MODELING AND ANALYSIS OF INTENTIONAL AND UNINTENTIONAL SECURITY VULNERABILITIES IN A MOBILE PLATFORM

Mohamed Fazeen Mohamed Issadeen, B.Sc.

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

December 2014

APPROVED:

Ram Dantu, Major Professor
Kathleen M. Swigger, Committee Member
Robert Akl, Committee Member
Rodney D. Nielsen, Committee Member
Barrett R. Bryant, Chair of the Department of Computer Science and Engineering
Coastas Tsatsoulis, Dean of the College of Engineering
Mark Wardell, Dean of the Toulouse Graduate School

Mohamed Issadeen, Mohamed Fazeen. Modeling and Analysis of Intentional and Unintentional Security Vulnerabilities in a Mobile Platform. Doctor of Philosophy (Computer Science and Engineering), December 2014, 149 pp., 29 tables, 51 figures, references, 112 numbered titles.

Mobile phones are one of the essential parts of modern life. Making a phone call is not the main purpose of a smart phone anymore, but merely one of many other features. Online social networking, chatting, short messaging, web browsing, navigating, and photography are some of the other features users enjoy in modern smartphones, most of which are provided by mobile apps. However, with this advancement, many security vulnerabilities have opened up in these devices. Malicious apps are a major threat for modern smartphones. According to Symantec Corp., by the middle of 2013, about 273,000 Android malware apps were identified. It is a complex issue to protect everyday users of mobile devices from the attacks of technologically competent hackers, illegitimate users, trolls, and eavesdroppers.

This dissertation emphasizes the concept of intention identification. Then it looks into ways to utilize this intention identification concept to enforce security in a mobile phone platform. For instance, a battery monitoring app requiring SMS permissions indicates suspicious intention as battery monitoring usually does not need SMS permissions. Intention could be either the user's intention or the intention of an app. These intentions can be identified using their behavior or by using their source code. Regardless of the intention type, identifying it, evaluating it, and taking actions by using it to prevent any malicious intentions are the main goals of this research.

The following four different security vulnerabilities are identified in this research: Malicious apps, spammers and lurkers in social networks, eavesdroppers in phone conversations,

and compromised authentication. These four vulnerabilities are solved by detecting malware applications, identifying malicious users in a social network, enhancing the encryption system of a phone communication, and identifying user activities using electroencephalogram (EEG) for authentication. Each of these solutions are constructed using the idea of intention identification. Furthermore, many of these approaches have utilized different machine learning models.

The malware detection approach performed with an 89% accuracy in detecting the given malware dataset. In addition, the social network user identification model's accuracy was above 90%. The encryption enhancement reduced the mobile CPU usage time by 40%. Finally, the EEG based user activities were identified with an 85% accuracy. Identifying intention and using it to improve mobile phone security are the main contributions of this dissertation.

ACKNOWLEDGMENTS

This dissertation would not have been possible without the help of many people. First, I would like to thank my mentor and advisor Dr. Ram Dantu for his support and guidance. His continuous encouragement and advice has helped me immensely academically. To Dr. Kathy Swigger, Dr. Robert Akl, Dr. Rodney Nielsen, thank you very much for agreeing to be on my Ph.D. committee and giving me valuable advice in this research.

Dr. P. Guturu, Prof. of Elec. Eng., UNT, I cannot thank him enough for helping me publish the paper on social networking which was a major part in this research. Further, I would like to thank Dr. Z. Wang, Assoc. Prof., Comm. Studies, UNT, for helping me in the inception of this research work. To Dr. O. Garcia, Prof. of Elec. Eng., UNT, thank you for helping me to contribute to the 'Information Theoretic' security paper. I would also like to recognise the support given by the National Science Foundation for this research under grants CNS-0627754, CNS-0619871, CNS-0551694, CNS-0751205, CNS-0821736 and CNS-1229700. I would also like to thank all the professors who taught me at UNT and helped me in achieving this.

To Mallory Schier, I sincerely thank you for helping me edit and proofread this dissertation. Your patience and tolerance is greatly appreciated. To all my past and present lab mates, thank you very much for sticking with me and having faith in me. The constant discussions, exchange of ideas and brain storming and valuable feedback have all been instrumental. To my Denton Sri Lankan friends, thank you for making me feel at home. You all have been amazingly supportive.

To my family, without them I would not be here today. My parents have been my supporting pillar and strength throughout my life. They still continue to lookout for me and have supported me in all my decisions. I am grateful for them and I cannot thank them enough. To my sister, thank you for taking care of our parents while I am away from home. Finally, to my ever loving wife Zaynab Wazeer for her unwavering love and support in this journey; her advice, humor and understanding have made my days brighter. I cannot thank her enough for her support and persistence. I would not have achieved this without her tremendous support.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

1.1. Introduction and Background

Would you ever give a briefcase which contained all your health records, address, social security number, all your contacts, your bank information, a transcript of every conversation you have ever written or said, personal photos and videos, and a list of every place you have ever been. . . to a complete stranger wearing a hoodie and sunglasses?

No?

Well, any smartphone contains all the information that was in that briefcase, and more. In essence, a breach of any smartphone is damaging to the owner, and even worse, it is an especially rewarding ”lock” to pick for people who want to use that information to gain an unfair advantage. Based on the available attacks and malware nowadays, phones are also a relatively easy thing to attack. Therefore, protecting a phone is of utmost importance for any individual.

The goal of this dissertation is to address some of the many security vulnerabilities on the mobile platform with the idea of Intention of the agent. These vulnerabilities come in different forms (see Figure 1.1), including:

- Malware on any open app market/store
- Malicious users called trolls
- Eavesdropping
- Illegitimate authentication

1.2. Motivation

As of April 2014, there are 143 million smartphones being used in the U.S. [79]. That is, by January 2014, 90% of American adults owned a cell phone, 58% owned a smartphone. With

The content in this chapter is reproduced from Mohamed Fazeen and Ram Dantu, “Another free app: Does it have the right intentions?”, Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on, July 2014, pp. 282289, with permission from IEEE.

FIGURE 1.1. Mobile phone security vulnerabilities that are discussed in this dissertation.

these phone usage statistics, 68% of the phone users witnessed unwanted sales or marketing calls at least once in their usage. Text messaging wasn't second to the mobile phone calls. Among 79% of cell phone owners used text messaging services and 69% of them claimed that they received an unwanted spam or a text message [82]. With the innovation of smartphones, mobile phone applications, or apps became one of the most popular elements in smartphones. As of June 2014, there were about 1.3 million Android apps enlisted in the Google Play market, 300000 apps were found in the Windows phone store, and 75 billion apps were found in the Apple store. Further, there are about 102,062 million mobile apps downloaded worldwide by the year 2014 [81]. Users are spending more and more time in using their smartphones. From March 2013 to March 2014, smartphone users spent 2 hours and 42 minutes per day on the phone. This is a four minute increase compared to that of previous year [76]. With these ever increasing trend on mobile phone usage, recent years witnessed an explosive 250% growth in the use of Android devices as well. In the first quarter of 2013, for the first time the smartphone shipment outpaced the feature phones [93]. Out of the above, Samsung took a lead over Nokia and Apple, by pumping more Android devices into the market. Consequently, the number of app downloads from the Google Play reached to about 11 billion downloads [59]. During this time consumers have seen a remarkable growth in the Android malware. According to a Mobile Threat Report published by F-Secure [34], Android

was the most heavily targeted mobile operating system in second quarter of 2012, with a 64% increase in Android malware from Q1 to Q2 in 2012. A report published in the NakedSecurity web site by Graham Cluley on June 2012, described that the top five malware app types in the Android platform are Andr/PJApps-C of 63.4%, Andr/BBridge-A of 8.8%, Andr/Generic-S of 6.1%, Andr/BatteryD-A of 4.0%, Andr/DrSheep-A of 2.6%, and other of 15.1% of the malware [23]. The most common malware type is the Andr/PJApps-C, which is a repackaged app [111, 37].

Traditional malware identification is mainly rooted on signature based identification, and these detections can be evaded easily by adopting a transformation or a polymorphic attack. [85][40]. This malware can change its signature slightly such that it is different from the known signature. Then the signature based anti-malware tools cannot detect the new signature, though it is the same old malware that it used to identify. However, if the anti-malware tool is capable of identifying its behavior, no matter how much its signature changes, the tool will be able to filter the apps with malicious intention. Therefore, I believe, identifying the intention of any activity is crucial in determining its maliciousness. This motivated me to find a solution for the malware problem as a contribution to the mobile phone security.

Mobile phones and an online social network are like bread and butter... They are closely attached to each other. Recent statistics indicate that 60% of the time spent on social media are from smart phones [4]. Therefore, security threat in social networks directly affect smart phones. Thus, identifying threat models in social media will be helpful in providing mobile phone security. ”Average users may receive up to 17 ’dangerous’ Tweets per day” as of 2011. Which means, ”3.5billion nasty Tweets are sent every day” by someone or from something [92]. I believe that identifying the intention of such users plays a major role in identifying and defending such behavior in social networks. This motivated me to find a solution for the mis-behaving user identification based on intentions as a contribution to the mobile phone security.

Voice over IP (Voip) conversations (like Skype, Viber) are very popular among mobile users these days. However, many vulnerabilities can be identified in these systems that are affecting the integrity of the mobile phone security. A survey by A. Keromytis discuss about security vulnerabilities in Voip with 245 publications [50]. This survey pointed out 30 different citations on

voip eavesdropping, interception and modification. This implies that eavesdropping is not only a serious threat in such communications, but also that many attack methods exist to breach the communication by hackers. Most of these vulnerabilities are addressed in security research. However, many restrictions apply when enforcing some of the obvious solutions (like encrypting a voice conversation in mobile platform) due to its resource limitations. Having said that, user intention can be used to improve these existing solutions in mobile platform. This motivated me to improve the currently existing phone conversation encryption methods using user intention as a contribution to the mobile phone security.

Generally people don't put secure passwords on their mobile phones. A report published in June 2013, pointed out that, from a sample of 1,656 adult smart phone users, 39% did not take at least the smallest measure to protect their data. "Many failing to implement a simple password protection on their devices. "Of those surveyed, only 31 percent regularly backed up data [46]. Further, people tend to lose their phones. A survey by McAfee studied 439 sample organizations to see how many of their employees lose their smartphones. In a year, 142,706 smartphones were reported lost by these company employees. Out of these missing phones, only 9,298 (7%) were recovered. "13% of the missing smartphones were lost in the workplace, 29% were lost while traveling, and 47% were lost while employees were working away from the office, either at home or hotel rooms [99]. The lost phone may get into another persons hand, and next thing you notice is all your data and belongings compromised. A good user authentication system can save the day at least by protecting the data and other valuable information. Intention of the user can be used to produce a strong authentication system by using owners biometrics like brain waves. This motivated me to find a solution for the user authentication by using intention with brain waves (EEG) as a contribution to mobile phone security.

Intention can be of many forms. This dissertation takes a look into the importance of identifying the intention of an agent and use it to provide security for a mobile system. As mentioned earlier, I discuss four mobile security applications under this intention identification; a mobile app malware identification, a user role identification, context-aware encryption system, and finally EEG based authentication.

1.3. Definition of Intention

Intention, by definition, is a course of actions that a user or an agent plans to follow. Thus, identifying the intention of an agent is critical in determining the purpose of the agent. In the domain of security, identifying the intention will lead to identifying the maliciousness of the agent. This can be utilized to refine the potential threats to a system. Intention identification is modeled in several ways and current theories deal with several questions such as 'What are the knowledge requirements for the identification of the maliciousness of an intention' [94, 18].

Unintentional Activities: Similar to intention, unintentional activities can be defines as the actions that are did not planned to follow or accidental actions. For instance, one can mistakenly pronounce something as a bad word thought it was unintentional. Unintentional activities are also a vulnerability for a system and need to be evaluated and planned properly to provide better security system.

The novelty of this work is to find the intention of an agent and using it to provide mobile security solutions. For instance, the devised algorithm for malware identification finds the intention of an app and uses it to identify its potential risk factor. Since I use a static code analysis approach, this potential malware identification can be performed even before the app is installed on a device. Intuitively, if an app requests to access a certain set of system resources which are unrelated to its intended task, it portrays a malicious intention. For instance, if a calculator app has an intention to perform numerical calculations, but it also requests the permission to access short message service sending (SEND_SMS), this illustrates that it is probably a potential permission misuse or a suspicious malicious activity. Thus, based on these analyses, I try to determine whether an app goes beyond its actual intention to perform any unintended activities.

Following are several different types of intentions defined in this dissertation.

(1) User-intention

(2) Developer-intention

(3) Application-intention (or App-intention)

(4) Task-intention

(5) Alternate-intention

(6) Malicious-intention

(7) Benign-intention

Intentions are addressed in detail in the next chapter.

1.4. Thesis Statement

How do you protect every day users of mobile devices from the attacks of technologically competent hackers, trolls, eavesdroppers, and illegitimate users? My approach to this issue is to protect mobile device users from hackers, trolls, eavesdroppers, and illegitimate users by modeling behavioral patterns and optimizing encryption using the idea of intention of an agent.

1.5. Problem Definition

It is a complex issue to protect every day users of mobile devices from the attacks of technologically competent hackers, illegitimate users, trolls, and eavesdroppers. I observed that there were a lot of attacks on phones via malware and I thought it would be an important topic to address.

Intention identification is very important in determining safety and security of a system. As explained above intention can be of different forms. Different type of intention can aid in providing different kinds of security solutions. For instance, identifying user-intention can be used to determine different types of users in a system and filter out malicious users. On the other hand app-intention can be used to identify malicious applications or malware apps and defend the system against them. The main goal of this dissertation is to introduce the concept of intention, identify the intention and use it in different applications to provide secure solutions on a mobile platform.

1.6. Contributions

For malware detection: I collected statistical information about permission requests of phone applications. I applied machine learning to identify the task-intention of the phone apps. Then I applied probability models to find; what the most probable set of permission requests for each task-intention category is. Then I compared the unknown app's permission requests

with its task-intention group's permission request distribution. My method was able to detect new malware with 89% accuracy, even if it was not known to the system, such as Zero day malware.

For detecting trolls in social network: In one approach, I determined how strong of a friendship bond social network users had on Twitter using their user-intentions such as replying a friend, how soon you reply the friend, etc. Since social network users have various bonds with other users in their network, ranging from acquaintance to close friend, I used fuzzy logic (which means approximate reasoning rather than exact number values). My method successfully used this information to determine different types of users, but especially spammers and lurkers.

In another approach, when above networking information are lacking, I took the time stamps of the posts and graphed their frequency for different time periods. Then I used machine learning models to detect the differences between bots, news broadcasters, and friends. This way the user-intention of bursting tweet messages, send it as a regular user reveled their role in the network. Both of these methods detected trolls with an average of more than 90% accuracy.

When optimizing the encryption of a secure phone conversation: I used real time speech recognition on mobile phones to determine the user-intention of the conversation. If the intention is to convey a sensitive word it will be encrypted. This way, I create a model which encrypted only sensitive words. This method reduced CPU usage time for encryption by 40%. Since the existing data (VOIP) communication model for secure phone conversations (Secure Real-time Transport Protocol, SRTP) does not support my proposed optimization, I implemented a custom protocol within a phone application. Further, this model can be easily remodeled as a system to protect speech conversation from uttering inappropriate words or sensitive information that user not supposed to reveal. It can be achieved by simply identifying a set of predefined restricted words list and block them in realtime when the user has spoken them unintentionally. This unintentional modeling can be useful in protecting the user from embarrassments, as well as some security breaches, such as passwords

mistakenly being pronounced.

To profile user behavior in regard phone authentication: I used EEG (brainwave activity patterns) to identify repeated activities a user performs when using a phone, such as swiping, pinching, tapping. I extracted these features from the EEG experiment and used a machine learning model to distinguish between these activities, with 85% accuracy. This way user intention can be revealed and can be used in mobile phone security such as authentication. One advantage of using EEG in authentication is that it can provide a biometric using EEG for authentication [20]. However, in this work I took the first step towards solving the authentication problem and I present this as future work.

1.7. Dissertation Outline

Chapter 1: This chapter provide an introduction to the concept of this dissertation. Further, it summarizes the problem definition and the rest of the document.

Chapter 2: Intention is a key term in this dissertation. In this chapter, different types of intentions are identified and its importance are discussed.

Chapter 3: This chapter briefly describes the usage of intention identification. Four different problems are explained in this chapter related to two different types of intentions (user-intention and app-intention). These four applications are 1. Malware identification, 2. User role identification, 3. Context-aware encryption, and 4. EEG based behavior identification. Each of these problems are solved using different type of intention identification. The following chapters describe how each problem is solved using the introduced intention identification concepts.

Chapter 4: Task-intention is part of the app-intention. Identifying the task-intention is the first step of malware identification in my approach. This chapter describes on how to determine the task-intention of an android app by using machine learning models. This task-intention identification should not be confused with the malicious-intention identification. Task-intention simply says the main functionality of the application. On the other hand, malicious-intention identification reveals whether the app is malicious or not. However, the task-intention is used to determine if an app is malicious or not.

Chapter 5: Permission requests of Android applications are studied in this chapter. I-shapes are constructed from these permission requests. Constructing the I-shape is crucial in malware detection. This chapter addressed the description of I-shape construction using permission requests.

Chapter 6: As mentioned earlier, app-intention could be used to determine the maliciousness of an application. This chapter explains, how to utilize the app-intention and its sub types to identify malware app samples in an Android mobile system.

Chapter 7: In contrast to app-intention, user-intention is useful in identifying the malicious behavior of a user in a system. It can be utilized in user based systems like, online social networks, to identify different types of users and protect against malicious users. Since, modern smart phones are closely attached to online social networking systems as a direct user terminal, it is important to protect such social network systems, so that not to propagate malicious activities to other mobile users. For instance, spamming in social media directly affect mobile users. This chapter demonstrate how to identifying different types of user roles such as leaders, lurkers, spammers etc. by identifying their user-intentions in a popular online social network (Twitter). These intention are modeled using the behavioral footprint and clues of a user in the social network.

Chapter 8: Security in a mobile-phone-communication is a big challenge in the field. Eavesdropping is one of the major privacy breaches. In June 2013, Edward Snowdens massive classified information leak ignited a huge debate about the privacy in phone communication. Encrypting the communication is one of the solutions. However, providing encryption with existing solutions such as Secure Real-Time Protocol (SRTP) in a mobile phone is expensive as it consumes a lot of mobile resources. However, it is intuitive that unsensitive information does not need to be encrypted with high strength algorithms. Thus, determining the user-intention can retrieve the sensitivity of the spoken information. Based on this intention, the algorithm determines whether to encrypt the portion of that spoken segment or not. This way it can spare many resources while provide a similar strength of security in the conversation. This idea is explained in detail in this chapter.

Chapter 9: User-intention identification its usages can be extended beyond the methods discussed in the previous chapters. In the recent years, wearable sensors drawn a great attention in many disciplines [72, 86]. It is even appealing for many developers to use these devices with smart phones (Ex: Samsung Galaxy Gear with Samsung Phones, Apple watch with iPhone). In this future work section, I predict that Electroencephalogram (EEG) signal monitoring in a smart phone will become a more common practice. Especially, it is more natural to combine EEG with devices like Oculus Rift virtual reality system [68] and Google Glass [56] to produce useful applications. This chapter describes the preliminary experiments I did for identifying different activities of a user on a phone by utilizing EEG, including methodology and results. In this chapter, I discuss on how to utilize EEG in mobile environment to obtain behavioral user-intention and its feasibility.

Chapter 10: The discussion and conclusion of the dissertation is presented in this chapter.

CHAPTER 2

INTENTION AND MOBILE PLATFORM

2.1. Intention and Its Types

As mentioned in the introduction chapter, intention is a course of actions a user or an agent plans to follow. In the context of the mobile platform, an intention can be of many forms. See figure 2.1 for a hierarchical view of these different intention types. Type of the intention as well as the agent is important in addressing many mobile security vulnerabilities addressed in this dissertation.

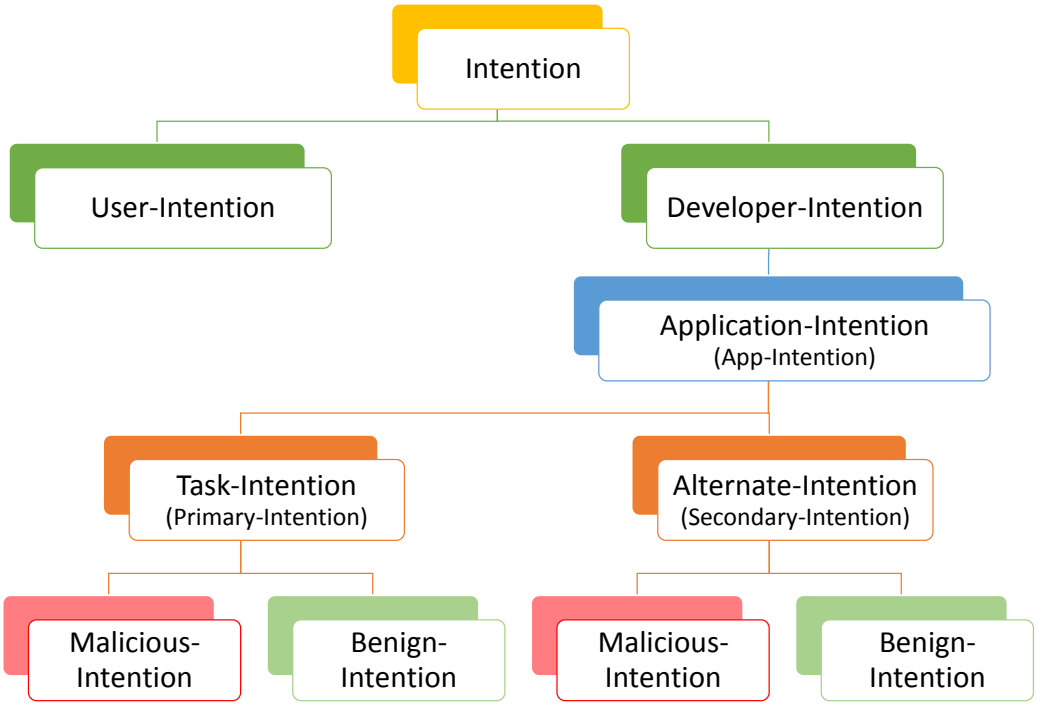


FIGURE 2.1. Different types of intentions and hierarchical view of those intention types.

2.1.1. User-intention

The user intention can be defined as the intention of the user who is using the software. In this context, a user can be any agent, either a human entity or another software or machine. A user's intention can directly affect the behavior of the application. For instance, consider the scenario of someone named Adam using a text messaging app. If Adam uses it with the intention of sending a message to Bob, the app will behave as a regular message sender. However, if Adam used it to

send random messages to millions of people everyday with the intention of selling his products or spreading illegal information, the behavior of the app becomes a spammer. Thus, even with a benign app, user-intention can change the benign behavior of the app. Further, user-intention also can reveal the type of the user. User types can be a regular-user, hacker, spammer, novice-user, expert-user etc. This is very useful in implementing security measures in the app itself to self-resist any malicious demands from a malicious user. H. Zhang et al. work can be identified as another example of user-intention is utilized in security domain. In their work, they identified software flows or malicious code activities in computer a network using the user-intentions [109].

