribution to the literature because this information may give attackers extra avenues.

We know that this information, knowing whether the user is child or adult, can be combined with malicious activities such that if attackers know that the user touching the screen is child, they obtain extra information for their malicious activities. For example, remember the security enhancement for SMS malware. When an application tries to send SMS message to short number, a pop-up will appear on the screen and ask approval of the user. An attacker can modify their SMS malwares such that SMS will be sent if the attacker detects that the user is child.

In addition, if attackers know that the user is child, they can try to subscribe the user to premium services by bringing various links or commands on the touchscreen. They may also obtain unlock pattern or PIN of the smartphones. While the child user plays a game, a pop can be appear in the game and ask the unlock password or PIN to continue.

On the other hand, this information can be used for restricting some features or programs of the smartphones. For instance, if an application detects that the user is child, it can limit some features like Internet connection. In addition, it can prevent the child user to delete files from smartphones by limiting authorisation of the child user. Therefore, this information can be also used as security enhancement for childrens.

Alternatively, application developers can design their programs such that if the program is used by child users, background of the program can be changed according to children or menus of the program can be more simple to facilitate children usage.

For future works;

- Accuracy rate for PIN inference will improved by using other sensors like orientation sensor together with accelerometer.
- Wavelet transform will be used instead of FFT and DCT.
- The application we developed for data acquisition can also be modified to collect data for larger groups.
- For both PIN inference and user age interval identification, data can be collected for different scenarios like walking and running.
- We used touch events in Android to detect tap positions in sensor readings. Instead of using touch events, algorithm for tap detection can be developed.

# REFERENCES


1. IDC, *Worldwide Smartphone OS Market Share*, 2015, `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`, [Accessed November 2015].

2. Aviv, A. J., B. Sapp, M. Blaze and J. M. Smith, "Practicality of Accelerometer Side Channels on Smartphones", *Proceedings of the 28th Annual Computer Security Applications Conference*, pp. 41–50, ACM, 2012.

3. Schmidt, A.-D., *Detection of Smartphone Malware.*, Ph.D. Thesis, Berlin Institute of Technology, 2011.

4. Haldar, S. and A. Aravind, *Operating Systems*, Pearson Education India, 2010.

5. Jobs, S., "Third Party Applications on the iPhone", *Apple Inc*, 2007.

6. Smith, A., "US Smartphone Use in 2015", *Pew Research Center*, 2015.

7. Aycock, J., *Computer Viruses and Malware*, Vol. 22, Springer Science and Business Media, 2006.

8. Lenny, Z., "Malware: Fighting Malicious Code [M]", *Indiana: Prentice Hall*, pp. 34–45, 2003.

9. Dezfouli, F. N., A. Dehghantanha, R. Mahmod, N. F. B. M. Sani, S. B. Shamsuddin and F. Daryabar, "A Survey on Malware Analysis and Detection Techniques", *International Journal of Advancements in Computing Technology*, Vol. 5, No. 14, p. 42, 2013.

10. Boldt, M. and B. Carlsson, "Privacy-Invasive Software and Preventive Mechanisms", *Systems and Networks Communications, 2006. ICSNC'06. International*

*Conference on*, pp. 21–21, IEEE, 2006.

11. Dunham, K., *Mobile Malware Attacks and Defense*, Syngress, 2008.

12. You, I. and K. Yim, "Malware Obfuscation Techniques: A Brief Survey", *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, pp. 297–300, IEEE, 2010.

13. Apvrille, A., "The Evolution of Mobile Malware", *Computer Fraud and Security*, Vol. 2014, No. 8, pp. 18–20, 2014.

14. Felt, A. P., M. Finifter, E. Chin, S. Hanna and D. Wagner, "A Survey of Mobile Malware in the Wild", *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, pp. 3–14, ACM, 2011.

15. Strazzere, T., "Security Alert: HongTouTou, New Android Trojan, Found in China", *The Lookout Blog*, 2011.

16. Zhou, Y. and X. Jiang, "Dissecting Android Malware: Characterization and Evolution", *Security and Privacy (SP), 2012 IEEE Symposium on*, pp. 95–109, IEEE, 2012.