In chapter 7 I discuss about how to determine the user-intention in a social network environment by using user's app usage pattern (in these examples I used several clues such as the number of Tweets per day, mean reply delay etc.) and how to identify different roles of these users. Identifying these user-intention is useful in securing a user based systems.

Chapter 8 is another example of utilizing user-intention in a secure environment. In this example, the user-intention is used to enhance the performance of a secure system that can be very useful in resource constrained environment such as mobile devices. This chapter explains about context-aware encryption based on user-intention such that, encryption can be relaxed or tightened. Here, the identified user-intention are whether the user intended to communicate sensitive information or the non-sensitive information. This work can be remodeled to provide security against unintentional security vulnerabilities in mobile phone conversations.

2.1.2. Developer-intention

Similar to user-intention, developer-intention can be defined as the intention of the developer of a software or an app. Software developers develop applications in achieving many different solutions. For instance, a developer can intend to develop a software to monitor the internet traffic in a smart phone. Mainly the developed software reflects the developer's intention of the software. Further, it is hard to determine what the actual intention of the developer is. The only way to determine this would be to analyze the software itself and its documentations. For this reason, in this dissertation I focus on the app-intentions rather than the direct developer intention. This is because at the end of the day, the intention of the app is the only important aspect in securing a system

rather than the actual developer-intention. The app-intention is explained in the next section.

2.1.3. Application-intention (App-intention)

As mentioned earlier, the app-intention is closely related the developer-intention. The app-intention can be defined as the intention of an application. App-intention can be clearly determine its behavior. If the app has malicious intentions then it will behave as a malicious app and vice versa. Therefore, intention of an app is a clear indicator in determining malware applications. In this dissertation I provide a behavioral based malware identification solution based on its intention identification. Therefore, identifying the app-intention is very critical in this work. Chapter 4 is dedicated to explain about these app-intention identification.

App-intention can be divided into several type of intentions.

- Task-intention
- Alternate-intention
- Malicious-intention
- Benign-intention

Thus, all the above intention types are referring to the app-intention. Here onwards in this dissertation whenever I refer to any of the above type of intentions, the reader should understand that it refers to a type of app-intention.

2.1.4. Task-intention

Task-intention can be defined as the operations and services an app is programmed to perform during its execution life cycle. For instance the task-intention of the Facebook application is to provide a social networking interface for the user. Every application has its own task-intention. An app cannot exist without its task-intention. An app is always intended to perform one or more activities. Some examples of app task-intention are communication, finance, gaming, photography, music & video playback, navigation, etc.

2.1.5. Alternate-intention

All the intentions that deviate from the task-intention can be identified as alternate-intentions of an app. For instance, an example app's task-intention is to provide a banking solution. If that app

provides a SMS facility in addition to its main task intention, it can be considered as an alternate-intention. Alternate-intentions not necessarily harmful to a system. It can be legitimate behavior intended by a regular benign developer. However, identifying the alternate-intention can reveal any clues on if the app possess any hidden malicious behavior or not.

2.1.6. Malicious-Intentions

If an app is intended to perform any harmful activities in a system, such as stealing data, sabotaging system integrity, or opening up vulnerabilities to attacks, it can be identified as having malicious-intentions. In general, if an app has harmful alternate-intentions (other than task-intention) then it has malicious-intentions. Further, if the task-intention itself is malicious, it is not necessary to find any alternate intentions to determine its maliciousness. It can be directly identified as a malicious app. However, usually this is not the case. Most mobile malware apps are repackaged in order to hide the malicious behavior under a good benign behavior within a regular app. This approach helps determine hidden malware samples as well.

There are many types of malicious intentions such as stealing user information, corrupting user data, hijacking user controls, etc. which are alternate to its task-intention.

2.1.7. Benign-intention

If an app does not have any malicious-intentions, then it is defined as having benign intentions. Thus, the definition of a benign-intention is mutually exclusive from the malicious-intention. In my opinion, when securing a system, an app is suspicious until proven benign. Therefore, as the definition suggests, I try to identify malicious-intention and only if it does not depict any such maliciousness may I then label it as possessing benign-intentions. Thus, in malware identification, the ultimate goal is to identify such applications that do not have any malicious-intentions so that they can be labeled as benign.

2.2. Android Platform and Android Applications (Apps)

Android is a mobile operating system developed by Google Inc. which is built on top of the well known Linux kernel. The Android platform is specifically design for mobile environment

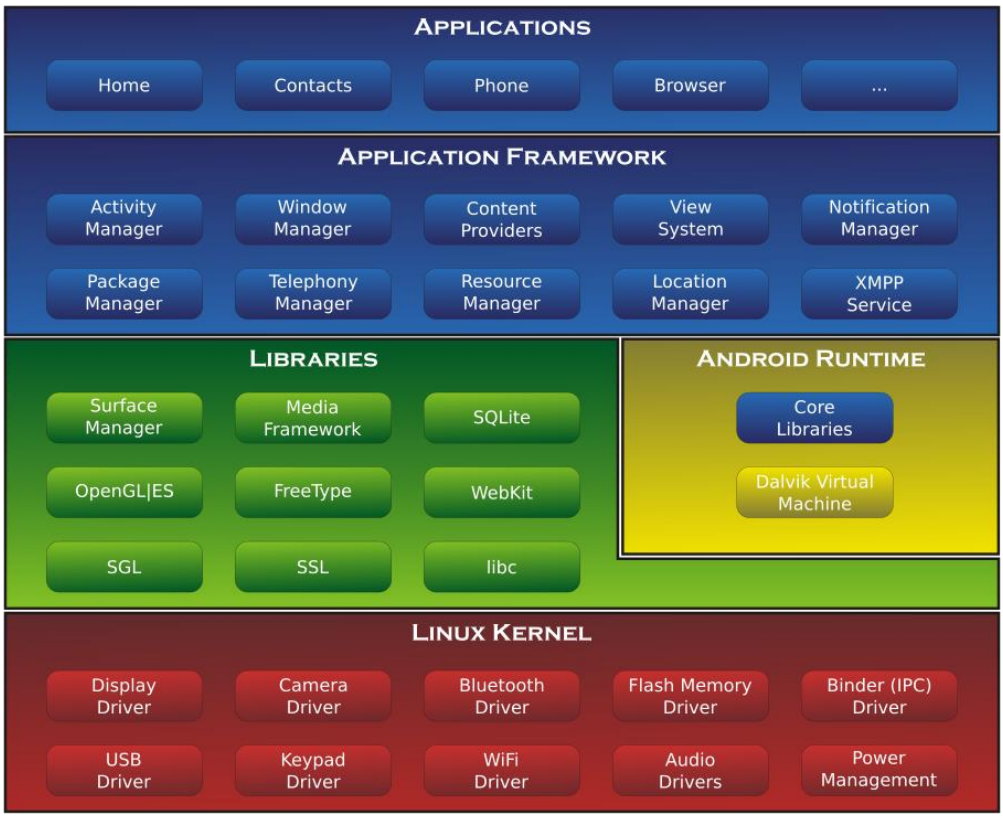


FIGURE 2.2. Android software stack. Uploaded in Wikipedia by Smieh. Free license.

with touch screen inputs such as tap, swipe, pinch etc. Figure 2.2 depicts the architecture of the android platform [30].

According to the Android architecture, the bottom most layer is the Linux kernel which is written in C or/and C++. However, upper layers such as android built-in apps are all written in java.

Since Android is a free and open source platform, third parties can add functionalities and do not have to rely on Google (ex: Multimedia Codecs, Sophisticated shell environment, etc.). This means that it is much easier to both understand and build applications on top of the Android platform. This is one of the main reasons why I chose the Android platform as the main testing ground in this dissertation. Many concepts discussed in this dissertation can be extended beyond the Android platform. For instance, application permissions are used in many other platforms such as Facebook apps, Google Chrome apps etc. [87] and concepts built on permission requests in this dissertation can be extended into such platforms as well. However, for simplicity and to construct

the concept, I have chosen Android as a viable platform.

The Android platform includes many features, included;

- Reuse and replacement of components
- Dalvik virtual machine
- Integrated browser
- Optimized graphics
- SQLite DB owned by each application (can be shared)
- Media support
- GSM Telephony
- Bluetooth, EDGE, 3G, and WiFi
- Camera, GPS, compass, and accelerometer
- Rich development environment

2.3. App Permissions

More information on Android can be found in the next chapter. Intention is related to Android however, mainly in regard to App permissions. Android OS separates the privileges for its apps and provides restricted access to each resource. In this way the Android OS enforces restrictions on Android user apps from performing specific operations [31]. If the app needs a certain resource to be utilized in its functionality, then it first needs to obtain certain permissions from the OS and these permissions will be visible and prompted to a user who is installing the app in an Android system. This gives the user an idea about the functionalities that the app possesses.

However, many general users are not aware of the technical or functionality aspects of many of these permission requests. To improve the understandability of these permissions, Google has improved the description on the android app permission request when it is prompted for a user. See figure 2.3. Though these descriptions give a general idea on what an app can perform with these permissions, most users are not much of an enthusiast about reading these description. Instead, the malware identification approach explained in this dissertation can give the user statistical confidence on whether the app is safe to install or not. This is the essence of 'malware identification using intention identification' that I discuss throughout this dissertation.

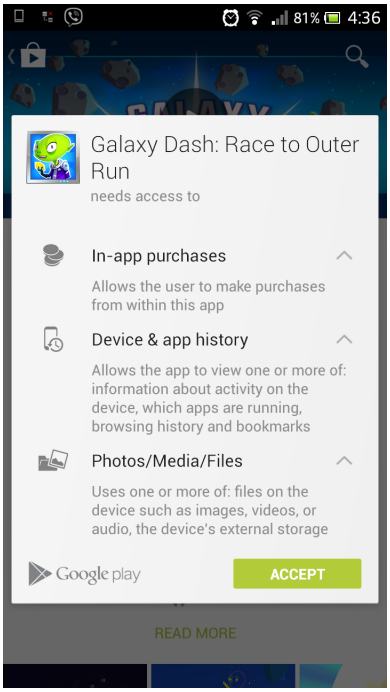


FIGURE 2.3. Android app permission request prompted when a user clicked the install button in Google play.

2.4. I-Shape of the Task-Intention

On understanding the importance of identifying the task-intention of an app, I expect all the apps with the same task-intention to behave in a similar way to perform activities and utilize system resources. Therefore, each app in the same task-intention group possesses a similar set of permission requests as their requirements are similar. Thus, when the permission requests are extracted from these same task-intention apps, they follow a similar permission request probability distribution. This can be formulated by constructing the permission request histogram for that task-intention category. Normalizing this histogram results in the probability mass function (PMF) of the permission requests, which have a specific shape for a given category of application [32]. I name this shape of the PMF as the I-Shape of the category. See Figure 2.4. Extracting the I-shape of my dataset is discussed in chapter 5.

2.5. Importance of Task-Intention in Malware Detection

As mentioned earlier, all the apps have a task-intention regardless of its nature of being malicious or not. Therefore, task-intention to app mapping is bijective (one-to-one onto mapping.

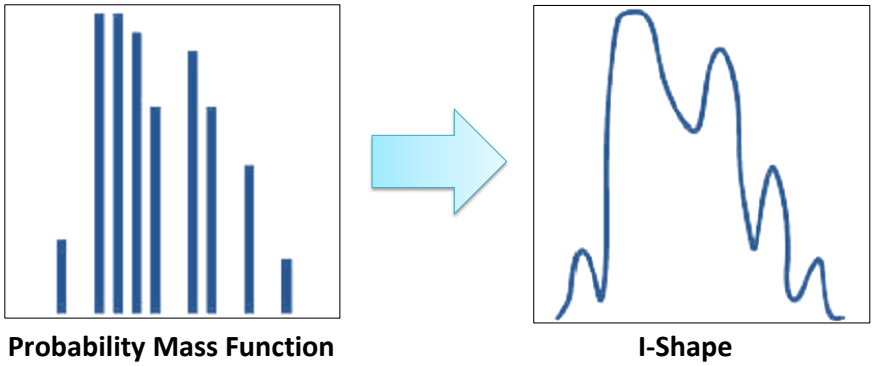


FIGURE 2.4. Probability mass function defines the shape of an app's intention category called the I-Shape. The smoothed curve is used to visualize the shape of the PMF. This smooth curve is not used in calculations rather the PMF is used.

It is assumed that an app has only one main task-intention). On the other hand, the mapping of malicious-intention to the app is a many-to-many so that an app can either have a benign intention, malicious intention, or multiple malicious intentions. Further, a malicious intention can be in many Apps. One of the main goals in mobile security is to identify such malicious apps (which has malicious intentions) and either disable them or eliminate them. Task-intention is very useful in determining its malicious intentions. For instance, if an app performs activities such as transmitting personal information to an external server, it is hard to determine whether this behavior is malicious or benign unless the actual task-intention is known. If its task-intention is to actually collect personal information such as the TurboTax app, the above mentioned data transmission activity is totally legitimate and can be considered a benign app. Alternatively, if these behaviors are performed by an app whose task-intention is to monitor battery level, it is suspicious and harmful. Thus, this yields the importance of identifying the task-intention such that it can be used to identify the malicious-intention. In this dissertation, my main goal is to identify the task-intention of an app and then use it to determine its malicious-intention.

CHAPTER 3

MOBILE SECURITY APPLICATIONS OF INTENTION IDENTIFICATION

In chapter 2, different types of intentions were identified and its usefulness in a mobile security system was discussed. In this chapter, two major intention types are emphasized, which are the app-intention and the user-intention. Under these two intention types, four security applications are discussed. They are malware identification, user role identification, context-aware encryption, and user identification using EEG. See Figure 3.1. Malware identification is an application of app-intention, and the other three are three different applications of user-intention. This chapter gives a brief idea about these three applications. Later chapters discus these application in detail.

3.1. App-intention and Malware Detection

Although signature-based malware detection is the most popular malware identification method, it has many drawbacks. For instance, malware samples can evade signature-based detection by adopting various transformations, ranging from trivial transformations like changing package names, to complex transformations like byte code encryption [85]. Therefore, behavioral-

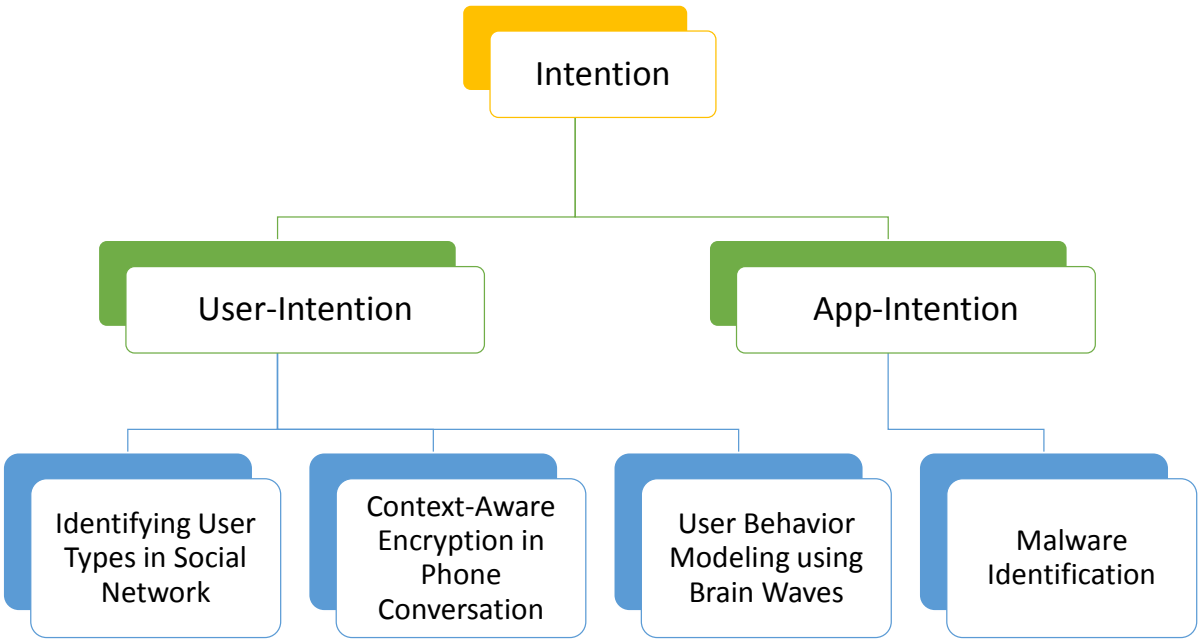


FIGURE 3.1. Intention type used in each mobile security application.

based static analyses can provide more robust and adaptive solutions for malware identification in conjunction with signature based solutions. Under the app-intention based solution, I introduce a novel framework to calculate the risk level of an Android app by using its requested permissions and its intentions. My approach performs a static analysis on various Android app samples and identifies potential malware samples based on its behavior. I only used machine learning models to find the *Task-Intention* of an application which is very different from malware classification from machine learning which was discussed in B. Sanz et al [90]. Thus, I only train the system with regular benign app samples which are abundant and safe. Furthermore, the used machine learning models do not need to be trained with any malware samples or its signatures to identify task-intention (task-intention is a part of app-intention). In fact, I never used any malware samples during the first phase of the algorithm. Also, the permission requests in this approach are not a set of features in the feature vector.

In malware detection I am following a two phase approach to solve this problem. In phase 1, I build and train several machine learning models to find the task-intention of an app. Once the models are trained, it can be used to identify the task-intention of any unknown app. Further, the I-shapes for each of these task-intention groups was constructed in this phase. Then in phase 2, I used the already trained model (in phase 1), to determine the task-intention of an unknown app. Once I identify the task-intention, the I-shape which corresponds to this task-intention was retrieved. Then I compare and contrast the permission requests of the unknown app with its I-Shape to determine whether it is a potentially malicious app or not. If the permission requests deviates from the I-Shape, I then determine this unknown app as potentially malicious. The implemented framework involves employing the machine intelligence along with the app's permission requests. In order to detect potentially malicious apps, the intention of an app is identified and its risk factor is calculated.

This unique potential malware identification is implemented using the I-shape which corresponds to the task-intention of the unknown app. This I-Shape is compared with the requested permission by using a matching algorithm that will generate a ratio called matching ratio. If the matching ratio is more than a certain confidence threshold, the app is potentially safe. Otherwise,

I flag the app as potentially unsafe and it is subjected to further analysis or quarantined.

The algorithm consists of four main sections:

(1) Feature extraction. (Chapter 4)

(2) Machine learning and training to identify the task-intention. (Chapter 4)

(3) Permission extraction to form the I-Shape. (Chapter 5)

(4) Malware detection. (Chapter 6)

The first three belong to phase 1, while potential malware identification belongs to phase 2. The explained overview architecture of both phase 1 and 2 are shown in figures 4.1 and 6.1. Malware identification is described in Chapters 5, 4 and 6.

3.2. User-Intention

3.2.1. Identifying User Types in Social Network

User-intention will be useful in determining the user's behavior. In chapter 7, user-intention identification is used to determine different behaviors of users in an online social network system. Then this identification picks out malicious users such as spammers based on their behavior (user-intention). Twitter is the social network utilized in this example. In this application, I looked into two different approaches; context-dependent and context-independent. The context-dependent approach is utilized when information about the network and tweet messages are abundantly available for analysis. It is intuitive that this information will provide the contextual information about the network. Otherwise, the context-independent approach is followed. In each approach the user-intentions (user behavior) are identified differently. In context-dependent approach, since the contextual information is available, user behaviors are recognized mainly based on how they interact with other users. When contextual information is lacking, the user behavior is identified mainly by the user's messaging patterns or in this case tweet pattern. In each approach, different machine learning model are utilized. More details are discussed in chapter 7. The methodologies used on Twitter social network to identify different user types can be extended to other social medias as well as to direct mobile phone functions such as on SMS and phone calls. Thus, same model can be easily extended to identify SMS spamming, phone spamming and other type of users.

3.2.2. Context-Aware Encryption in Phone Conversation

Another mobile security application of the user-intention identification is discussed in chapter 8. This chapter describes how to enhance the performance of a multimedia encryption system for mobile platform using the user intention. For this example, I will narrow down multimedia to be a voice communication through the Internet Protocol (IP) channels. This concept could be extended to apply in other multimedia applications such as video, text etc., when the multimedia information content can be recognized. In this particular example, the constraints of a mobile platform resources and how to utilize these resources efficiently without compromising the security of the data is described. In this chapter, I introduce a novel encryption protocol in a mobile voice communication using user-intention known as context-aware encryption. The key idea is to use the user-intention to recognize the sensitivity of the information. When the information is sensitive enough, I perform a regular stronger encryption while rest of the unsensitive information is left unencrypted or encrypted with less strength to save mobile phone resources.

3.2.3. User Behavior Modeling using Brain Waves

User-intention identification is important in user authentication. It is a clear fact that if the user is a human being, the user-intention is closely related to that human's intention. Since all these intentions are generated in the brain, it is interesting to explore the feasibility of extracting the user-intentions directly from the brain. Electroencephalogram (EEG) is a way of measuring the echoes of the brain signals using external probes. EEG is a main candidate in this approach. It is used to recognize the user-intention of a human being based on different activities and behaviors exerted on a system. The user-intentions extracted from the brain can be utilized in security systems directly. There are many researches working on EEG based security solutions, including in areas such as EEG based user authentication [51, 78], user identification [20], and cryptographic key generation using EEG [8].

My proposed goal is to use the user behavior on a smart phone to authenticate the user. Users often use activities like swiping, tapping, pinching, etc. on a smart phone. The EEG can be recorded while performing these tasks to identify such events. Then, these unique EEG signals can be used to identify different users [51, 78, 20]. As a preliminary step towards this goal, I have

performed experiments to identify such different primitive tasks using EEG with a single user. In chapter 9, I present the results of this experiment to explain the feasibility of the approach. This work is preliminary, as it needs to be extended to a complete authentication system. Additionally it can possibly be used someday to model a complex mind control system for a mobile device.

CHAPTER 4

IDENTIFYING TASK-INTENTION OF ANDROID APPLICATIONS FOR MALWARE DETECTION

This chapter first explains the dataset that I used in malware identification under app-intention identification section. Task-intention is one of the main sub category of app-intention. Later in this chapter, the task-intention identification is discussed in detail. It is important to understand that since task-intention is the main category of app-intention, one can safely interchange these two terms. However, app-intention is a more general term and task-intention is a more specific term.

Task-intentions can be identified using machine learning models. The idea here is to extract a set of features from mobile apps to group them into its relevant task-intention group using machine learning models. To achieve this, two different machine learning approaches were experimented; supervise-learning and unsupervised-learning. Based on these experiments, supervised-learning models did not perform well in task-intention identification. On the other hand unsupervised-learning models performed better. Thus, the unsupervised-learning models were utilized in the final malware detection process. All these information are covered in this chapter.

4.1. Mobile App Data Set

The foundation of app-intention identification and malware detection is a reliable data source. For this application, I have collected three different android apps datasets (APK files). The first dataset consists of malware samples downloaded [111] from North Carolina State University (NCSU) under the project title Android Malware Genome Project (AMGP). The second and third datasets were created by collecting benign apps from Google Play. All the collected necessary apps are reverse engineered to obtain the java source code and the XML files.

The content in this chapter is reproduced from Mohamed Fazeen and Ram Dantu, "Another free app: Does it have the right intentions?", Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on, July 2014, pp. 282−289, with permission from IEEE.

Data Set	**Count**
Benign Applications - With Class Labels	
Business	59
Communication	61
Finance	60
Games	61
Media&Video	63
Medical	59
Music&Audio	62
Photography	60
Productivity	64
Transportation	64
Other	68
Sub Total	681
Benign Applications - Unlabeled	1049
Malware Samples (In 49 families)	273
Total	2003

TABLE 4.1. The datasets for intention based potential malware identification.

The malware data set obtained from NCSU originally consisted of 1,260 android malware (APK) samples from 49 different malware families. This particular dataset is imbalanced with some malware families having close to 300 samples over others with only one sample. Thus, for uniformity, I restricted the number of samples to 10 per family, yielding a total of 273 malware samples from 47 malware families in the final malware dataset.

First a smaller collection of Android apps were manually downloaded from Google Play. I collected about 681 apps to construct this dataset. All these application samples were labeled with its intention class.

Then, I improved the smaller benign dataset further by downloading additional benign

android apps from Google Play. In this case I obtained the samples without the class labels for the unsupervised learning model. This collection alone totaled up to 1,049 apps, which were exposed to unsupervised machine learning algorithms. With this collection, the total dataset increased to more than 2000 app samples. See Table 4.1.

All the above described datasets consists of android apps in the form of Android platform installation package files called APK files. I used `dex2jar` and `JD-GUI` tools to extract the java source code and the `APKtool` to obtain the XML files, the `AndroidManifest.xml` and the `String.xml`.

4.2. Machine Learning Models for Task-Intention Identification

4.2.1. Introduction

Machine learning models can be mainly divided in to two categories, supervised learning and unsupervised learning. When the used train data set labeled with actual class labels it can be used to infer the classification model and this is called supervised learning [12]. When the train data set does not provided with the class labels, thus the clusters are inferred based on the unlabeled training set [12]. In the task-intention identification problem, I initially used several supervised learning models. But later I realized that the application category labeling is not reliable in the training set. Thus, I utilized couple of unsupervised models and I obtained better results.

4.2.2. Supervised Learning

Bayesian

Several Bayesian models such as Naive Bayes, Naive Bayes Multinomial, Bayesian Network were tested while training for this category. The smaller benign dataset, responded well to all the Bayesian models, especially the Naive Bayesian Multinomial.

In the Bayes approach, $P(\mathcal{C}_i|\mathcal{X})$, the probability of a given feature vector $\mathcal{X}$ belonging to a pattern class $\mathcal{C}_i \in \{Business, Communication, Finance, Games, Media\&Video, Medical, Music\&Audio, Photography, Productivity, Transportation\}$ is given by:

$$P(\mathcal{C}_i|\mathcal{X}) = \frac{\prod_{j=1}^{n} P(X_j|\mathcal{C}_i).P(\mathcal{C}_i)}{P(X_1, X_2, \ldots, X_n)} \quad (1)$$

In the above $\{X_1, X_2, \ldots, X_n\}$ are the components of the pattern vector $\mathcal{X}$, $P(\mathcal{C}_i)$ is the prior probability of the class $\mathcal{C}_i$, and $P(\mathcal{X}) = P(X_1, X_2, \ldots, X_n)$ is the probability of the sample vector.

Multilayer Perceptron

In this approach, a feed-forward neural network was used. During training phase, the pattern vectors with known class labels are presented at the input layer, and the outputs are observed. The weights connecting the neurons in the penultimate layer to those in the output layer should be adjusted so as to minimize the following mean square error function:

$$E = \sum_{k=1}^{n} (t_k - O_k)^2 \tag{2}$$

where t_k and O_k are the target (expected) and actually observed output values of the k^{th} output neuron. Since this weight training takes place in the backward direction, this algorithm is known as the back-propagation learning algorithm. Once the network is trained with all the training patterns, it can be used to classify the unknown patterns.

This model consisted of h number of hidden layers where h was calculated using the equation 3.

$$h = \frac{numberofattribs + numberofclasses}{2} \tag{3}$$

The system was trained by 500 epochs with the learning rate of 0.2.

Random Forest

Random Forest is a classifier formed by an ensemble of decision tree classifiers $\{h(\mathbf{X}, \Theta_k), k = 1, \cdots\}$ where the $\{\Theta_k\}$ are independent and identically distributed random vectors, and $\mathbf{X}$ is the input vector. The random forest classifies X into the class with maximum vote considering the votes for the most popular class at $\mathbf{X}$ by the constituent classifiers.

For the random forest approach I used 100 trees in the model.

4.2.3. Unsupervised Learning

K-mean clustering

An iterative clustering algorithm which selects K random points as cluster centers (means) according to calinski-harabasz criterion. In this algorithm, each data point X_n is assigned to K clusters where $k = 1, ..., K$. Under 1-of-K coding scheme a binary indicator variable $r_{nk}\epsilon 0, 1$ is defined. Then the distortion measure can be defined as in equation 4 [12]

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2 \tag{4}$$

where, μ_k is the center or the prototype associated with the kth cluster

The parameters of the k-mean model used here include; a seed, iterations and clusters of values 100, 500, and 10 respectively.

Expectation Maximization (EM) Clustering

Expectation Maximization (EM) algorithm, being an important algorithm of data mining, assigns a probability distribution to each instance which indicates the probability of it belonging to each of the clusters. The result of the cluster analysis is written to a band called class indices. For instance value '0' refers to the first cluster; a value of '1' refers to the second cluster, etc. This algorithm provides extremely useful results for the real world data set. In general, for a set of observed variables X and a set of latent variables Z and a parameter θ, the joint distribution can be defined as $p(X, Z|\theta)$ and if it is known, the goal in EM is to maximize the likelihood function $p(X|\theta)$ with respect to θ. [12]

The EM algorithm implemented utilizes a 100 seed with 100 iterations to categorize 10 clusters.

Table 4.8 depicts a sample of how the apps are clustered by the EM model. Similar apps, or different versions of the same app are grouped into the same cluster while different types of apps are falling under different clusters indicate the cohesiveness of the clusters.

Cluster 1
$ZFishThemeGOLauncherEXv1.0.apk$
$ZLoveThemeGOLauncherEXv1.2.apk$
$T-LOVELYCATGOLOCKERTHEMEv1.apk$
$SChristmasThemeGOLauncherv1.0.apk$
$MonkeyZThemeGOLauncherEXv1.0.apk$
Cluster 8
$Garfield'sDefense2v1.0.8.apk$
$CandyCrushSagav1.0.6.apk$
$ESPNXGamesv1.0.1.apk$
$8BallPoolv1.0.6.apk$
$GlowHockey2v1.0.4.apk$
Cluster 3
$B-RhymesDictionaryv1.5.6.apk$
$BigEncyclopaedicDictionaryFv1.0.apk$
$EnglishHungarianDictionaryFv1.0.apk$
$EnglishMongolianDictionaryFv1.0.apk$
$FrenchPortugueseDictionaryFv1.0.apk$

TABLE 4.2. Clustering results showing similar type of application in a single category.

4.3. Feature Extraction

It is intuitive that an intention of an app is directly related to its functionality. Thus, I was looking for a set of features that could represent the functionality of the app. Taking these facts into account, I performed three types of feature extractions. First, only the API calls of the java source code was extracted (M1). Then, as the second feature extraction method, M2 is constructed by extending M1 with character frequencies of each app. I browsed through the strings of the java source and the content of `AndroidManifest.xml` and `String.xml` to count the character usage in each app. I made sure not to extract features from the `<uses-permission />`, `<permission />`, `<permission-tree />`, and `<permission-group />` tags from

the `AndroidManifest.xml` as I use them to identify malware samples. As the third method, M3, I replaced the character frequency in M2 with dictionary based word counting. First, I constructed a dictionary with common words used in apps and grouped them into root words. See table 4.4 for a small sample of the dictionary and their root word (or the group name). Then, I extracted strings from the same set of resources as explained in M2, to count the words in each group. Each word hit in a group adds a count to the group. Then this group word count was added to M1 to construct M3 features.

Then, I tested three different machine learning models against these three feature extraction methods and I concluded that "M3: API Calls + Dictionary Based" was the best performing feature extraction method. This feature extraction method was used in both intention classification and clustering. More details about the feature extraction methods M1 - M3 are described below.

4.3.1. Method 1 (API Calls)

This approach of feature extraction entirely focuses on the import statements in each java file. On reverse engineering the app, it generates multiple java source files, ranging from 10s to 1000s. Firstly, I hand labeled and grouped the android API packages by looking at the functionality of the given application package. Table 4.3 illustrates a small section of the hand labeled packages. Further, a Matlab™ program script was devised to browse through each and every java file of individual apps to look for any `import` calls. On finding an `import` statement, the script extracts the imported package label and compares it against the list of all the android API packages. If there is a match, I mark the category based on the functionality this package belongs to as 1. Thus, following this approach provides a feature vector in a binary form with the dimension of the vector carrying the number of different labeled functionalities. Further, I counted and normalized the number of import calls across all the imports of that package by extending the approach, resulting in a scalar feature vector with real numbers instead of binaries.

The API package list was obtained from Android API reference at http://developer.android.com/reference/packages.html

Package Name	Group Label
android.graphics	Graphics
android.graphics.drawable	Graphics
android.graphics.drawable.shapes	Graphics
android.hardware	Input
android.hardware.input	Input

TABLE 4.3. Some Android API packages and their hand labels.

4.3.2. Method 2 (API Calls + Character Frequency)

Modifying the previous approach of feature extraction helps to include more features. The extended feature extraction implementation not only scans the java source files but also browses through the content of `AndroidManifest.xml` and `String.xml`. I extracted and retained all the attribute and tag values from the `AndroidManifest.xml`, except those tagged under `<uses-permission />`, `<permission />`, `<permission-tree />`, and `<permission-group />`. Thus, the machine learning models does not train any information with respect to the permissions. Post scrapping, the selected attribute information is considered as a long string and treated to calculate the character frequency for both the xml content in order to add them into the feature vector.

4.3.3. Method 3 (API Calls + Dictionary Based)

This method being an extension of Method 1, utilizes the scalar vectors along with building a dictionary of words and grouping them into a root word, instead of just counting the characters as described in Table 4.4. Selecting a key word is very critical and plays an important role in the implementation of this method. I researched a couple of applications at the Google Play to see what type of words best describe different categories of apps to carefully select key words and group them into buckets. Then I extracted the XML file as described in Method 2, in order to check the occurrence of the selected keywords in those XML files. If a keyword is found, the count of the bucket that key word belongs is increased by 1. Thus, on counting all the occurrences of the

selected keywords, the bucket is incremented and added to the feature vector correspondingly.

Key word	**Group**
device	business
move	business
keyword	business
email	office
card	office
folder	office

TABLE 4.4. Part of the dictionary which was used to extract the features in Method 3.

4.4. Task-Intention Identification by Supervised Learning

Initially, I started the task-intention identification with supervised machine learning models. To train and test these models I used the labeled benign app dataset. Though these models produced reasonable performance, it was not good enough for task-intention identification. It also lacked adoptability. Therefore, I later replaced this with an unsupervised model. However, I present the results of supervised models to compare with the unsupervised models. In this supervised approach, I followed the same steps as explained in Figure 4.1 with an exception of unsupervised learning method replaced by supervised learning model, and construction of classes instead of clustering.

Name	Machine learning model
Model 1	Bayesian
Model 2	Multilayer Perceptron
Model 3	Random Forest

TABLE 4.5. Different machine learning models used in the approach.

Three machine learning models were adopted: Naive Bayesian, Multilayer Perceptron, and Random Forest. See Table 4.5 The data mining tool WEKA was used to perform machine learning

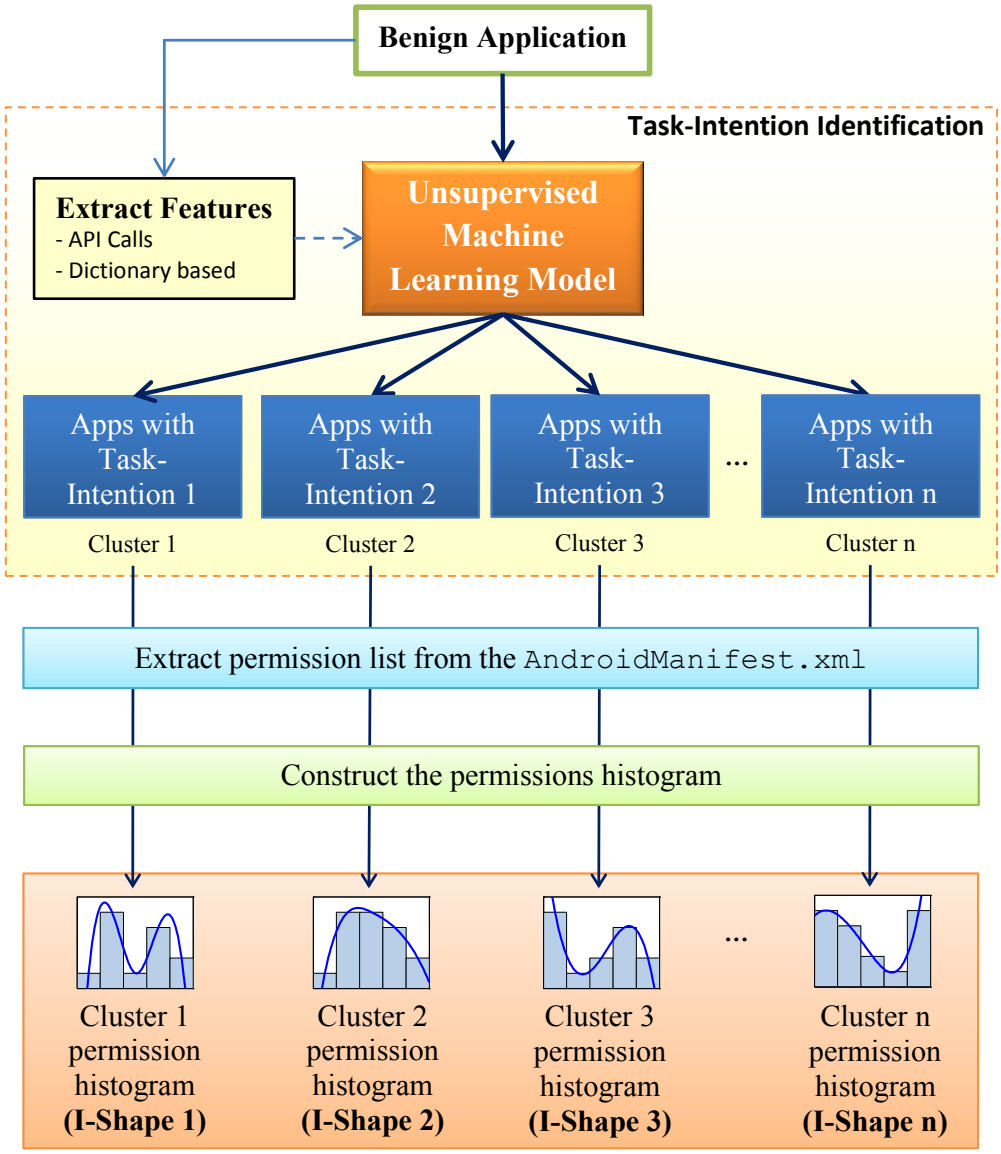


FIGURE 4.1. Phase 1 architectural diagram for task-intention identification. Here, I only train the machine learning models with benign apps.

training and analysis [45]. I used the default configurations provided by the WEKA for Naive Bayesian. A back-propagation feed-forward neural network was used in Multilayer Perceptron model. I set up the number of layers to be (Number of attributes + Number of classes) / 2. Then the system was trained with 500 epochs and a learning rate of 0.2. For the Random Forest model, the number of trees were restricted to 100.

The feature vectors created using different feature extraction approaches were used to train these machine learning models. Building a good machine learning model can help in classifying

		Model		
		Bayesian	Multilayer Perceptron	Random Forest
SD	M1	17%	33%	33%
	M2	35%	62%	42%
	M3	67%	56%	63%
LD	M3	56%	40%	59%

TABLE 4.6. The unsuccessful results of the supervised learning models against each feature extraction methods explained in section 4.3. To overcome this, I replaced these supervised learning models with the unsupervised learning models for task-intention identification. SD:Smaller Dataset, LD:Larger Dataset, Feature extraction methods M1-M3:Method1-Method3.

any unknown app to find its task-intention, such as Game or Business. I performed both 10-fold cross validation and 66% splitting of dataset when training and testing the model.

4.4.1. Unsuccessful Results of Task-Intention Id. by Supervised Learning

I witnessed several limitations for having obtained not-so high accuracy for the intention identification. See Table 4.6. App categorization in Google Play played a big role since I used the Google Play group labeling. Some applications did not have a clear border between categories, e.g. an application in the productivity category also showed relationship to the business category. Thus, there is a possibility of misclassification, thereby decreasing the performance and accuracy. The confusion matrix in the table 4.7(a) clearly illustrates this misclassification of different categories.

In this supervised learning approach task-intentions are decided simply by looking at the Google play category. This usage of Google Play group labeling could lead to an adversary attack where individuals can choose a category for their application that has all the right permissions. However, in the unsupervised learning (See section 4.5), task-intentions can be determined using its source code. One can easily change the labels to change the task-intention than to change the

(a) The confusion matrix

classified as										
a	b	c	d	e	f	g	h	i	j	
18	7	6	0	1	6	0	3	4	5	a
3	25	0	1	4	7	0	2	6	2	b
5	0	31	2	2	4	0	0	2	4	c
0	1	0	45	0	2	0	0	1	1	d
0	2	0	6	25	5	7	3	2	0	e
5	1	1	3	2	30	1	0	5	2	f
0	0	0	3	7	3	37	1	0	0	g
0	0	0	3	3	2	0	40	2	0	h
7	5	2	5	2	3	2	4	22	0	i
2	2	2	3	0	6	0	1	2	34	j

(b) Legend

a	=	Business
b	=	Communication
c	=	Finance
d	=	Games
e	=	Media&Video
f	=	Medical
g	=	Music&Audio
h	=	Photography
i	=	Productivity
j	=	Transportation

TABLE 4.7. The confusion matrix for the random forest (Model 3) classification of benign application dataset (large) with Method 3 feature extraction.

source code, strings and API calls to change its task-intention. For this reason, unsupervised approach is stronger than the supervised approach and it is one way of overcoming the drawbacks of supervised approach. Hence, I modified the task-intention identification by replacing the supervised technique with the unsupervised algorithms. I utilized the clustering algorithms to identify a task-intention. The following section describes the implementation of the unsupervised intention identification methodology.

4.5. Task-Intention Identification by Unsupervised Learning

After learning that the supervised model is unsuccessful, I improved the algorithm by replacing it with an unsupervised learning model. The downloaded benign apps were clustered into self identified task-intentions and I calculated their I-Shapes for each cluster. Since, unsupervised learning model determines its clusters, I say the task-intentions are self identified. Once the I-shape is constructed, each unknown app is classified into identified clusters and analyzed for potential maliciousness as explained in Figure 4.1. In this model two clustering methods were utilized: K-mean clustering and Expectation Maximization (EM) clustering. I used all the benign apps regardless of whether they are labeled or not to train the models. For the K-mean clustering, I used 100 as the seed, 500 as the number of iterations and restricted the number of clusters to 10. In EM clustering, I used the same parameters except iterated the model only 100 times.

Table 4.8 depicts a sample of how the apps are clustered by the EM model. Similar apps, or different versions of the same app, are grouped into the same cluster, while different types of apps falling under different clusters indicate the cohesiveness of the clusters.

4.6. Results of Task-Intention Id. by Unsupervised Learning

As mentioned in section 4.4.1, class labels are unreliable. Also, a hacker can purposely mislabel the app to delude the system. The unsupervised learning model was constructed to overcome such drawbacks. In fact, because the unsupervised learning model does not require any type of class labeling, this shows the model is immune to such deceptions. First, the benign apps were clustered into 10 clusters. Then the I-Shape for each of these clusters were constructed. The malware samples were then interrogated with the above clustering model to identify what cluster it

Cluster 1 - Mostly Themes
$ZFishThemeGOLauncherEXv1.0.apk$
$ZLoveThemeGOLauncherEXv1.2.apk$
$T-LOVELYCATGOLOCKERTHEMEv1.apk$
Cluster 8 - Mostly Games
$Garfield'sDefense2v1.0.8.apk$
$CandyCrushSagav1.0.6.apk$
$ESPNXGamesv1.0.1.apk$
Cluster 3 - Mostly language dictionaries
$B-RhymesDictionaryv1.5.6.apk$
$BigEncyclopaedicDictionaryFv1.0.apk$
$EnglishHungarianDictionaryFv1.0.apk$

TABLE 4.8. Clustering results showing similar type of application in a single category.

will be classified into. The second column of table 4.9 depicts this classification.

I evaluated the benign app clustering using the classification approach [103, 104]. First, the clustered benign apps were labeled with the corresponding cluster. Then, a naive Bayes supervised machine learning model was used with a 10-fold cross validations to evaluate the performance. This classification produce an accuracy of 88.5%. Therefore, unsupervised learning approach performed better than supervised learning in grouping apps into its task-intentions.

4.7. Summary

Task-intention identification using machine learning models were explained in this chapter. The dataset consisted of 1730 benign Android apps and 273 malware samples. Two different types of machine learning models were attempted. Supervised learning models did not produce successful results. However, unsupervised learning models performed well in grouping apps into correct clusters based on the cluster performance analysis and manual observations of the clusters. In the next chapter, these clustered apps are used to construct the I-shape of the cluster using their permission requests.

Cluster #	Benign	Malware
Cluster 0	21 (2%)	1 (0%)
Cluster 1	232 (22%)	134 (49%)
Cluster 2	57 (5%)	1 (0%)
Cluster 3	272 (26%)	111 (41%)
Cluster 4	36 (3%)	0 (0%)
Cluster 5	41 (4%)	0 (0%)
Cluster 6	41 (4%)	0 (0%)
Cluster 7	11 (1%)	0 (0%)
Cluster 8	206 (20%)	23 (8%)
Cluster 9	132 (13%)	3 (1%)

TABLE 4.9. Results obtained by unsupervised clustering of apps.

CHAPTER 5

ANDROID APP PERMISSIONS AND I-SHAPE ANALYSIS FOR MALWARE DETECTION

I-shape is a key component in identifying malicious apps in my malware detection approach. In chapter 2, section 2.4, I have introduced the concept of I-shape. In this chapter, I discuss how I extracted the permission requests from the applications in the dataset and used them to create the I-shapes.

5.1. Permission Extraction & I-Shape Construction

Every android app consists of a file named `AndroidManifest.xml` which describes the important information of an app to the Android OS. The Android OS security feature allows the app to access critical functionalities on a need basis, which are requested via this `AndroidManifest.xml`. Thus, in this file, under the tag `<uses-permission />` one can find different types of access permission request the app requesting from the OS for it to execute. If a permission is not defined in this manifest file, even if the app's java code attempt to access the resource with APIs, the OS will refuse these requests and will throw a runtime exception. Therefore, the manifest file is the most reliable place to look for what type of permissions are required by the app to execute on an Android operating system.