17. Elish, K. O., D. Yao and B. G. Ryder, "User-Centric Dependence Analysis for Identifying Malicious Mobile Apps", *Workshop on Mobile Security Technologies*, 2012.

18. Chien, E., "Motivations of Recent Android Malware", *Symantec Security Response*, 2011.

19. Dini, G., F. Martinelli, A. Saracino and D. Sgandurra, "MADAM: A Multi-level Anomaly Detector for Android Malware.", *MMM-ACNS*, Vol. 12, pp. 240–253, Springer, 2012.

20. Suarez-Tangil, G., J. E. Tapiador, P. Peris-Lopez and A. Ribagorda, "Evolution,

Detection and Analysis of Malware for Smart Devices", *Communications Surveys and Tutorials, IEEE*, Vol. 16, No. 2, pp. 961–987, 2014.

21. Egele, M., T. Scholte, E. Kirda and C. Kruegel, "A Survey on Automated Dynamic Malware Analysis Techniques and Tools", *ACM Computing Surveys (CSUR)*, Vol. 44, No. 2, p. 6, 2012.

22. La Polla, M., F. Martinelli and D. Sgandurra, "A Survey on Security for Mobile Devices", *Communications Surveys and Tutorials, IEEE*, Vol. 15, No. 1, pp. 446–471, 2013.

23. Alpaydin, E., *Introduction to Machine Learning*, MIT Press, 2014.

24. Witten, I. H. and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2005.

25. Altman, N. S., "An Introduction to Kernel and Nearest-neighbor Nonparametric Regression", *The American Statistician*, Vol. 46, No. 3, pp. 175–185, 1992.

26. Yang, L., D. H. Widyantoro, T. Ioerger and J. Yen, "An Entropy-based Adaptive Genetic Algorithm for Learning Classification Rules", *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, Vol. 2, pp. 790–796, IEEE, 2001.

27. Refaeilzadeh, P., L. Tang and H. Liu, "Cross-validation", *Encyclopedia of Database Systems*, pp. 532–538, Springer, 2009.

28. Cai, L. and H. Chen, "TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion.", *HotSec*, 2011.

29. Xu, Z., K. Bai and S. Zhu, "Taplogger: Inferring User Inputs on Smartphone Touchscreens Using On-board Motion Sensors", *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 113–124, ACM, 2012.

30. De Luca, A., A. Hang, F. Brudy, C. Lindner and H. Hussmann, "Touch Me Once and I Know It's You!: Implicit Authentication Based on Touch Screen Patterns", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 987–996, ACM, 2012.

31. Matsuo, K., F. Okumura, M. Hashimoto, S. Sakazawa and Y. Hatori, "Arm Swing Identification Method with Template Update for Long Term Stability", *Advances in Biometrics*, pp. 211–221, Springer, 2007.

32. Liu, J., L. Zhong, J. Wickramasuriya and V. Vasudevan, "User Evaluation of Lightweight User Authentication with a Single Tri-axis Accelerometer", *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, p. 15, ACM, 2009.

33. Aviv, A. J., *Side Channels Enabled by Smartphone Interaction*, Ph.D. Thesis, Pennsylvania State University, 2012.

34. Owusu, E., J. Han, S. Das, A. Perrig and J. Zhang, "ACCessory: Password Inference Using Accelerometers on Smartphones", *Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications*, p. 9, ACM, 2012.

35. Foo Kune, D. and Y. Kim, "Timing Attacks on Pin Input Devices", *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pp. 678–680, ACM, 2010.

36. Shen, C., S. Pei, T. Yu and X. Guan, "On Motion Sensors as Source for User Input Inference in Smartphones", *Identity, Security and Behavior Analysis (ISBA), 2015 IEEE International Conference on*, pp. 1–6, IEEE, 2015.

37. Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang and C.-J. Lin, "LIBLINEAR: A Library for Large Linear Classification", *The Journal of Machine Learning Research*, Vol. 9, pp. 1871–1874, 2008.