I-shape is composed of an app's permission requests. In general, I-shape represents the shape of the permission requests probability distribution in a group of apps. Here, the app group must be homogeneous, in order for the I-shape to represent a meaningful information. Thus, I grouped the apps based on its task-intentions such that it forms a cohesive homogeneous group. Now, when the I-shapes are constructed for each of these task-intention groups, the I-shape dictates that this is the most probable permission request signature these type of apps request. It will be very useful in determining the malicious behavior of an unknown app. This is discussed in the next chapter.

The content in this chapter is reproduced from Mohamed Fazeen and Ram Dantu, "Another free app: Does it have the right intentions?", Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on, July 2014, pp. 282−289, with permission from IEEE.

To construct the I-shape, I automated the permission extraction process using a Matlab™ script. Two types of extractions are performed. The first type involves extracting the permissions requested and their counts of how many times they are requested by all the malware samples in the dataset. The figure 5.1 depicts what types of permissions were requested by all the malware in the dataset and how many times they were requested in overall. Note that, here I performed the permission extraction from the malware apps. However, none of these malware permission are used to construct the I-shape. I-shapes are always constructed from the permission requests of the benign apps. The second type of permission extraction describes this.

It is intuitive that these permissions are common for a given task-intention. For instance, if the task-intention of a set of apps is to provide an SMS communication, it is clear that all these apps must possess similar permission requests such as SMS_SEND, SMS_RECEIVE etc. In the second extraction type, I extracted all the permissions of benign apps based on its given task-intention group to build the permission-request-histograms. Then, probability mass functions (PMF) for each of these categories are constructed using those histograms. Consequently, the permission I-Shapes are constructed by using these PMFs for each of these same categories, as illustrated in figure 5.2. Here, the used task-intentions are the ones I identified in the previous chapter. Following this approach, I collected permission-requests of the benign apps in the dataset and then the I-Shapes were created for the two benign datasets. The malware dataset is not used to construct the I-Shape. However, permissions of the malware dataset was also extracted as it will also be used in the malware identification process, which is explained in the next chapter.

5.2. Summary

I-shape construction for the task-intention groups was discussed in this chapter. In the next chapter, malware detection is explained by using these identified I-shapes and task-intentions.

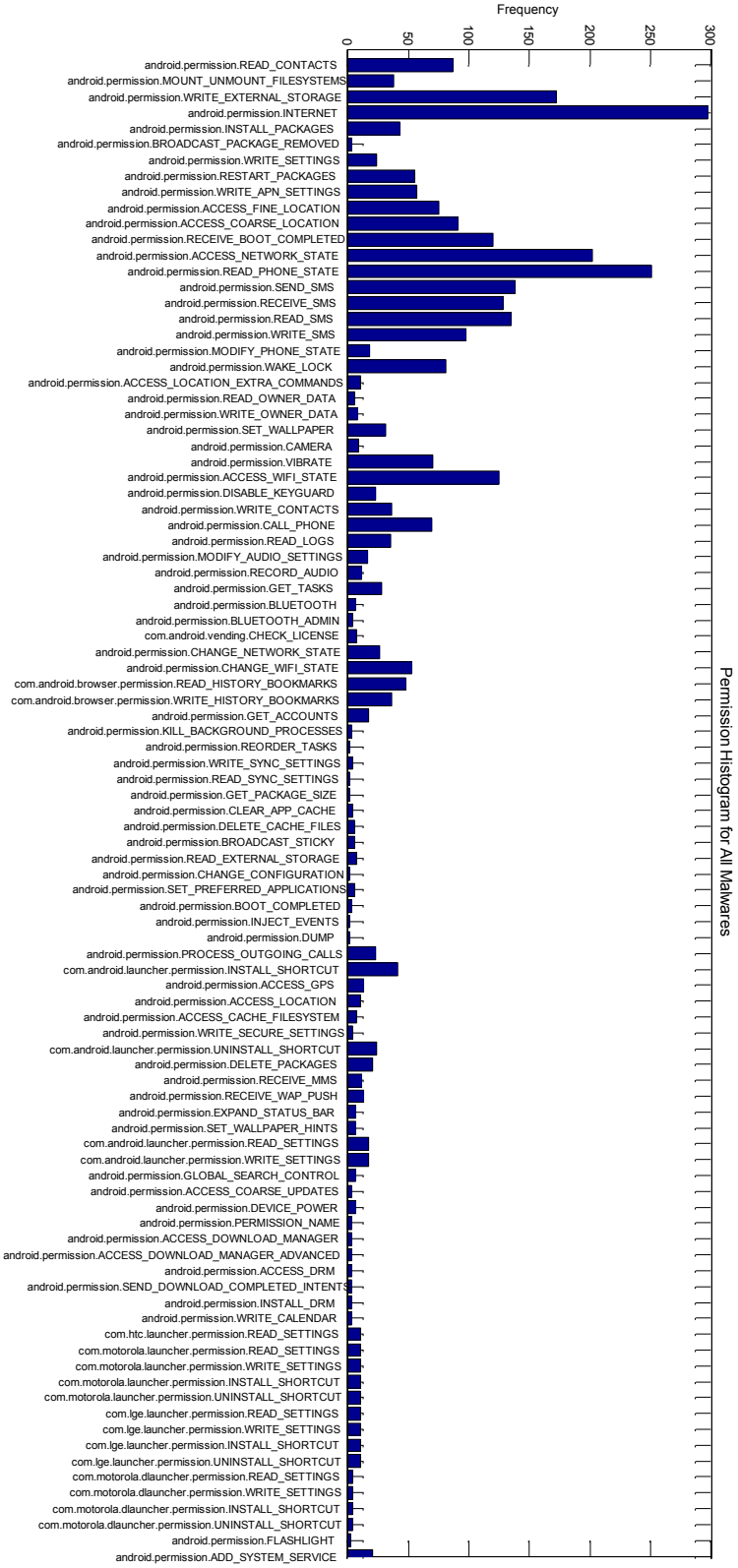
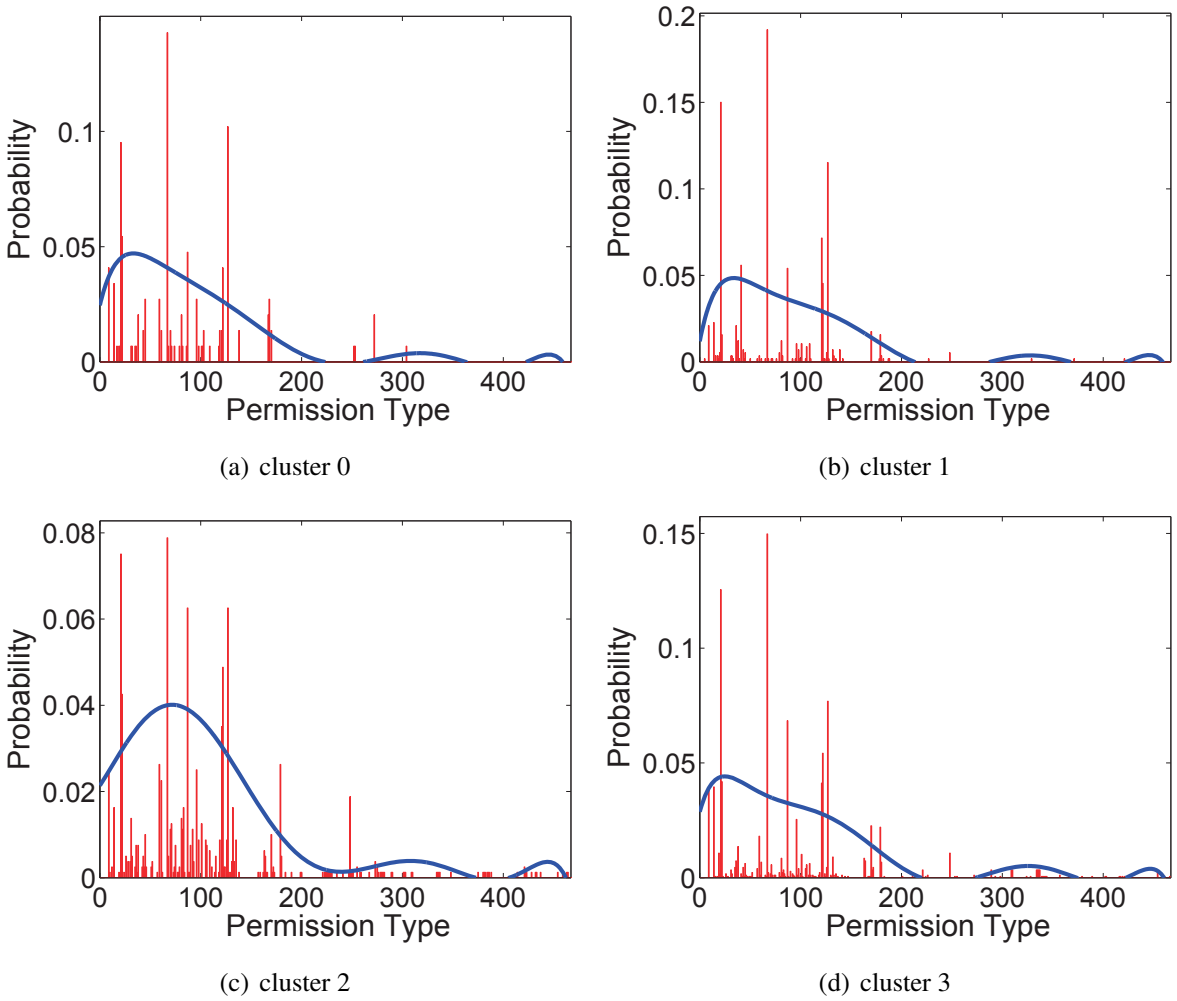


FIGURE 5.1. All types of permissions requested by the malware and their frequency.



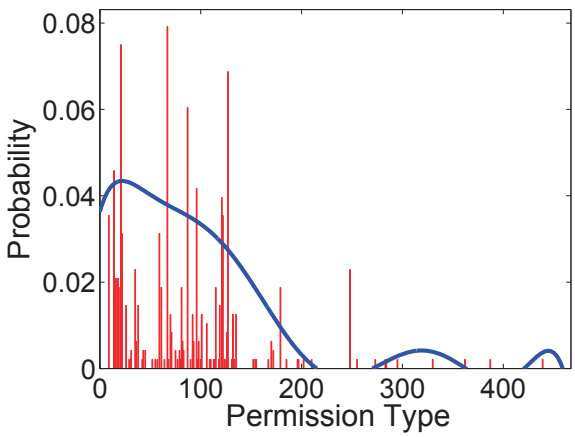
(a) cluster 0

(b) cluster 1

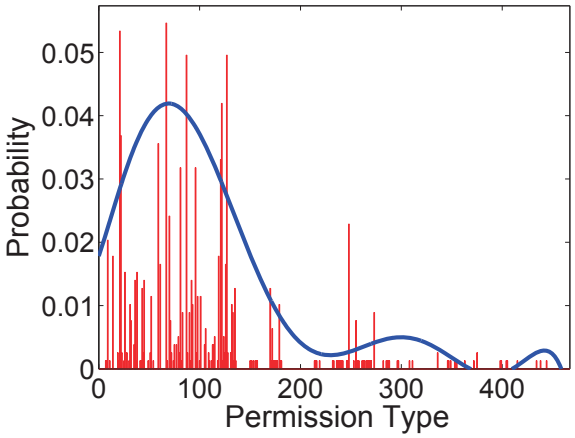
(c) cluster 2

(d) cluster 3

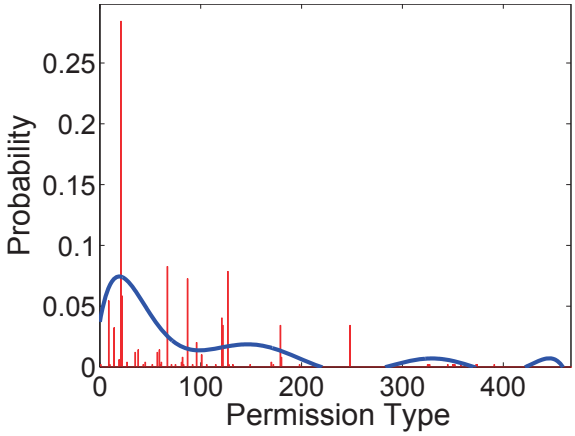
FIGURE 5.2. The I-Shapes of the clusters I obtained. Each I-Shape is constructed by using the probability mass functions (PMFs). PMFs are constructed by normalizing the corresponding permission histograms [32]. List of the permission types (x-axis) is presented in figure 5.1. Note that the smoothed curve is used to visualize the shape of the PMF only. In calculations, probability values of each permissions are used.



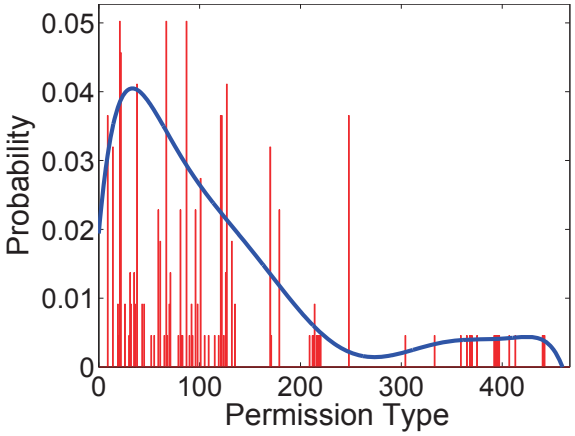
(e) cluster 4



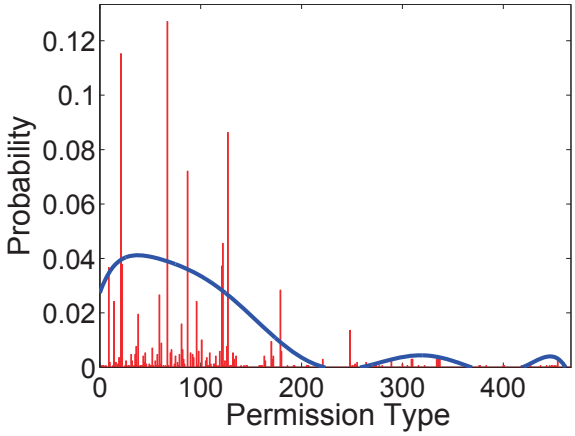
(f) cluster 5



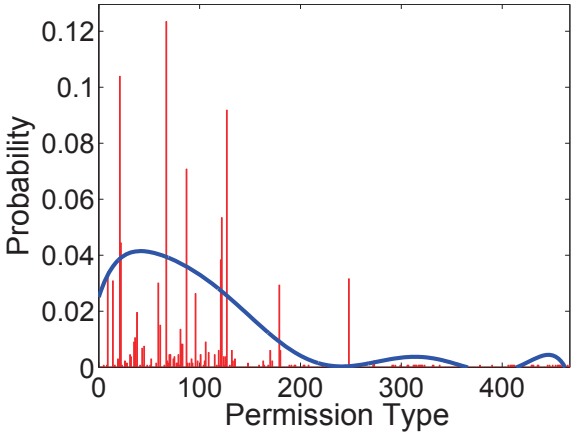
(g) cluster 6



(h) cluster 7



(i) cluster 8



(j) cluster 9

FIGURE 5.2. The I-Shapes of the clusters (cont.).

CHAPTER 6

TASK-INTENTION AND MALWARE IDENTIFICATION

This is the final stage of the malware identification algorithm. Malware identification is a mobile security application of app-intention identification. To determine the maliciousness of a given android app, the above mentioned task-intentions and I-shapes were used. Therefore, it is advised to read the above chapters before getting into this chapter.

6.1. Malware Identification

This malware detection algorithm tries to identify whether a given unknown app is potentially harmful or safe. First, features were extracted from this unknown app and the feature vector was constructed. Then I applied it to an aforementioned trained machine learning model and obtained the app category c to which it was classified. Now, the task-intention of this unknown app is identified. Thus, I retrieved the probability mass function H_c corresponding to the identified task-intention group. According to the aforementioned method, I also extracted the permission-requests of this unknown app, P. Then I used a constant threshold T to extract the most probable permission list from H_c according to the following method;

Let, $H_{c_i} \in \{$*Permissions in the PMF*$\}$ where $i \in \{1, 2, \ldots, n\}$

if the value of $H_{c_i} > T$ **then**
    Retain this permission and add to the list L_P
else
    Discarded the permission and do not add to the L_P
end if

The value of T is taken as 0.001 based on the average probability value of all the I-Shapes. Now I have two lists, a list of permissions L_P that this app is supposed to have, and the permissions

The content in this chapter is reproduced from Mohamed Fazeen and Ram Dantu, "Another free app: Does it have the right intentions?", Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on, July 2014, pp. 282−289, with permission from IEEE.

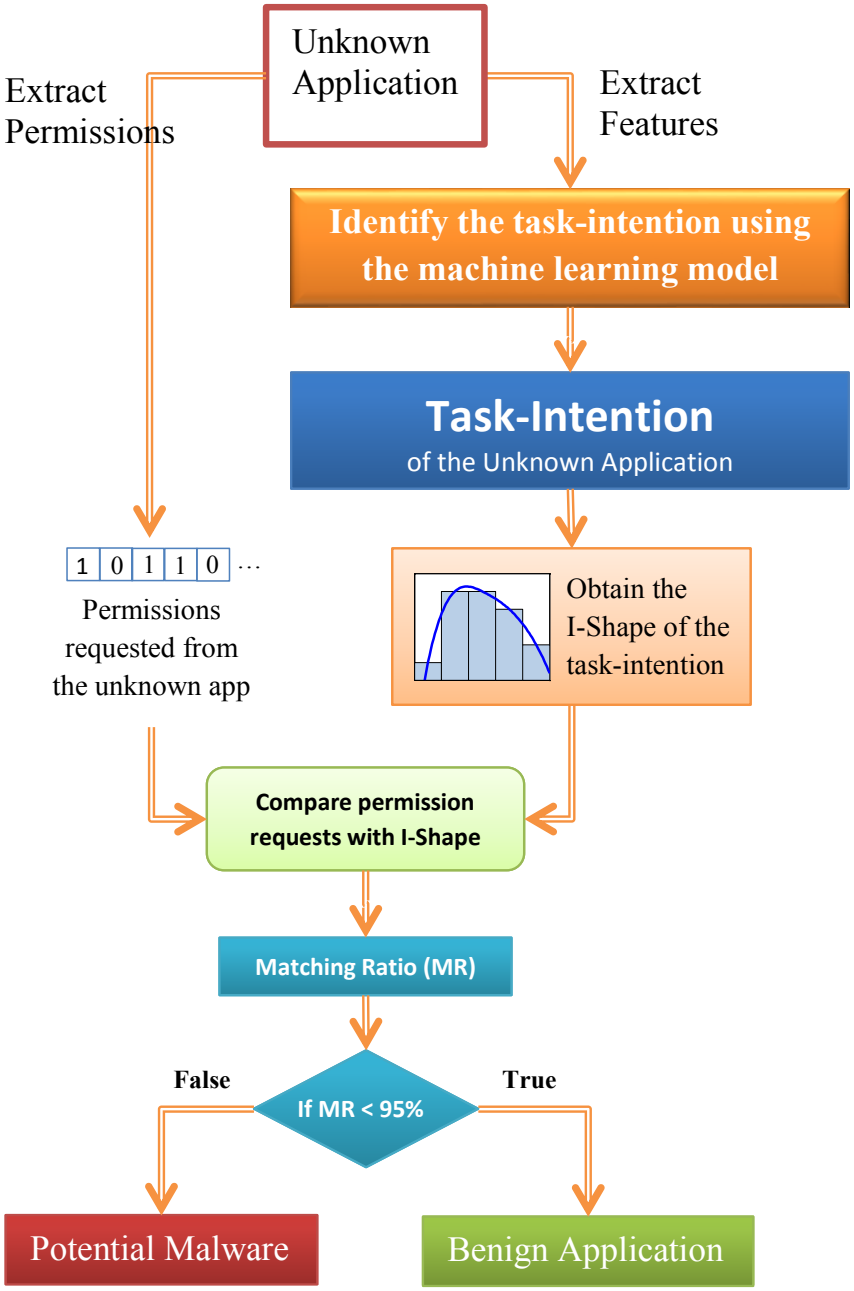


FIGURE 6.1. Phase 2 architectural diagram for potential malware identification. Here, I compare the permission request list of the unknown app with I-Shape of its task-intention.

that this unknown app requests P. I then searched for each P in L_P. If a permission was found, I increased a counter named 'matched permission counter' M_Cnt by 1. Then, I calculated the matching ratio by using the equation, $MR = M\_Cnt/count(P)$. I used a confidence value of above 95% in MR to label the unknown app to be safe. Otherwise I labeled them as unsafe.

	Actual	
	Malware apps (+)	Benign apps (-)
Predicted (+)	True Positive (TP)	False Positive (FP)
Predicted (-)	False Negative (FN)	True Negative (TN)

TABLE 6.1. Confusion matrix terms.

I picked a higher value of 95% to make sure that I-Shape and the permission requests are well matched to determine whether it is benign.

To test the system performance, I tried to identify the malware in the malware dataset (positive class) and in a fresh benign app dataset (negative class). Ideally, I would expect all the apps in the malware dataset to be detected as malware, while from the benign app dataset I would expect to see none. The table 6.1 depicts the definition of the confusion matrix.

By considering the malware dataset as the positive class, I calculated the sensitivity using the equation "*Sensitivity (%)* $= \frac{TP}{TP+FN} \times 100\%$"

Also, I calculated the specificity by considering the benign app dataset using the equation "*Specificity (%)* $= \frac{TN}{TN+FP} \times 100\%$"

Then I calculated the balanced accuracy using the formula 5 to determine the performance of the system [14]

$$\textit{Balanced Accuracy} = \frac{\textit{Sensitivity} + \textit{Specificity}}{2}\% \tag{5}$$

6.2. Potential Malware Identification Results

This section portrays the final performance of the potential malware identification. I obtained these results by performing both phase 1 and phase 2 sequentially. In phase 1, I decided that the most suitable approach is the unsupervised approach. Therefore, I used two different unsupervised learning models as the phase 1 task-intention identification to obtain the results in phase2. Thus, I had two overall accuracies corresponding to two different phase 1 unsupervised learning models. See Table 6.2 for malware detection accuracies.

Further, for comparison purposes, I also used the supervised learning model for phase 1 to obtain results in phase 2. By choosing the supervised machine learning model as the task-intention identification, I produced a balanced accuracy of 70% in detecting potential malware samples in the best case. This was improved up to 75% using unsupervised models. However, the accuracy of the unsupervised approach was 89% due to better recognition of benign samples. That is out of 1500 benign samples about 1436 identified as benign (TN). Further, out of 273 malware samples 137 identified as malicious (TP). This yield an accuracy of 89%. Further, K-mean clustering sensitivity results indicated that it performs better in detecting malware samples than that of in EM. However, EM was better in identifying benign apps. Due to better identification of benign apps in EM, it produced a higher accuracy than k-mean as the dataset had more malware samples than benign samples. Yajin Zhou et al., tested these same malware samples in four popular commercial Android anti-virus tools to evaluate their performance [111]. Their results depicted that in the best case, the Lookout anti-virus software tools detected only 79.6% of the malware samples. However, both AVG and Norton anti-virus tools detected only 54.7% and 20.2% respectively. I wanted to point out that they used a bigger malware sample of 1260 malware apps while this only used a subset of that. Further, this was a sensitivity results. My approach obtained a similar sensitivity performance as the Lookout, when K-mean clustering was used. See table 6.2. However, my approach outperformed other two popular tools. In general, this approach not only exceeded the performance of the existing tools, but it also exhibits a safer approach, as I do not need to execute the malware sample in order to detect its malicious payload.

My detailed analysis of potential malware detection indicated that samples in some malware families were completely identified by this algorithm, while some of them completely evaded. See table 6.3. Malware families like Gone60, KMin, Plankto, zHash mainly consist of stand-alone malware samples [111]. This algorithm identified all the sample in these families. Also, repackaging malware families like BeanBot, DroidKungFuSapp, GoldDream are also completely detected. Malware families like AnserverBot, BaseBridge, and Plankton are also classified as 'update attack' class and these families also were detected with higher accuracy. However, some repackaging malware families like DroidDreamLig, DroidKungFu1, DroidKungFu2, DroidKungFu3 were detected

Clustering algorithm used- in intention identification	Sensitivity	Specificity	Balanced Accuracy	Accuracy
k-mean	77%	68%	72.5%	70%
EM	50%	100%	75%	89%

TABLE 6.2. Overall potential malware identification performance when unsupervised learning (clustering) is utilized. Due to better identification of benign apps in EM, it produced a higher accuracy than k-mean as the dataset had more malware samples than benign samples.

with lower accuracies. GoldDream malware family is the only family in the database that belongs to both repackaging and stand-alone. My algorithm identified all the samples in this family as well. In overall, this algorithm identified 50% of the repackaged malware apps and 55% of standalone malware apps.

6.3. Related Work

Rassameeroj and Tanahashi[84] clustered apps based on their permission requests. It was concluded that it is possible to detect malicious apps based on permission requests as long as there is a careful selection of the permission set. D. Barrera et al. explains how Android permission can be utilized in mobile security [9]. They utilized Self-Organizing Map (SOM) to evaluate the access control permission requests. Their analysis was based on about 1,100 Android apps to study the permission usage patterns and they claimed that certain permissions are very frequently used, while some are not.

A. Shabtai et al. also adopted a machine learning model to classify android apps into two groups (tools and games) with a dataset of 2,850 apps [91]. R. Perdisci and M. U's showed, how unsupervised learning can be used to perform malware clustering and how it can be evaluated. The database consisted of about 3,000 malware samples.[75].

Apps in Facebook also follow a permission based access control. M. Frank et al. utilized a probabilistic model to mine permission request patterns from Android and Facebook applications with a large number of samples [41]. Though their goal was not to identify malware samples, they used an unsupervised learning model to find permission request patterns. One of their findings indicated that Android app categories are related to its permission request patterns. This fortifies my own argument that similar task-intention apps are related to its permission request patterns.

Family	Det.	Total	(%)	RPKG	Upd	DrR	StA
BeanBot	8	8	100	✓			
CruseWin	2	2	100				✓
DroidKungFuSapp	3	3	100	✓			
FakeNetflix	1	1	100				✓
GamblerSMS	1	1	100				✓
GGTracker	1	1	100			✓	✓
GingerMaster	4	4	100	✓			
GoldDream	10	10	100	✓			✓
Gone60	9	9	100				✓
HippoSMS	4	4	100	✓			
KMin	10	10	100				✓
NickyBot	1	1	100				✓
NickySpy	2	2	100				✓
Plankton	10	10	100		✓		✓
Walkinwat	1	1	100				✓
zHash	10	10	100				✓
ADRD	9	10	90	✓			
AnserverBot	9	10	90	✓	✓		
BaseBridge	9	10	90	✓	✓		
Geinimi	8	10	80	✓			
Pjapps	8	10	80	✓			
DroidDream	4	10	40	✓			
GPSSMSSpy	2	6	33.3				✓
DroidKungFu1	3	10	30	✓			
DroidKungFu2	3	10	30	✓			
DroidDreamLight	2	10	20	✓			
DroidKungFu3	2	10	20	✓			
jSMSHider	1	10	10	✓			
Asroot	0	8	0				✓
Bgserv	0	9	0	✓			
CoinPirate	0	1	0	✓			
DogWars	0	1	0	✓			
DroidCoupon	0	1	0	✓			
DroidDeluxe	0	1	0				✓
DroidKungFu4	0	10	0	✓			
DroidKungFuUpdate	0	1	0	✓	✓		
Endofday	0	1	0	✓			
FakePlayer	0	1	0				✓
Jifake	0	1	0	✓		✓	
LoveTrap	0	1	0				✓
RogueLemon	0	2	0				✓
RogueSPPush	0	9	0				✓
SMSReplicator	0	1	0				✓
SndApps	0	10	0				✓
Tapsnake	0	2	0				✓
YZHC	0	10	0				✓
Zsone	0	10	0	✓			

TABLE 6.3. Malware samples identified by this method for all 47 malware families and their attack method. Results are sorted based on performance on different families. Legend: Det. - Detected, RPKG - Repackaging Attack, Upd - Update Attack, DrR - Drive-by Download Attack, StA - Standalone.

Y. Zhou and X. Jiang from North Carolina State University instated the Android Malware Gnome Project to collect Android malware samples [111]. This is the source of malware apps used in this work. As an extension for this work, Y. Zhou et al. also published a systematic study about detecting malicious applications on popular Android markets [112]. In their malware detection they followed a two step approach, where they first performed a permission based filtering, followed by behavioral footprint matching. In the first phase, the apps were filtered by looking at the permissions which were probable for certain malware family. In this approach, I stressed the extraction of I-Shape for a similar task-intention group. Their results showed that the official market had a low infection rate of 0.02% compared to that of alternative markets, which ranged from 0.2% to 0.47%.

X. Wei et al. studied about how Android app permission requests evolved from the time period of 2009 to 2011 [100]. Their findings indicated that permission requests tend to grow and aimed towards providing access to new hardware features, including dangerous permissions. They also identified that most of the applications are over privileged. These negative impacts motivated me to implement this security system.

Undocumented permissions always seem unnecessary for applications which intend to do other jobs. These permissions may be used to create a more complete user profile by actively collecting personal information, which can be dangerous, as discussed by Hao Chen et al. [89]. Further, they developed a framework which identifies a set of security-sensitive API methods, specifies their security policies, and re-writes the bytecode to interpose the invocations in a given application that is being used [11].

The manifest and its permission requests do allow the possibility of ascertaining the app's functionality, thereby providing a good initial platform for identifying malicious activities. As explored by [39] and [111], unnecessary permission requests cause an app to be over-privileged. These permissions, in combination with other popular requests, could possibly lead to privacy leaks, thereby causing the apps to become malicious.

G. Canfora et al. classified Android malware using three different metrics; the occurrences of a specific subset of system calls, a weighted sum of a subset of permissions that the application

required, and a set of combinations of permissions. They evaluated 200 malware and 200 benign apps to assess the performance of these metrics. Again, this model mainly differed from my approach as they trained the models with malware features. [16]

B. Sanz et al. conducted a similar study to categorize Android malware applications. Their dataset consisted of about 820 samples from 7 categories. The WEKA tool was used to perform the data mining. It is interesting to note that one of their feature extraction methods is the frequency of occurrence of the printable strings, which is similar to one of ours. It is very important to note that their approach is different in several ways. My approach does not use the machine learning models to identify malware samples rather I used them to find the task-intention of an app. Further, I used the permission requests to identify potential malware apps, they are not a set of features in the feature vector while in Sanz et al. approach it is in the feature vector. Further, my approach can be extended to identify zero day malware apps while in Sanz et al. approach zero day malware identification depends on the performance of the classifier. [90]

6.4. Summary

Identifying potentially harmful Android apps by using the intention of those apps were highlighted in this work. Machine learning models were used to identify the task-intention of an app. Once it is known, I retrieved the most probable permission-requests for that task-intention group called the I-Shape and compared it with the permission-requests of the unknown application. Based on this comparison, I identified whether an app is potentially malicious or not. This method could be utilized to identify the safety of an app before its installed or to identify brand new malware apps. I used both supervised and unsupervised machine learning models to determine the task-intention of an app and the unsupervised model outperformed the other. I obtained an accuracy of 89% in identifying potentially harmful apps. I believe this accuracy can be increased by improving the dictionary and by training more benign app samples. Thus, it leads to a life long learning, which will evolve into better performance. In the next chapter, identification of leaders, lurkers, associates and spammers in a social network is discussed.

CHAPTER 7

USER-INTENTION AND MALICIOUS USER IDENTIFICATION IN SOCIAL NETWORK

7.1. Introduction

Recent years have witnessed a proliferation of social networks with popular applications such as Twitter, Flickr, YouTube, LiveJournal, Orkut, and Facebook [38]. These networks are poised for further growth with emerging applications such as social television (TV) [55] to share people's thoughts with family and friends while watching TV alone at home. With this rapid pace of growth in social networks (SN), there has also been a growing interest in the Internet research community in the SN analysis to address various aspects of social networking. Wu and Zhou performed a SN analysis using Del.icio.us, a free SN bookmarking web service that permits users to tag each one of their bookmarks with freely chosen index items [107]. They have shown that the patterns of users' tagging can be detected by visualizing the users' tagging behaviors and tag's evolution. They also established that the users within a subscription network share more common interests than random pairs of users in Del.icio.us. Using a large dataset containing 11.3 million users and 328 million links from multiple online SNs namely Flickr, YouTube, LiveJournal, and Orkut, [64] developed a large-scale measurement procedure to analyze the structure of multiple online social networks, and found significant structural differences between the SNs and previously studied networks, particularly the Web. SNs have a much higher fraction of symmetric links and exhibit much higher levels of local clustering. These properties may be used to design effective SN algorithms and applications. For an analytical study on information dissemination in large-scale SNs, [17] use the data including 11 million photographs from favorite markings of 2.5 million users on the Flickr to address the following questions: i) how widely does information propagate in the SN? ii) How quickly does information propagate? iii) what is the role of word-of-mouth exchanges between friends in the overall propagation of information in the network? Results of

The content in this chapter is reproduced from Mohamed Fazeen, Ram Dantu, and Parthasarathy Guturu, "Identification of leaders, lurkers, associates and spammers in a social network: context-dependent and context-independent approaches", Social Network Analysis and Mining 1 (2011), no. 3, 241−254, with permission from Springer.

their analysis indicate that: i) even in case of popular photographs, information does not spread widely throughout the network, ii) it spreads slowly, and iii) information exchange between friends accounts for over 50% of all favorite markings, and it incurs a significant delay at each hop.

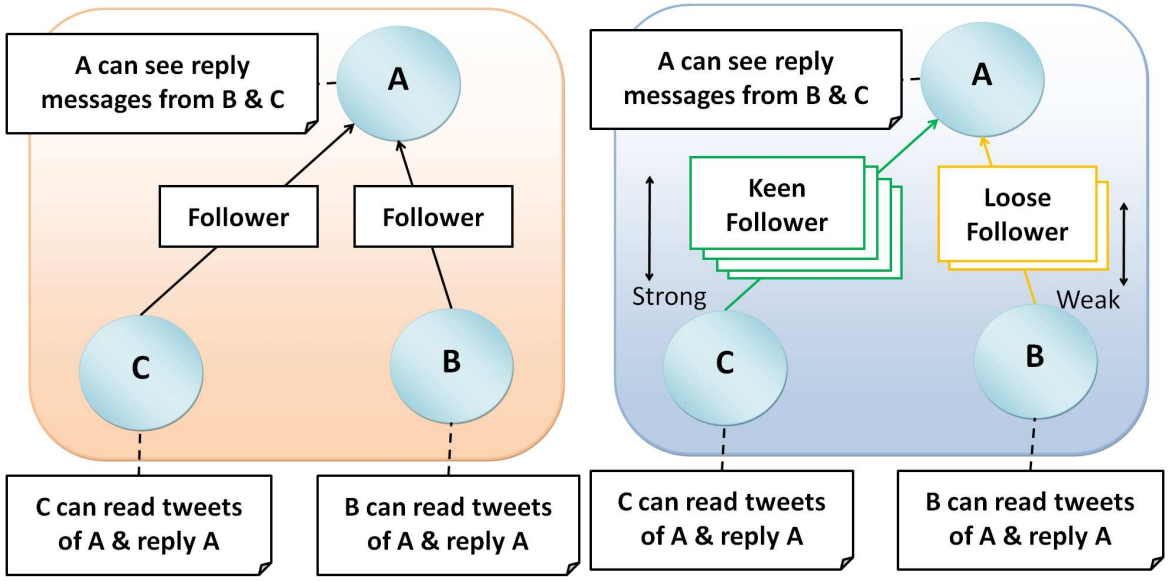
In a work on SN content analysis, [63] employ ISIS, a general stochastic model (with a set of sequential statistical tests) for Interacting Streaming Information Sources, to identify items that gather a higher attention in social media. In a similar application context, [19] identify messages dealing with the trending topics or special events in an SN using visualization techniques and artificial intelligence based data mining methods.

In the context of new challenges in this field related to privacy, background knowledge, and data utility, many researchers addressed the anonymization (user identity suppression) problem. [110] present a short but systematic review of the existing anonymization techniques for privacy preserving publishing of social network data. Considering trust between users in a social network as a parameter similar to the reputation of a specified user rather than a quantification of preference and profile matches between users, [53] propose a fuzzy logic based system to compute the trust values for individual users in an SN by propagation and aggregation through the network the trust values provided on a scale of 0 to 10 by the users about the other directly connected (hence well known) users. [57] address the problem of detection of spam among blogs, the SN media similar to a micro blogger like Twitter, but with more capability. The authors basically identify the spammers from the repetitive temporal regularity of contents and consistent linking patterns. The temporal regularity, in turn, is measured by using the entropy of the "blog post time" difference distribution. Experimental results indicate that a high accuracy of 90% in spam blog detection can be achieved by their method. [102] address the problem of identification of influential users in twitter using an improved page-ranking system called TwitterRank based on the concept of homophily, the tendency of individuals to associate and link with similar others, or those with similar topics of interests.

In the context of SN privacy, I address in this dissertation the problem of identification of types users in a twitter network. I categorize the twitter users into 4 types: i) Leaders (those like the news groups, who start tweeting, but do not follow any one there after, though they could have

many followers), ii) Lurkers, who are generally inactive, but occasionally follow some tweets, iii) spammers, the unwanted tweeters,also called as twammers, and iv) close associates, including friends, family members, relatives, colleagues, *etc*. Two classification approaches have been proposed here to address this problem: i) context-dependent classification for situations where an abundant amount of tweet data is available; here I employ a fuzzy classification scheme, and ii) context-independent classification (based on the user tweet patterns) that is suitable when enough information is not available about the network context.

Organization of the rest of the chapter is as follows. Section 7.2 presents my approach to context-dependent classification of twitter users. In section 7.3, I outline a context-independent classification approach to handle situations where sufficient information regarding twitter users under consideration is lacking. Details of experimentation and results are presented in section 7.4. Finally, summary is presented in section 7.5.



(a) Social network represented as a simple graph

(b) Social network represented as a graph with weighted links

FIGURE 7.1. Two graphical views of a social network.

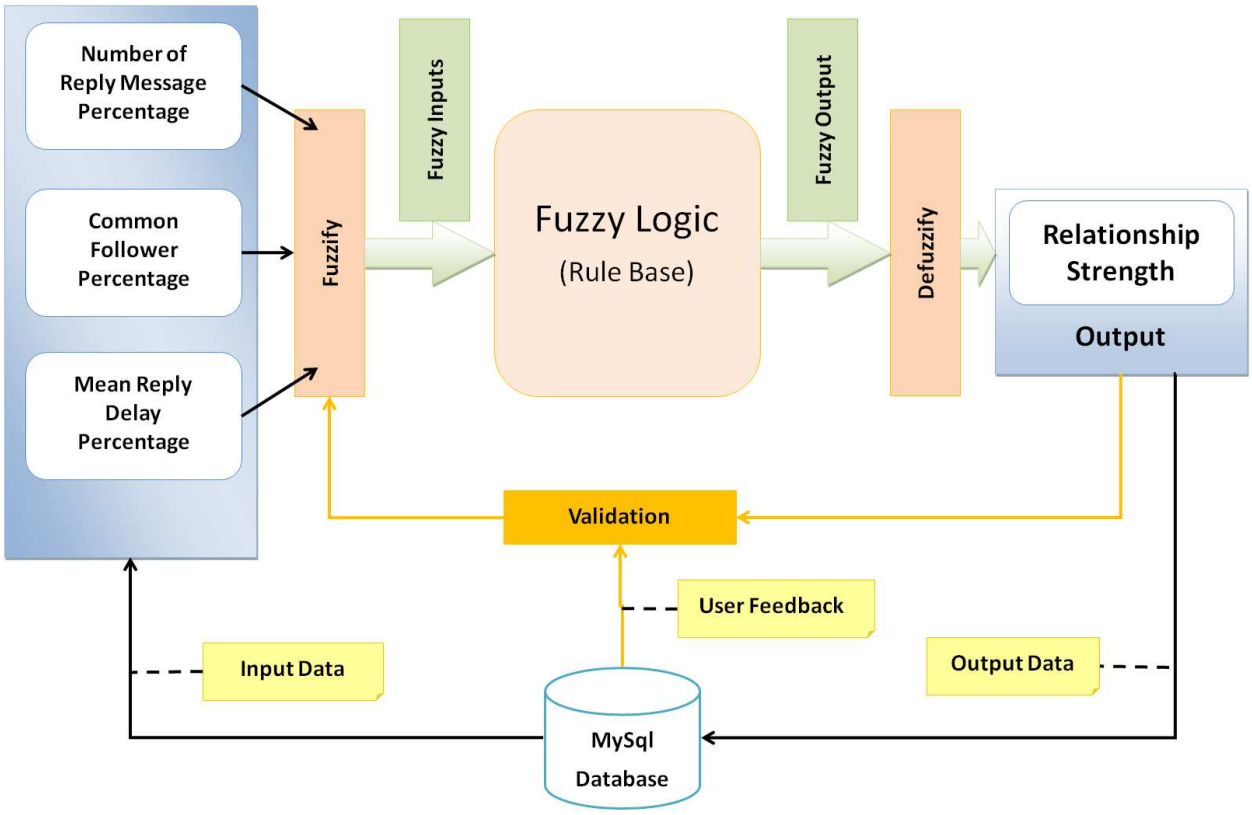


FIGURE 7.2. Proposed fuzzy system architecture for link (relationship) strength Evaluation.

7.2. Context-Dependent Classification of Twitter Users

Social networks are formed by social groups of people that are linked by social bond or relationship. In a group, one can follow another or one can be followed by the other. In the twitter jargon, those two types of individuals are called followers and followees, respectively. Unlike in the case of emails, the mutual relationships between individuals/groups in the twitter network can be tracked by monitoring the tweets. Even though it is possible to compile email message and response pairs, the sparseness of data there makes it difficult to estimate the strength of relationship between the corresponding individuals. In the twitter (generally, SN) domain, on the other hand, it is possible to estimate the SN link (relationship) strengths by using the followee-follower message statistics. At a gross level, it can be said that better the mutual communication, higher the link (relationship) strength. Further, if the link between two individuals is stronger, it is unlikely that either of them is a twammer.

Since most of the tweets occur between close associates, the twitter data for this group is generally overabundant compared to the other three groups, and this data imbalance poses problems for an identification of the users, particularly those belonging to sparsely populated classes. Hence, I follow a two stage process. In the first stage, I estimate the strength of links in the social network and eliminate a large number of users with strong social bond (or link strength) because they naturally classify into the group of close associates. Then, in the second stage, I perform a linear classification of the four user types mentioned in section 7.1, using number of tweets and the followee-follower ratio as two features considering only those tweeters with weak link strength (less than 15% of maximum strength) between them.

TABLE 7.1. Sample rules of the fuzzy rule base.

No.	Rule
1	IF replies IS VL AND common_followers IS VL AND mean_reply_time IS VL THEN rel_strength IS VL;
5	IF replies IS VL AND common_followers IS VL AND mean_reply_time IS VH THEN rel_strength IS L;
10	IF replies IS VL AND common_followers IS L AND mean_reply_time IS VH THEN rel_strength IS M;
25	IF replies IS VL AND common_followers IS VH AND mean_reply_time IS VH THEN rel_strength IS H;
75	IF replies IS M AND common_followers IS VH AND mean_reply_time IS VH THEN rel_strength IS VH;

For a good estimation of the link strengths, I make use of the fact that compared to an SN representation as a simple graph with nodes and links as shown in Fig. 7.1(a), its representation as a graph with weighted links as in Fig. 7.1(b) provides better insights into not only the twitter spam identification problem but also various other SN problems. However, since the relationship strength depends upon a vague concept such as the "keenness" of the followers, I develop here a fuzzy logic approach for estimation of the relationship (link) strength and implement it using jFuzzyLogic

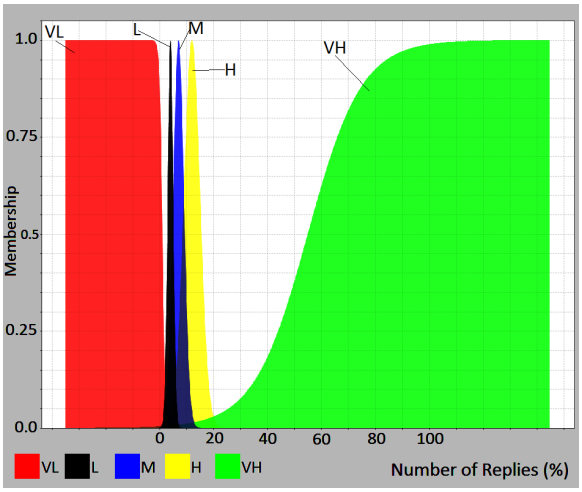
[21], an open source code in the Java language for the fuzzy control language (FCL) defined by the International Electrotechnical commission (IEC)'s standard 1131-7 [25]. Three parameters "Reply Message Percentage", "Common Follower Percentage", and "Normalized Mean Reply Delay" have been considered as indicators of the keenness with which a tweeter is followed, and hence used to constitute the input set of this system depicted in Fig. 7.2. Percentage of reply messages among the total messages and the promptness (or inversely the delay) with which a user responds are obviously indicators of the user keenness in following a conversation. Similarly, since common followers of the tweeters on either side of a link suggest a sharing of similar ideas or topic of interests between the tweeters, percentage of common followers among the total tweeter population is a good indicator of relationship strength. To bring the "mean reply delay" parameter into the range [0 100] just like the other two, I normalized it with a scaling factor that sets the maximum delay to 100. The fuzzifier module of the system performs fuzzy quantification of each one of the 3 inputs into 5 levels (linguistic terms/values)- Very Low (VL), Low (L), Medium (M), High (H), and Very High (VH), and determines for each input parameter the membership values in each category (level) based on the parameter distributions for each level that are pre-configured at the time of system setup. The fuzzy logic (rule base) processes the fuzzy inputs (level-membership tuples for the three input parameters) generated by the fuzzifier and generates a fuzzy output (level-membership tuple) for the relation (link) strength. The rule base itself is generated using a meta-rule (implemented in Java) that assigns integer values 0 through 4 for the linguistic terms VL though VH, respectively, and maps each input term triplet onto an output term. Specifically, the output linguistic variable will assume the values VL, L, M, H, VH for values of S, the sum of the integer values corresponding to the three input linguistic terms, in the ranges $0 \leq S \leq 2$, $2 < S \leq 4$, $4 < S \leq 7$, $7 < S \leq 9$, $9 < S \leq 12$, respectively. In Table 7.1 , I present 5 typical rules among the total 125 rules (for 3 input variable each with 5 possible linguistic values). The output linguistic values depends on the rules that were fired and the membership value of the output is determined by aggregating the membership values from the input level-membership tuples of the corresponding rule. Since the input variables can be members of different categories (levels) with different membership values, multiple rules could fire and yield multiple output level-membership

tuples, which are finally mapped onto a real number by the defuzzifier. The defuzzifier is pre-configured to use the well known center of gravity method for generating the crisp output (real number) from the linguistic term-membership tuples.

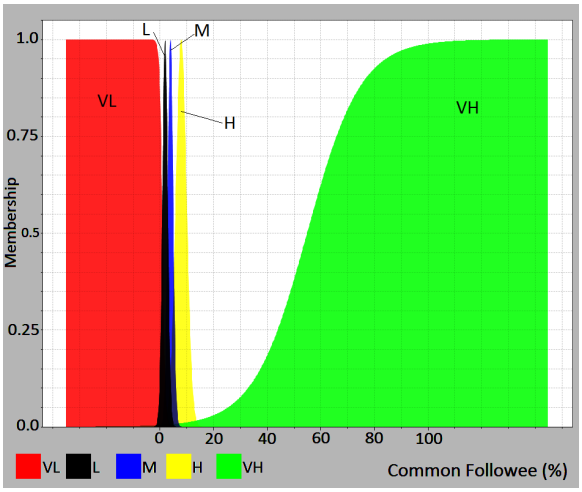
TABLE 7.2. Membership distribution functions for the input and output variables.

Fuzzy Sets→ Crisp Inputs/ Output↓	VL	L	M	H	VH
Input 1: Reply Message %	S(-2, 1)	G(4, 1)	G(7, 3)	G(12, 3)	S(0.1, 55)
Input 2: Com. Follower %	S(-2, 1)	G(2, 1)	G(4, 1)	G(8, 2)	S(0.1, 55)
Input 3: Mean Reply Time	S(-0.3, 25)	G(50, 10)	G(75, 5)	G(95, 2)	S(4, 100)
Output: Rel. Strength	G(0, 15)	G(25, 10)	G(50, 10)	G(75, 10)	G(100, 15)
Com. Follower ⇒Common Follower; Rel. Strength ⇒ Relational Strength					

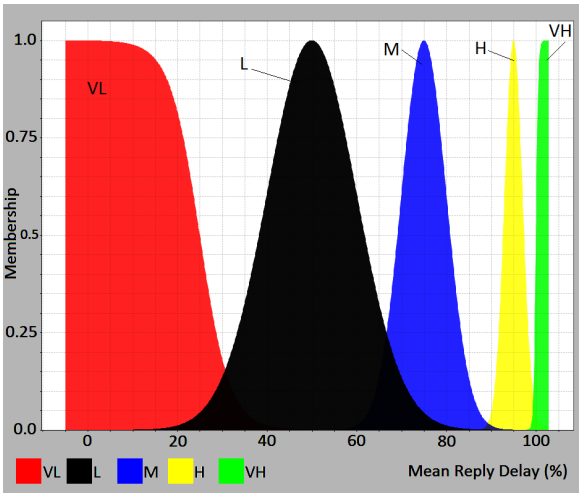
The jFuzzyLogic software facilitates the users to define the input/output parameter distributions for each one of the linguistic classes (terms) for determination of the class membership values based on the parameter values. For these distributions, it is possible to use either standard functions such the Gaussian or sigmoid or custom functions. I have used mostly Gaussian function except for the fringe VL and VH classes for the inputs where I used sigmoids. Table 7.2 gives the functions chosen for each one of the inputs and output, and the 5 linguistic classes. In this table, G represents the Gaussian distribution, and S, the sigmoid. For the Gaussian function, the two parameters represent mean and for the Sigmoid function, the two parameters are the gain and center, respectively. The distribution selection and parameter tuning is done by repeated trails to get a smooth and gradually increasing output. Fig. 7.3 depicts the membership functions for the



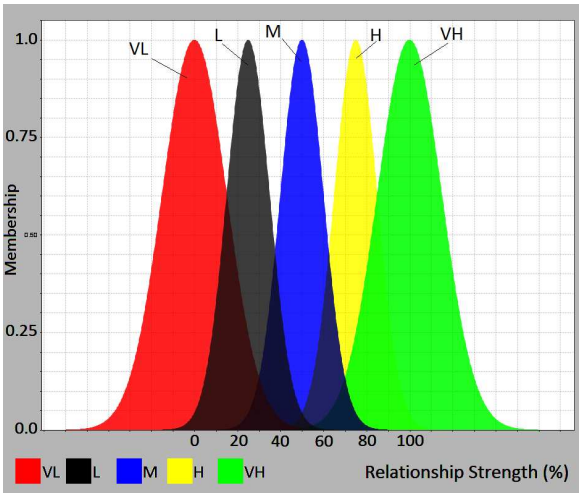
(a) Membership function for the "Reply" Message input



(b) Membership function for the "Common Follower" input



(c) Membership function for the "Mean Reply Delay" input



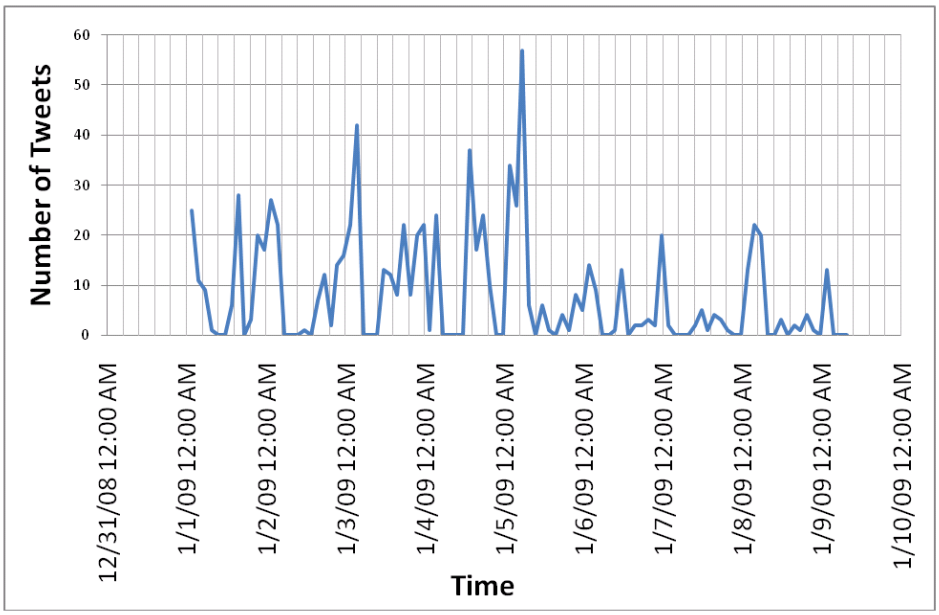
(d) Membership function for the "Relationship Strength" output

FIGURE 7.3. Membership functions for various linguistic classes of the 3 input and 1 output variable.

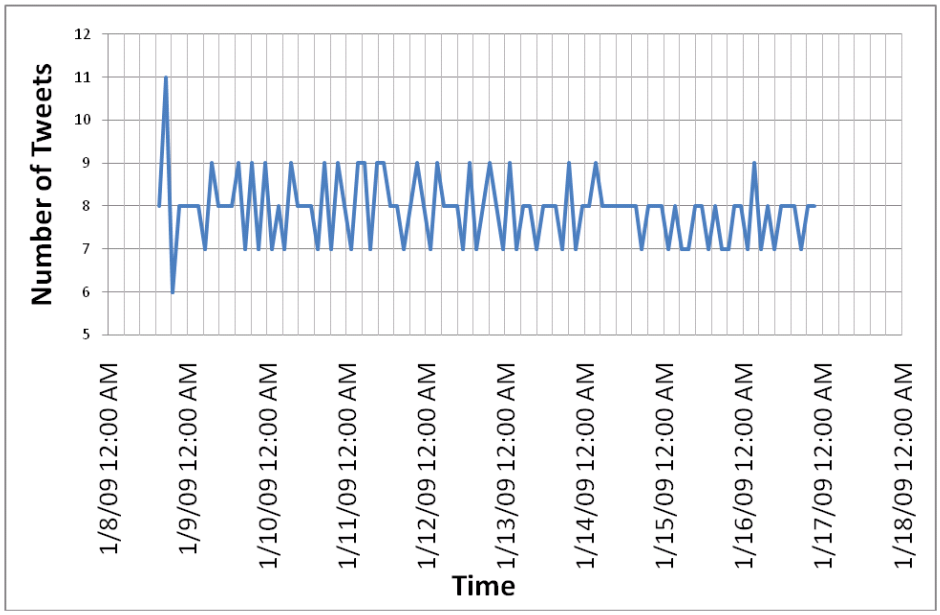
3 inputs and 1 output. It may noted here that though some graphs extended for the values the parameters beyond the range $[0\ 100]$, the membership values are computed only for values within the range.

7.3. Context-Independent Classification of Twitter Users

Oftentimes, extensive contextual information for the estimation of the relationship between two tweeters is not available. Particularly, the twammers (or tweet spammers) catch us by surprise



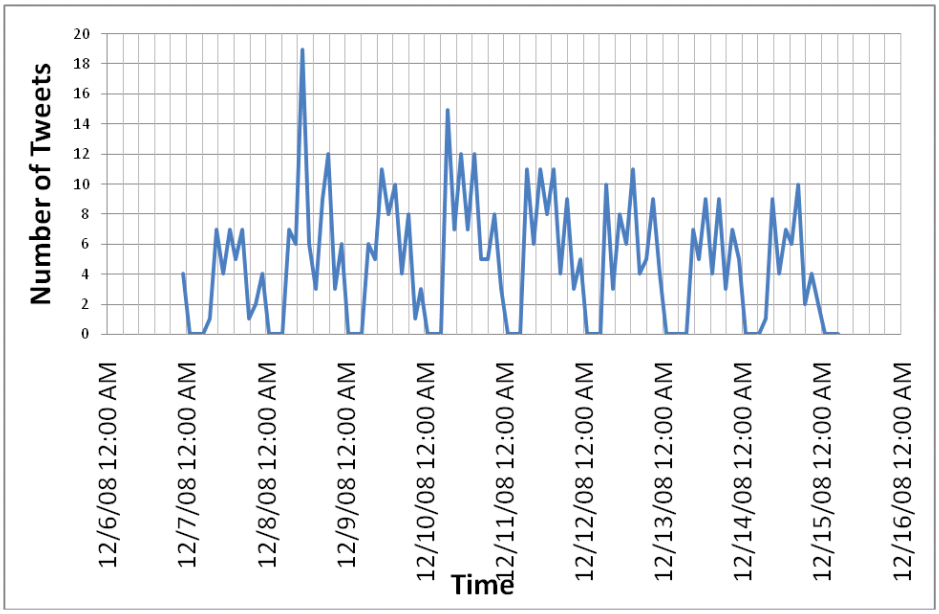
(a) Tweet patterns of a close associate



(b) Tweet pattern of a spammer

FIGURE 7.4. Tweet patterns of the three different classes of tweeters (though actual time series patterns extend over a period of 25 to 28 days, they have been truncated to 10 days here because periodicity is not visible on a compressed scale).

thereby rendering the approach detailed in section 7.2 useless. Hence, I propose in this section an approach for user (tweeter) type detection simply based on the generic tweeting patterns of the



(c) Tweet pattern of a news blogger

FIGURE 7.4. Tweet pattern of the three different classes of tweeters (cont.).

tweeters belonging to different tweeter classes. This approach derives support from the empirical evidence about the distinctive nature of the tweet patterns collected from different types of tweeters. Fig. 7.4 depicts the sample 10-day tweet patterns from the twitter database that was developed by a procedure described in the following section on experimental results. The tweet pattern shown in Fig. 7.4(a) is that of a normal tweeter, who could be a friend, colleague, or family member, and hence does not show much periodicity. The pattern shown in Fig. 7.4(b) is that of a spammer with user name "la_hora," who was spamming from a bot named "Ya son las 02:50!". It is worth noting the periodic nature of the pattern with no distinction between day time and night time segments. When the tweet messages were analyzed most of them were similar and meaningless. Tweeting occurred every 10 minutes, and hence nearly a constant number of messages were generated each day. Finally, the patern in Fig. 7.4(c) is that of a news blogger with USER-ID "spitsnet." I verified that this person is a news blogger by following the links. Further evidence comes from the observation that this person tweets in the dutch, and Sp!ts(pronounced Spits), according to Wikipedia [105], is a tabloid format newspaper freely distributed in trains, trams and buses in the Netherlands. It is interesting to notice that this pattern resembles a harmonic series with a very low number of

tweets in the midnight and a high number during the day time. Thus it is possible to attempt the same tweeter classification problem using a context independent approach of matching an unknown tweet pattern with the prototypes (or the labeled samples) from different classes. The user classes I consider here are the leaders (e.g. the news blogger), spammers, and associates. Since it is difficult to identify lurkers with sparsely available data, I consider only a three-class problem that excludes this class in the context-independent approach.

For email spam detection, the statistical Bayesian approach has been very popular, and is found to be effective. In a typical work in the Bayesian framework, Ma, Tran, and Sharma [61] used a technique called "Negative Selection" to detect spam email without any prior knowledge about the spam emails. Yeh and Chiang [108] carried out a re-evaluation on the Bayesian filter for email spam detection, and found that though it yields high accuracy on plain text email messages, it is not as effective with modern emails with multimedia content and message encoding. They suggest the use of a scheme combining different spam detection strategies. Thus, it may be seen that both the Bayes and MLP classifiers are competitive at least for the email spam detection problem. Hence, these two classifications tools have been considered for my investigations into the problem of user type detection based on the tweet patterns. I have included in this repertoire of classification tools, the random forest method developed by Breiman [13].

In the Bayes approach, $P(\mathcal{C}_i|\mathcal{X})$, the probability of a given tweet pattern vector $\mathcal{X}$ (formed by the time samples over a specific period of tweet frequencies) belonging to a pattern class $\mathcal{C}_i \in \{Associate, Spammer, Leader\}$ is given by:

$$\begin{aligned} P(\mathcal{C}_i|\mathcal{X}) &= \frac{P(\mathcal{X}|\mathcal{C}_i).P(\mathcal{C}_i)}{P(\mathcal{X})} \\ &= \frac{P(X_1, X_2, \ldots, X_n|\mathcal{C}_i).P(\mathcal{C}_i)}{P(X_1, X_2, \ldots, X_n)} \\ &= \frac{\prod_{j=1}^{n} P(X_j|\mathcal{C}_i).P(\mathcal{C}_i)}{P(X_1, X_2, \ldots, X_n)} \end{aligned} \tag{6}$$

In the above $\{X_1, X_2, \ldots, X_n\}$ are the components of the pattern vector $\mathcal{X}$, $P(\mathcal{C}_i)$ is the a priori probability of the class $\mathcal{C}_i$, and $P(\mathcal{X}) = P(X_1, X_2, \ldots, X_n)$ is the probability of the sample vector. The third line in the above equation is a consequence of the assumption that the compo-

nents of $\mathcal{X}$ are conditionally independent; a Bayes classifier using this assumption is termed as a naive Bayes classifier. Now, since $P(X_1, X_2, \ldots, X_n)$ does not depend upon the chosen class, the Bayes classification algorithm boils down to assignment of the pattern vector $\mathcal{X}$ to the class $\mathcal{C}_i$ for which $P(\mathcal{C}_i).\prod_{j=1}^{n} P(X_j|\mathcal{C}_i)$ is maximum. For this computation, it may be assumed that $P(X_j|\mathcal{C}_i)$ conforms to a normal distribution as follows:

$$P(X_j|\mathcal{C}_i) = \frac{1}{\sqrt{2\sigma_{ji}^2}} e^{-\frac{(X_j-\mu_{ji})^2}{2\sigma_{ji}^2}} \tag{7}$$

Here μ_{ji} and σ_{ji} are the mean and standard deviation, respectively, of the j^{th} component of the pattern vectors belonging to the class $\mathcal{C}_i$.

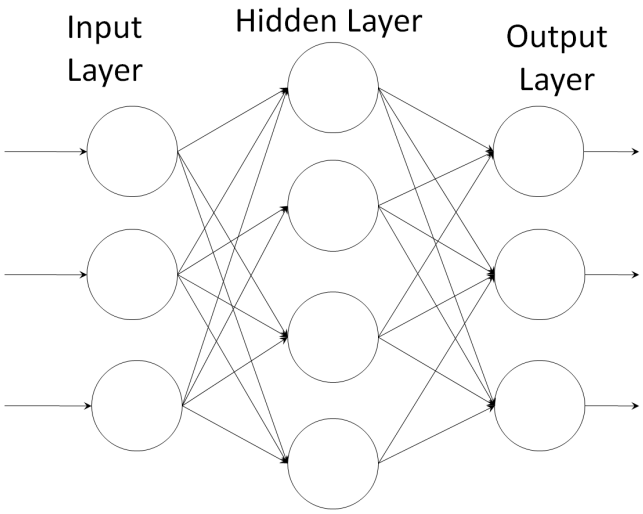


FIGURE 7.5. A typical three layered feed-forward network.

In the MLP approach, a feed-forward neural network [88] of the form shown in Fig. 7.5 is used. The circular elements are nonlinear (usually, sigmoid) functions representing neurons, and the lines are weights representing inter neuronal synapses. The leftmost layer of neural net is the input layer, and the rightmost is the output layer. The input layer can have as many neurons as the components of the pattern vector. The output neurons can be as many as the number (in this case, three) of classes in the classification problem. The neuron layers in between the input and output layers are called hidden layers; they are effective in producing good nonlinear mappings. During training phase, the pattern vectors with known class labels are presented at the input layer, and the

outputs are observed. All neurons but one in the output layer are expected to assume zero values; the one representing the input pattern's class should have a zero. If that does not happen, the weights connecting the neurons in the penultimate layer to those in the last (output) layer should be so adjusted so as to minimize the following mean square error function:

$$E = \sum_{k=1}^{n} (t_k - O_k)^2 \tag{8}$$

where t_k and O_k are the target (expected) and actually observed output values of the k^{th} output neuron. It can be shown that the local minima of this function can be achieved by gradient descent formulas that adjust the weights of neurons connecting the last layer first, then those connecting into the previous layer, and so on till those feeding into the first layer are modified. Since this weight training takes place in the backward direction from the output to input layers, this algorithm is known as the back-propagation learning algorithm [88]. Once the network is trained with all the training patterns, it is ready to classify the patterns with unknown class labels by producing high output at all but one of the output neurons.

According to Breiman [13], Random Forest is classifier formed by an ensemble of decision tree classifiers $\{h(\mathbf{X}, \Theta_k), k = 1, \cdots\}$ where the $\{\Theta_k\}$ are independent and identically distributed random vectors, and $\mathbf{X}$ is the input vector. The random forest classifies X into the class with maximum vote considering the votes for the most popular class at $\mathbf{X}$ by the constituent classifiers.

7.4. Experimental Results

TABLE 7.3. The Twitter database Statistics.

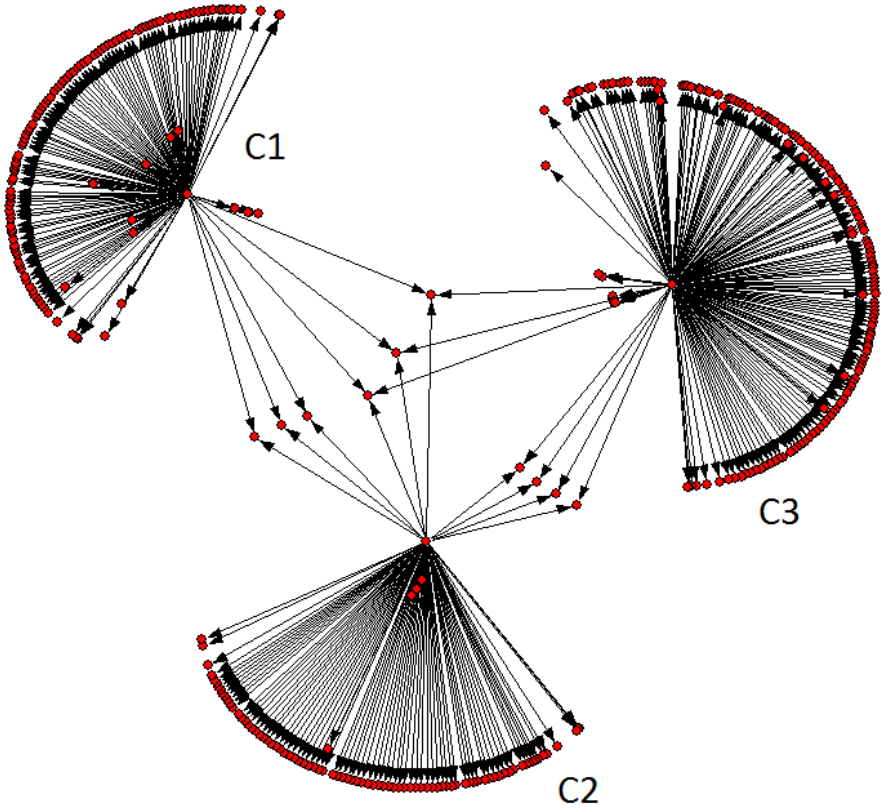
Total Users	Total Links	Total Tweets	Total Tweet Replies	Average Links/ User	Average Tweets/ User	Average Replies/ User
441234	2045804	6481900	2312927	5	15	5

7.4.1. Data Collection Procedure

The main requirement for this research is availability of a good data set that includes details of all the activities in the twitter network such as user profiles, the number of messages interchanged between the users, and the interactions between the users, etc. Since all the online social networks are based on real individuals, privacy settings make it very difficult to acquire a proper data set with all activities of each individual from the network. Most popular social network web-sites provide Application Programming Interfaces (APIs) for running their profile crawlers, but these are either restricted or need special permissions. However, I circumvented the privacy restrictions problem by simply programming the Twitter crawler to skip over the restricted users. Since the number of restricted users is very low compared to the total number of users in Twitter, this strategy is expected to have little effect on the results of the analysis. Another problem I faced while accessing twitter database is due to Twitter's API Rate Limiting policy which limits the number of requests per hour for the data records made through the API to 150. Luckily, this restriction could be waived to white-listed users with special permissions. I obtained these permissions from Twitter and could access as many as 20000 records per hour. Table 3 summarizes the statistics of the twitter database I developed using this process. However, since it is difficult to visualize such a huge network, I show only the results on 500-node subnet in subsection 7.4.2. Further the difficulties in hand-labeling the user types forced me to limit the experimentation only 528 randomly sampled tweeter records in subsection 7.4.3.

7.4.2. Results on Link Strength Determination and Context-Dependent Classification

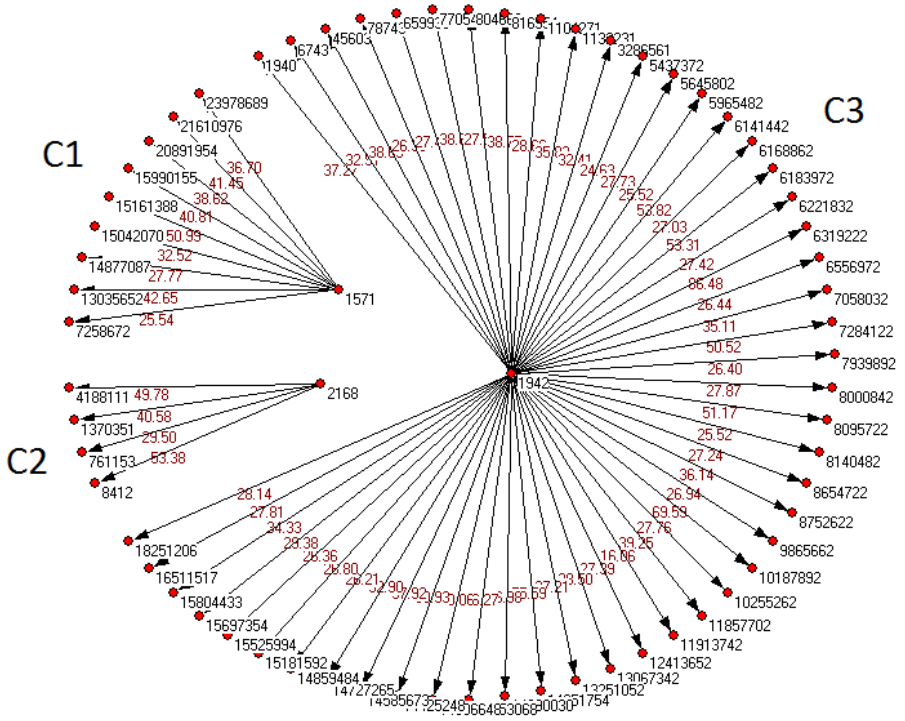
I determined the link strengths of a 500-node twitter network by applying the fuzzy logic based classification method discussed in section 7.2 on the twitter database developed by us. It is interesting to visualize some characteristic features of this network from its graphical depiction in Fig. 7.6. First, it can be observed that the network nodes form strong clusters, and the cluster structures don't change much when weak links are removed; this indicates that the same set of tweeters communicate frequently with one another though some are more involved than the other in tweeting. Next, since it is easy to infer that the tweeters with strong connectivity are close associates, the classification needs to be applied to the tweeters with low relationship strength. From



(a) Connectivity of a 500-node twitter network with all Links shown.

FIGURE 7.6. Changes in twitter network clustering with thresholding of links (the numbers at the nodes are the USER IDs and the numbers on the links are their strengths).

the figures 7.6(b) and 7.6(c), it is clear that a threshold of 15% (of maximum relational strength) for the link strength, is good enough to separate out the tweeters who are unambiguously close associates. Hence, I applied the simple linear classification algorithm using number of tweets and the followee-follower ratio as two components of the pattern vector only to the tweeters with the link strength below this threshold. Clear separation of the four tweeter classes as depicted in Fig. 7.7 suggests that this method holds promise for an effective identification of leaders, lurkers, spammers, and associates. In view of the enormity of the analysis required to be done for hand-labeling of the records for the purpose of validation of these classification results, I first formed a smaller validation set consisting of: i) the few records corresponding to the points in the leader, lurker, and spammer quadrants in the plot (Fig. 7.7), ii) those represented by the points on the boundary of the



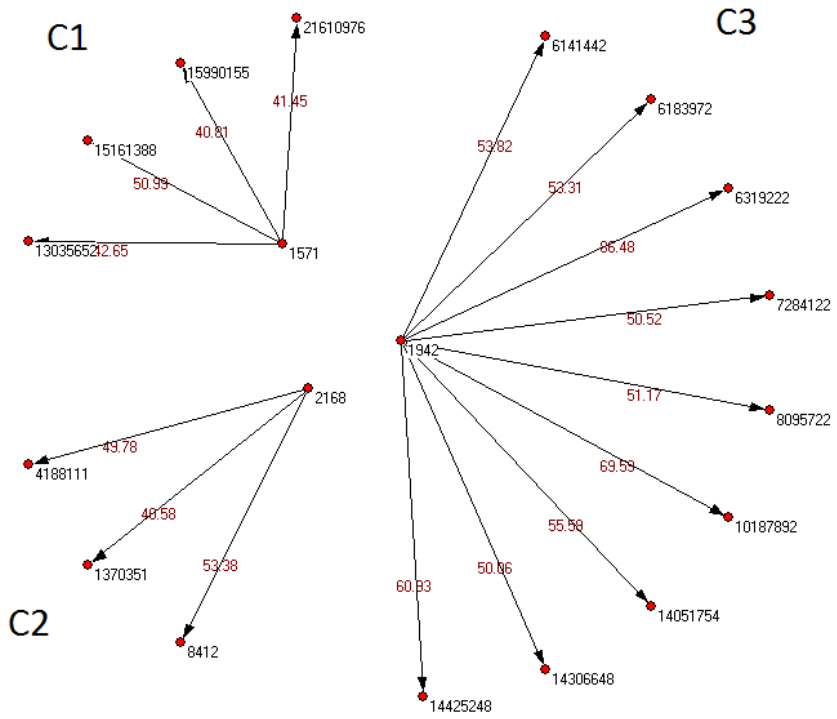
(b) Connectivity of the twitter network with only links having relationship strength above 15%.

FIGURE 7.6. Changes in twitter network clustering with thresholding of links (cont.).

"close associates" quadrant, and iii) those corresponding to a few randomly selected points in the middle of the "close associates" quadrant. I then hand-labeled each one of these records in the validation set by going through the profile information and the tweets contents. Finally, considering the hand labels of the records as the ground truth, I tallied the results of linear classification on the validation set with the ground truth. Validity of my classification approach has been established by a perfect tally.

7.4.3. Results of Context-Independent Classification

For this experimentation, I used a set of total 528 tweeting frequency patterns that includes 224 leaders (news groups), 164 spammers, and 140 associates. By using the 9 equally spaced time samples over a 10-day window as components, 10 dimensional pattern vectors are formed



(c) Connectivity of the twitter network with only links having relationship strength above 40%

FIGURE 7.6. Changes in twitter network clustering with thresholding of links (cont.).

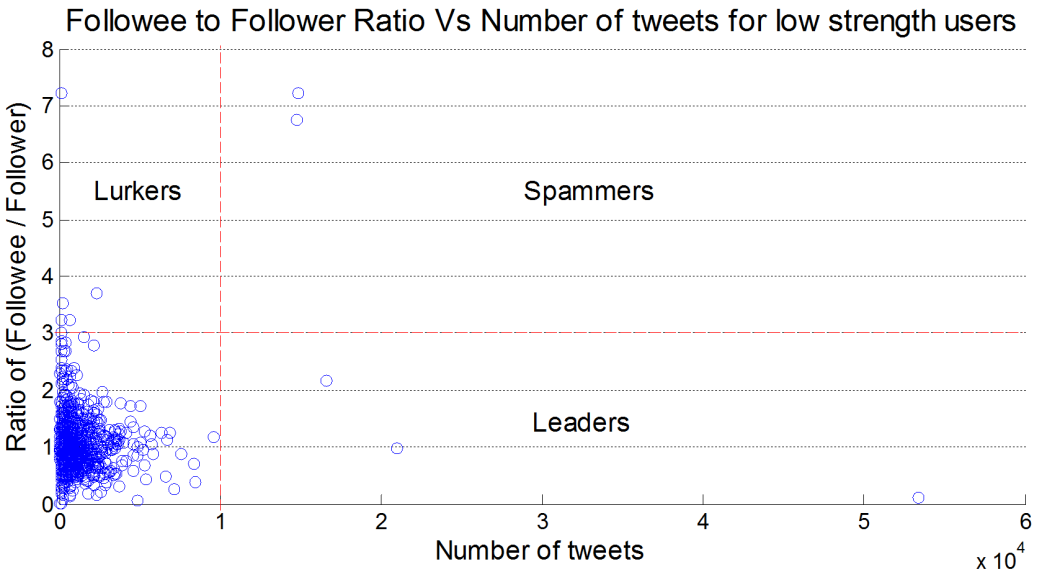


FIGURE 7.7. Linear separation of Leaders, Lurkers, Associates, and Spammers.

TABLE 7.4. Confusion matrices of the three classifiers in the TCV test procedure.

	Naive Bayesian Confusion Matrix			MLP Confusion Matrix			Random Forest Confusion Matrix		
Class↓→	L	S	A	L	S	A	L	S	A
Leaders(L)	224	0	0	220	4	0	223	1	0
Spammers(S)	0	163	1	1	161	2	2	162	0
Associates(A)	0	0	140	4	2	134	0	0	140
Classification Accuracy→	99.81%			97.54%			99.43%		

TABLE 7.5. Confusion matrices of the three classifiers in the R66T test procedure.

	Naive Bayesian Confusion Matrix			MLP Confusion Matrix			Random Forest Confusion Matrix		
Class↓→	L	S	A	L	S	A	L	S	A
Leaders(L)	75	0	0	73	1	1	75	0	0
Spammers(S)	0	56	0	0	55	1	0	56	0
Associates(A)	0	0	49	1	2	46	0	1	48
Classification Accuracy→	100%			96.67%			99.44%		

for each one of the 528 data records. For the MLP I have used 9 neurons (corresponding to the components of the pattern vector) in the input layer, 11 neurons in the hidden layer, and 3 neurons (corresponding to the 3 classes) in the output layer.

Since the classification results depend not only on the classifier chosen, but also on the training procedure adopted, I have considered three training/test procedures as follows for this experimentation: i) Ten-fold cross validation (TCV) method- Here the data is split into 10 sub-parts, and each sub-part is used in a round robin fashion as the test set with the remaining nine as training sets, ii)R66T method where randomly Selected 66% records form the training set and the rest are used for testing, and iii) O66T method where the first (oldest) 66% of the time ordered records form

TABLE 7.6. Confusion matrices of the three classifiers in the O66T test procedure.

	Naive Bayesian Confusion Matrix			MLP Confusion Matrix			Random Forest Confusion Matrix		
Class↓→	L	S	A	L	S	A	L	S	A
Leaders(L)	30	6	20	56	0	0	0	0	56
Spammers(S)	26	29	1	0	56	0	1	55	0
Associates(A)	2	0	66	20	0	48	54	0	14
Classification Accuracy→	69.44%			88.89%			38.33%		

the training set and the rest are used for testing; this method is an implementation of the strategy of applying old experience to new situations. The classifiers used for experimentation, as discussed in section 7.3 are the naive-Bayes (NB) classifier, multi-layer percetron (MLP), and and random forest (RF) classifier. In the experiments with each classifier-training procedure combination, I obtained the confusion matrices and classification accuracies for each one of the three classifiers under the above three test scenarios. Further, since classification accuracy does not reflect the correct performance when the data set is imbalanced with far more samples in some groups than the other, I employed the popular F-measure that is used in the database (DB) literature to measure the efficacy of retrieving the correct DB records. In case information retrieval, the records intended for retrieval as one class, and the rest are considered as irrelevant. The relevant records retrieved by an algorithm are considered as true positive (TP). Irrelevant records retrieved are considered as true negatives (TN) and the relevant records not retrieved are considered as false negatives (FN). Finally, the irrelevant records not retrieved are denoted as false positive (FP). The proportion of records retrieved from the total set of relevant records is termed as TP rate or recall (R), and the proportion of relevant records among the total set of records correctly processed (retrieved or omitted) by a data retrieval algorithm is called precision (P). The $Fmeasure$ is the harmonic mean of

TABLE 7.7. F-measure computations for the three classifiers in the TCV test procedure.

Class↓	Naive Bayes			
	TPR	FPR	P	F
Leaders	1	0	1	1
Spammers	0.994	0	1	0.997
Associates	1	0.003	0.993	0.996
Weighted Averages→	0.998	0.001	0.998	0.998
	MLT			
	TPR	FPR	P	F
Leaders	0.982	0.016	0.978	0.98
Spammers	0.982	0.016	0.964	0.973
Associates	0.957	0.005	0.985	0.971
Weighted Averages→	0.975	0.013	0.976	0.975
	Random Forest			
	TPR	FPR	P	F
Leaders	0.96	0.007	0.991	0.973
Spammers	0.988	0.003	0.994	0.991
Associates	1	0	1	1
Weighted Averages→	0.994	0.04	0.994	0.994

Note: TPR, FPR, P and F represent true positive rate, false positive rate, precision, and F-measure respectively.

TABLE 7.8. F-measure computations for the three classifiers in the R66T test procedure.

Class↓	Naive Bayes			
	TPR	FPR	P	F
Leaders	1	0	1	1
Spammers	1	0	1	1
Associates	1	0	1	1
Weighted Averages→	1	0	1	1
	MLT			
	TPR	FPR	P	F
Leaders	0.973	0.01	0.986	0.98
Spammers	0.982	0.024	0.948	0.965
Associates	0.939	0.015	0.958	0.948
Weighted Averages→	0.967	0.016	0.967	0.967
	Random Forest			
	TPR	FPR	P	F
Leaders	1	0	1	1
Spammers	1	0.008	0.982	0.991
Associates	0.98	0	1	0.99
Weighted Averages→	0.994	0.003	0.995	0.994

Note: TPR, FPR, P and F represent true positive rate, false positive rate, precision, and F-measure respectively.

TABLE 7.9. F-measure computations for the three classifiers in the O66T test procedure.

Class↓	Naive Bayes			
	TPR	FPR	P	F
Leaders	0.536	0.226	0.517	0.526
Spammers	0.518	0.048	0.829	0.637
Associates	0.971	0.188	0.759	0.852
Weighted Averages→	0.694	0.156	0.705	0.684
	MLT			
	TPR	FPR	P	F
Leaders	1	0.161	0.737	0.848
Spammers	1	0	1	1
Associates	0.706	0	1	0.828
Weighted Averages→	0.889	0.05	0.918	0.888
	Random Forest			
	TPR	FPR	P	F
Leaders	0	0.444	0	0
Spammers	0.982	0	1	0.991
Associates	0.206	0.5	0.2	0.203
Weighted Averages→	0.383	0.327	0.387	0.385

Note: TPR, FPR, P and F represent true positive rate, false positive rate, precision, and F-measure respectively.

recall and precision. These relations can be stated mathematically as follows:

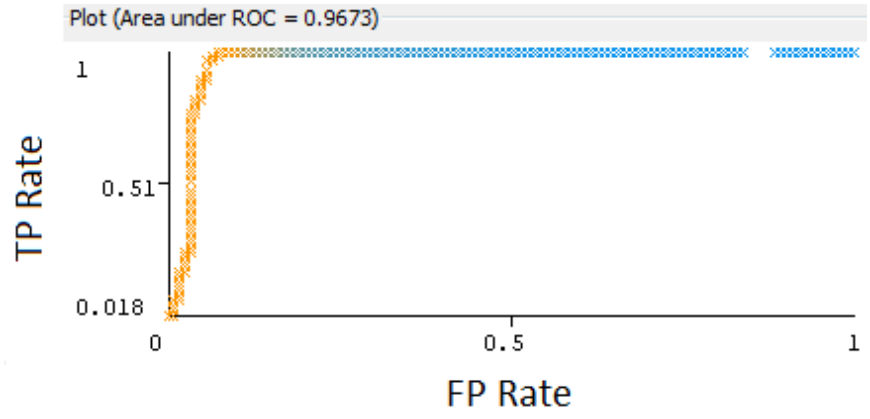
$$
\begin{aligned}
R &= \frac{TP}{(TP+FN)} \\
P &= \frac{TP}{(TP+FP)} \\
Fmeasure &= \frac{2}{(\frac{1}{R}+\frac{1}{P})} = \frac{2RP}{(R+P)}
\end{aligned} \tag{9}
$$

For adapting the F-measure to measure classification performance, I considered successively each one of the three user classes (i.e. leaders, spammers, and associates) as the relevant class and treated the remaining two together as a single irrelevant class. For different classifier and test method combinations, the TP, TN, FP, and FN are then be obtained from the corresponding confusion matrices, and the F-measures are computed. Ultimately, the overall F-measures are computed by an weighted average scheme in which individual class F-measures are given weights (in the interval [0 1]) proportional to the number of samples in the corresponding classes.

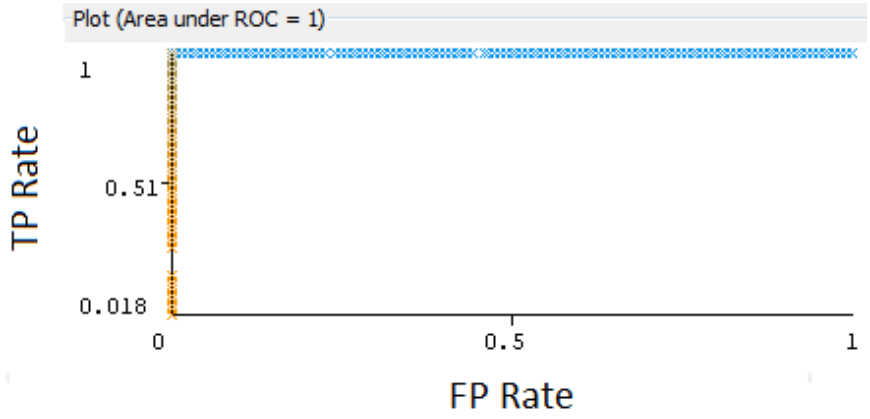
Tables 7.4, 7.5, and 7.6, respectively, present the confusion matrices for the 3 classifiers under the TCV, R66T and O66T test procedures described above. These results indicate that MLP outperforms the other two classifiers with respect to classification accuracy when the O66T test procedure is used, though its results are slightly shy of those with the other classifiers when the TCV and R66T test procedures are used.

Tables 7.7, 7.8, and 7.9 summarize the results of F-measure computations for the three classifiers under the TCV, R66T, and O66T procedures, respectively, using the confusion matrix data in tables 7.4, 7.5, and 7.6. These results include the F-measures for individual classes (lumping together the remaining two classes as an irrelevant class) as well as the weighted average measure computed by applying to the data of each class a weight proportional to its population in the training set. On F-measure also, the MLP exhibits a very good performance compared to the other two classifiers with O66T test procedure, and yields competitive results under the remaining two test procedures.The performance of the Random Forest method under the O66T procedure is below par.

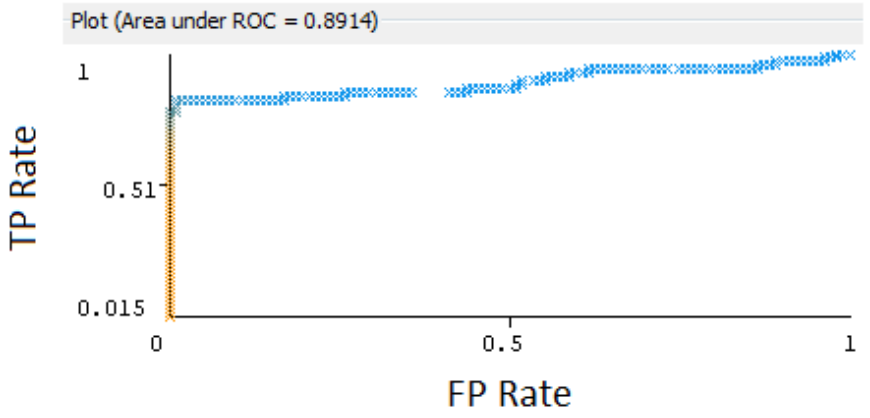
The ROC curves shown in Fig. 7.8 for the three user classes for the best classifier (i.e.



(a) ROC Curve for the "Leaders" Class



(b) ROC Curve for the "Spammers" Class



(c) ROC Curve for the "Associates" Class

FIGURE 7.8. ROC curves for the three classes of Users when MLP is used under the O66T Procedure.

MLP) under the most challenging test procedure (i.e. O66T) establish the viability of context-independent classification procedure.

7.5. Summary

In this chapter, I present two classification methods for twitter network user identification: i) context-dependent (data-heavy) method which employs a fuzzy logic approach to estimation inter-user relationship strengths in the first step and then a linear classifier to separate out the four user classes, and ii)context-independent method that uses tradition classifiers and generic user tweet patterns to distinguish between the users in situations where the availability of user information is limited.

This research enforces the conventional wisdom that spammers follow a large number of people (followees), but they themselves are followed by very few people. Specifically, as evidenced by the results of subsection 7.4.2, spammers are defined by the accounts that make more than 10,000 tweets in a 10-day interval (or equivalently over an average of 1000 tweets a day) and have a followee-to-follower ratio of 1.5 to 1 or more. The twitter leaders, on the other hand, can be distinguished by their high rate of tweeting, large number of followers, but a few, if any, followees, and hence by a followee-to-follower ratio much below 1. Close associated are marked by strong connectivity to their followers, low to moderate number of tweets (1000) per day, and small to moderate (less than 3) followee-follow ratio. Finally, lurkers is a rare class of tweeters, who follow many people, but they themselves rarely post or reply any tweets.

The results on the more challenging problem of classifying users with limited data, the MLP classifier has been found to outperform the naive Bayesian and Random Forest classifiers when the procedure of classifying the new patterns with the old patterns is adopted. Since that is how new spammers are identified generally, the MLP can be considered as an effective context-independent approach to spam filtering.

CHAPTER 8

USER-INTENTION AND CONTEXT-AWARE MULTIMEDIA ENCRYPTION

In a phone conversation, context-awareness refers to the identification of the conversation topic. Knowing what is being conversed will help determine the sensitivity of the conversation. If the spoken content in the conversation is sensitive, one can encrypt it while the unsensitive information could be encrypted with relaxed encryption algorithms or avoid encryption entirely. Phone resources can be saved by doing so, and this is the outline of this chapter.

To enforce this mechanism, it is important to know the context and the content of the conversation. The context can be identified by using the intention of the conversation (I.e. the intention of the user). If the user has spoken a sensitive word, the system must recognize the intention of that word as "to converse a sensitive information". Based on this intention, the system could perform the enhanced audio speech data encryption. This chapter illustrates how to utilize the user-intention in a mobile conversation to improve the performance of an existing encryption protocol. The user-intention in a phone conversation is referred to as context-awareness throughout this chapter.

8.1. Introduction

Voice over IP (VoIP) has been successfully implemented in many platforms and its pros and cons are discussed extensively in the research community [36]. With the growth of the smart phone industry, VoIP technology is inevitably becoming popular among the mobile phone users. However, since VoIP is an IP based communication, it suffers similar Internet security vulnerabilities and threats [29]. Many security features have been implemented to protect VoIP conversations, including utilization of strong encryption and scrambling [71]. In contrast, enforcing such exhaustive security measures could be expensive on mobile platforms which have limited resources and

The content in this chapter is reproduced from Mohamed Fazeen, Garima Bajwa, and Ram Dantu, "Context-aware multimedia encryption in mobile platforms", Proceedings of the 9th Annual Cyber and Information Security Research Conference (New York, NY, USA), CISR 14, ACM, pp. 53−56, 2014. DOI=10.1145/2602087.2602115 http://doi.acm.org/10.1145/2602087.2602115, with permission from ACM.

power. Here, I am exploring an efficient secure channel of VoIP communication on mobile platform that could enable context and content aware encryption. My intention is to save unnecessary CPU utilization during VoIP data encryption. This, in turn can save CPU time, battery power and other resources like memory, I/O, etc.

Context-aware systems are "Systems that adapt to the context of the user, application and their communication and computation environment, as well as to the changes to the context information over time" [73]. In this regard, the context of a conversation such as talking to a banker on a phone and providing the social security number are of main interest here.

For simplicity, I restrict this context-aware encryption method to VoIP content in this paper. However, this context-aware approach can be extended to other types of multimedia content, such as video, text, and pictures.

8.1.1. Motivation

Audio data in a mobile phone is the digitized analog audio information. Generally telephone conversations and wide-band speech data are sampled at a frequency of 8-16 KHz with a resolution of 8-bits (1 Byte). Thus, 5 minutes of audio data at least includes $5 \times 60 \times 8000 = 320000$ Bytes of information. Say, if I use the AES-128bit (Rijndael) with cipher-block chaining (CBC) to encrypt this data, it will take about 16.0 rounds per Byte to encrypt data [27]. Therefore, it would take about $320000 \times 16 = 5120000$ number of AES cycles to encrypt 5 minutes of speech. In terms of computation, this consumes a lot of processing time, energy and other resources.

Further, encrypting audio speech data is expensive than the same information that was stored in words (letters). For instance, in average a human can talk about 100-150 words per minute [7, 48]. By assuming the speaker speaks continuously for 5 minutes, during this talk time he/she will talk about $150 \times 5 = 750$ words. In English each word caries an average of 4.5 letters per word [80]. Thus, if the same speech put in words and letters, it caries about $150 \times 5 \times 4.5 = 3375$ letters. Since each ASCII character can be represented with 8-bit or 1 Byte the same speech can be represented with 3375 Bytes. The audio speech to character representation of the same duration speech ratio is about $320000/3375 \approx 95$. Its almost 100 time efficient than uncompressed audio. Thus compared to text data, audio data uses more data storage though the amount of information for

the speech session is about the same. Hence, encrypting all the audio data is expensive compared to that of in the text form. Especially computationally expensive encryption algorithms like AES will be inefficient such magnitude of data. It will be worse in mobile devices because of limited the power and other resources. Therefore, it needs a better mechanism to perform speech audio encryption with existing encryption algorithm.

However, human conversations carry less amount of sensitive information than the total speech. For instance, non-sensitive words (such as hi, hello, that, this, etc.), pauses between words, silence during listening on a telephone conversation are such non-sensitive information. Encrypting such information with computationally expensive algorithms is not necessary and we one can cut the cost by using less expensive algorithms.

According to M. Kennedy et al., in a smart phone, CPU is the second largest energy consumer next to the screen. Their results clearly indicate that CPU power consumption increases with its usage and it increases sharply when it exceeds 80% [49]. Therefore, saving CPU usage directly can save the power consumption. Further, in a VoIP technology survey by S. Karapantazis & F. Pavlidou claims Voice Activity Detection (VAD) can save 30-35% of the bandwidth by simply blocking blank voice activities[47]. My scheme explores further into the speech content to provide a more efficient encryption mechanism in VoIP to save many such resources in a mobile phone.

8.1.2. Problem Definition

The goal of this work is to implement a VoIP data encryption mechanism that identifies sensitive and nonsensitive data based on the context and content, and encrypts them differently to reduce the computation cycles. By doing so, I intend to conserve energy, CPU time and other resources while still preserving the security of the data on the mobile platform. Further, I aim to modify the existing sRTP protocols to enable the context-aware encryption mechanism.

8.2. Proposed Solution

The proposed algorithm consists of 5 phases. Each phase explains a sequence of operations performed on the voice data when it is transmitted from the sender to the receiver. Phase 1 to 4 provides a half-duplex channel and to achieve a full-duplex communication, two half-duplex

channels are used.

Phases one and two will intercept the audio data and analyze them for sensitive contents. Once sensitive information is detected, phases three and four will encrypt the data, construct the custom VoIP packets and transmit them via public channels. At the receiving end, the VoIP data will be captured and sorted into high- or low- strength encryption packets. Then the encrypted data are decrypted with the appropriate key and algorithm and converted into audio format. In phase five performance analyses will be performed.

Following are the 5 phases of the proposed architecture:

Phase 1: Intercepts the audio data and implements the word extractor module using a speech recognition engine to extract the text words from the audio speech data.

Phase 2: Identify sensitive words by using context and content in the extracted words. Ex: Numbers, Passwords, and similar sensitive words.

Phase 3: The sensitive information is encrypted with high-strength (Resource intense. Ex: AES, 3DES, RSA) algorithm while non-sensitive information is encrypted with low-strength (Less resource intense. Ex: Blowfish, DES, TEA, etc) algorithm and is transmitted through the network as data packets. *Note that I always encrypt the audio data (or words in audio form), not the recognized text words.* The recognized text words are only used to determine if the audio words are sensitive or not. It is always easier to process text data than audio data due to the difference in information content (bits).

Phase 4: Once the receiver gets the data, it is reconstructed into high- and low- strength encrypted data packets and decrypted accordingly. Then they are merged to reconstruct the audio voice stream. (In case of a less sensitive context and more performance demand, one can avoid the low-strength encryption. However, this can open the system to several attack vectors. See section 8.3)

Phase 5: Measure the performance by varying parameters.

The proposed architecture is depicted in Figure 8.1. The Figure 8.2 illustrates how sensitive parts of the speech data are encrypted, transmitted and decrypted.

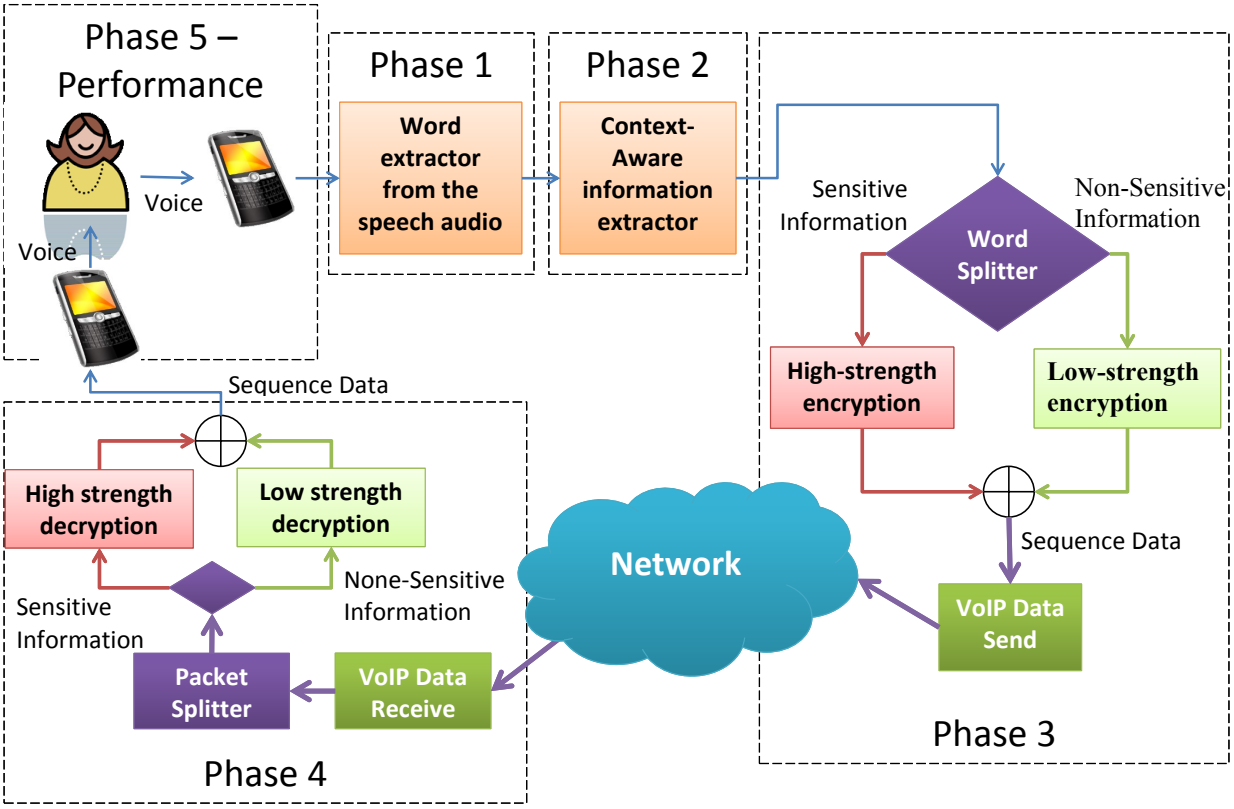


FIGURE 8.1. Proposed architecture.

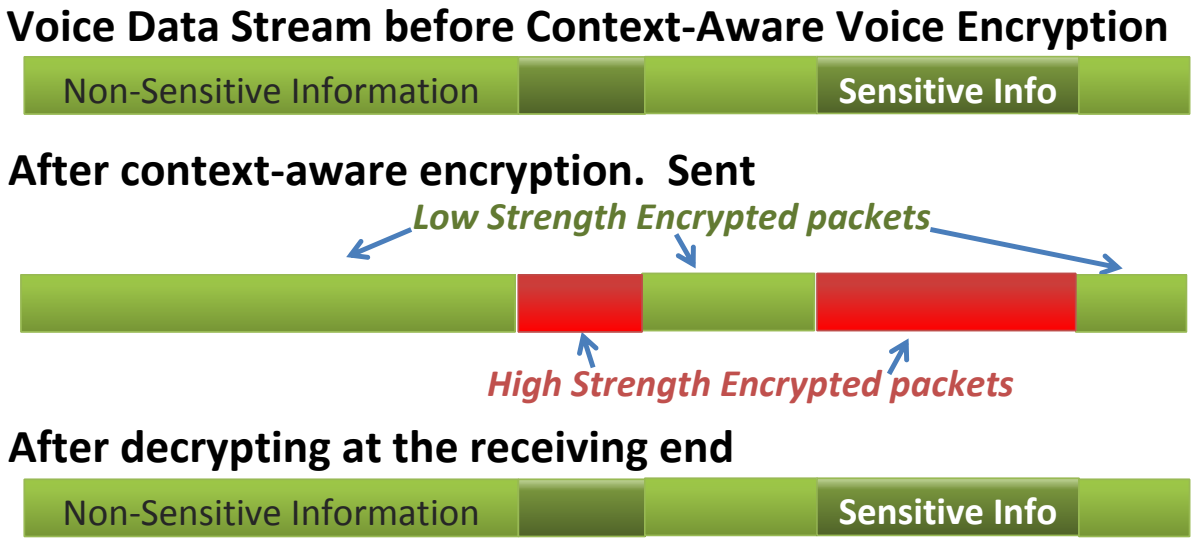


FIGURE 8.2. Context-aware speech audio data encryption and decryption.

8.2.1. Speech Recognition on Mobile Platform

One key challenge of this work is to convert the audio-representation of the speech information into text-representation in real time via a speech recognition tool. Such potential speech

engines are: Built-in speech recognition engine in the Android platform, "Dragon AudioMining" by Nuance [97], and CMUSphinx[1]. CMUSphinx is utilized in my algorithm due to its performance and portability. Also, CMUSphinx stores its databases and dictionaries in the phone storage (standalone) and does not need any remote access to process speech recognition.

8.2.2. Proposed Solution Vs Existing Solutions

Existing protocols like Secure Real-time Transport Protocol (sRTP) utilize the fulltime encryption of the data. The proposed approach uses two different algorithms: intense encryption for sensitive data and relaxed (or computationally light weight) algorithm for non-sensitive data. I cannot enforce the proposed scheme using sRTP, as it does not directly support a multiple encryption scheme [60]. Building a custom protocol is a lucrative alternative and can be used to improve the existing protocols. The key advantage over the existing method is resource saving.

Further, the system also can be used to totally block the sensitive information being transferred from the source. The content can also be controlled based on the recipient by using the caller id. For instance, a set of sensitive words being blocked to a person A may vary from person B and this could be pre-defined.

In recent mobile systems, manufacturers tend to equip mobile devices with dedicated crypto-coprocessors. These processors are designed with tamper resistant shielding to provide energy-efficient encryption system [6]. If these processors are provided, I could utilize these co-processors in my approach to save further resources.

8.3. Potential Attack Vectors and Solutions

In this section I identify potential attack vectors and their counter mechanisms to discuss the usability of this algorithm.

8.3.1. Sign-Post Sensitive Content Attack

In case of only encrypting the sensitive content and leaving non-sensitive information unencrypted (lets say mode 1), sensitive information will be marked as sign-posts. Hence, instead of needing to analyze all VOIP streams for the sensitive ones, the network adversary can just look for the ones which are encrypted.

Counter Attack: To prevent this type of attack I encrypt both sensitive and non-sensitive information, but with different strengths. This will leave the adversary to identify what cypher is corresponding to what algorithm which is not an easy task. Thus, it will simulate the regular encryption (full encryption). Further, as a precaution measure, one can encrypt the packet header with a low strength encryption to prevent any exposure of the packets identity. Since, the header contains a boolean and to increase the entropy of the cypher of the header, one can concatenate a random number to the header before encryption.

8.3.2. Known-Message Attack

1. : When the system is in mode 1 (See 8.3.1), due to some false negatives, there could be cases where a word can exist in the VOIP stream in both encrypted and unencrypted form. That's a starting point for crypto-analysis.

2. : Also, because of Kerkhoff's principle, one needs to assume that everything apart from the encryption key is public. Thus, network adversary could have access to the dictionary of sensitive words of the scheme. Therefore, the scheme is open to multiple known message and dictionary attack vectors.

Counter Attack: The first attack can be easily prevented by applying the same solution as in section "Sign-Post Sensitive Content Attack". Thus, any false negatives are also encrypted with less strength and not directly accessible to the adversary.

For the second attack I propose two solutions. The first solution is to store the sensitive words as its hashed digest in the database and then encrypt the database. Even if an adversary breaks the encryption, the words cannot be identified as it is in its digest format (similar to password storage). Further, this hashing provides faster searching of the word.

As a second solution, a random time spacing could be added for the sensitive words as shown in equation 10. Thus, the cypher will be different for the same word. Let t_l be the time length of the audio word and T_{upper} be the maximum time that can be added to a word. Thus, I find the time length of the word to be encrypted as t_e

(10) $$t_e = t_l + t_r \text{ where } t_r \text{ - is a random time value}$$

(11) $0 \leqslant t_r \leqslant T_{upper}$ and, $T_{upper} <$ Average word time length

8.3.3. Known-Context Attack

This attack vector also opens up when the system is in mode 1 (See 8.3.1). Human speech does not encode sensitive data in single words, but sensitivity may arise from the entire context of the sentence. In fact, in many cases BEEP-ing out single word from a stream of a conversation will quite often not protect the conversation's secrecy and the unencrypted section could reveal the majority of the secrecy or otherwise provide crypto analysis hint for the adversary.

Counter Attack: This can also be prevented by applying the same solution as in "Sign-Post Sensitive Content Attack" section.

8.3.4. Malleable and Replay Attack (DoS)

Again when the system is in mode 1 (See 8.3.1), the selective encryption of sensitive words introduces attacks along the lines of malleability and replay that an active network adversary may resort to. For instance, the adversary could repeat encrypted packets out of context mangle cipher-texts rendering sensitive content unusable to the very least and thereby have an easy option for DoS.

Counter Attack: Solution in section 8.3.1 can be used as a first line of defense in this type of attack. For further security, one can encrypt the audio data + the hash digest (SHA-2) of the audio data in the sensitive-words-data-packets. If the cypher is tampered, it can be verified with this approach. Once the tampering is detected the connection could be dropped before the system over loads.

8.3.5. Man in the Middle Attack (MIM)

Most of the above attack vectors are different types of MIM attacks. Packet sniffing is a main approach in MIM attack.

Counter Attack: To reduce the risk of packet analyzing in MIM attack, I do not include any encryption details in the data packet except the cypher text. In case of open header, even if it

carries a boolean that does not specify what encryption algorithm it corresponds to. Both sender and receiver should agree upon the encryption protocols before transmitting any data.

8.4. Prototype

An application prototype was built to demonstrate the proposed algorithm. This Android application uses CMU Sphinx as the speech recognition tool [1]. The recognition accuracy was enhanced by carefully selecting a proper acoustic model and appropriate (Number dictionary) dictionary model.

Also, for the encryption the AES-128 bit algorithm is used. For the demonstration purpose, I only encrypted the sensitive information with AES while non-sensitive section was left unencrypted. However, in an ideal scenario, the non-sensitive information will be encrypted with a less intense algorithm. Again, for the simplicity and demonstration purpose I only used numbers as sensitive information. In the future, I will select the sensitive words list not only based on the context but also based on the recipient. In this prototype, a half-duplex communication was demonstrated and two different Android devices were used, one as the audio data sender and the other as the receiver. The screenshots of the sender and receiver applications are shown in the Figure 8.3.

8.5. Results

Results are presented in terms of 'sensitive words detection performance' and 'encryption performance'. All these applications were tested on a Samsung Note II smart phone. Figure 8.4 depicts the performance of the sensitive words detection and Table 8.1 and Figure 8.5 depict the performance of the context-aware encryption and regular encryption.

8.6. Related Work

In news broadcasting station, slipping an undesired word is very critical. Since most news broadcastings are aired live to the public, there is no much time to correct the mistake either. More to the fact that a bad mistake can degrade the reputation of the broadcasting service as well as the news anchor. The US patent number US7437290 B2, invented by Damon V. Danieli, introduced a novel method of censoring undesired words in speech data by an automatic-censoring-filter that

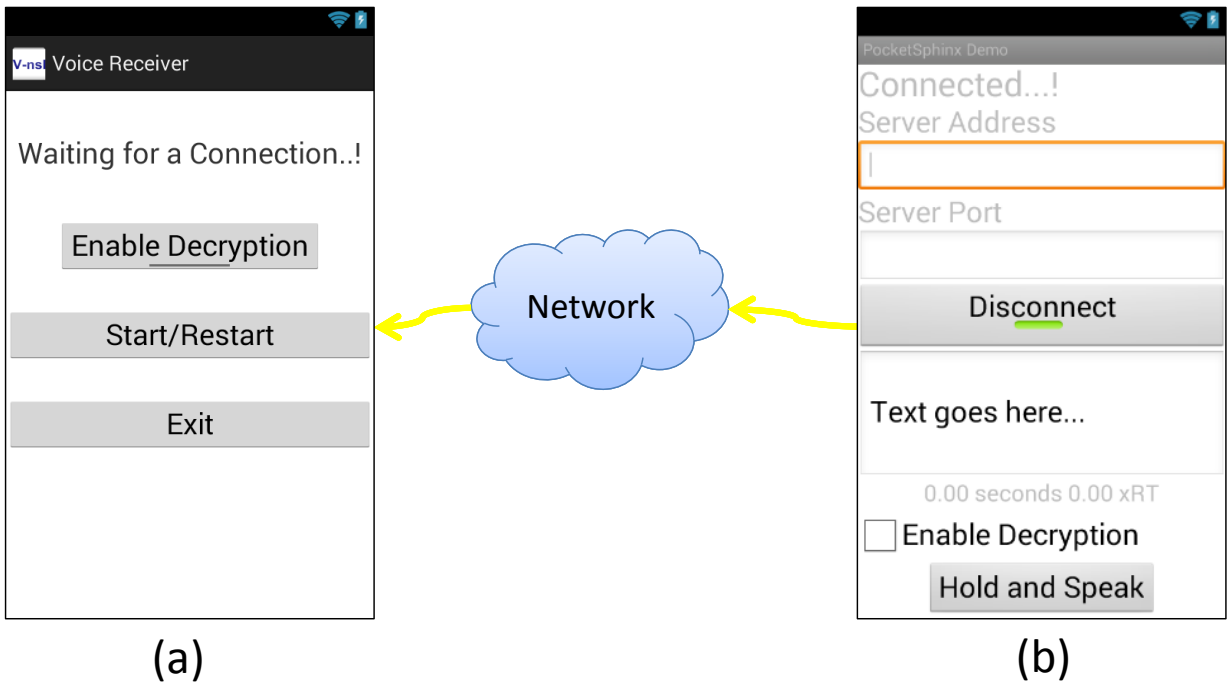


FIGURE 8.3. Screenshots of the implemented prototypes - (a) The receivers screen. (b) The voice sender's screen.

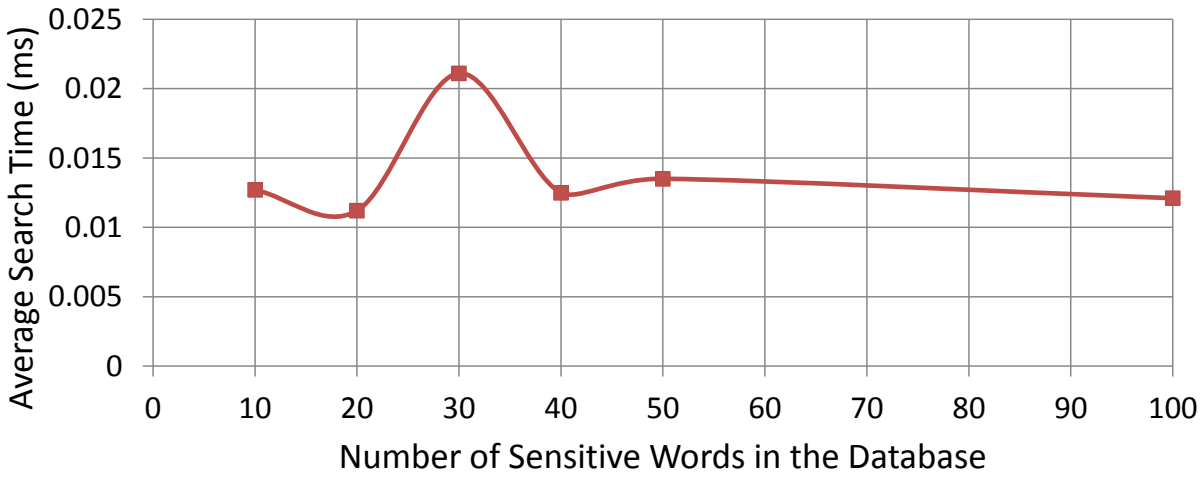


FIGURE 8.4. Average sensitive-words search time per detected word when the sensitive words database size is changing.

can be online in either real time or in batch mode [28]. His method also uses a similar method as proposed in this work where mispronounced phonemes and/or words were recognized and altered so that it cannot be audible or intangible. According his method these censored words either can be stored or made available for an audience in real time. In this approach, I am using a similar

Trial No.	Average time delay per buffer† (ms)	
	FULL Encryption	Context-Aware Encryption
1	11.3	6.7
2	9.1	6.1
3	11.3	5.4
4	8.5	6.8
5	10.4	5.7
Average	10.12	6.14

TABLE 8.1. The average processing time delay per buffer for full and context-aware encryption. Each recording was measured by speaking numbers from 1 to 5 two times. In the Context-aware encryption only uttering number 5 was considered as sensitive and encrypted. (†Buffer size = 256 bytes)

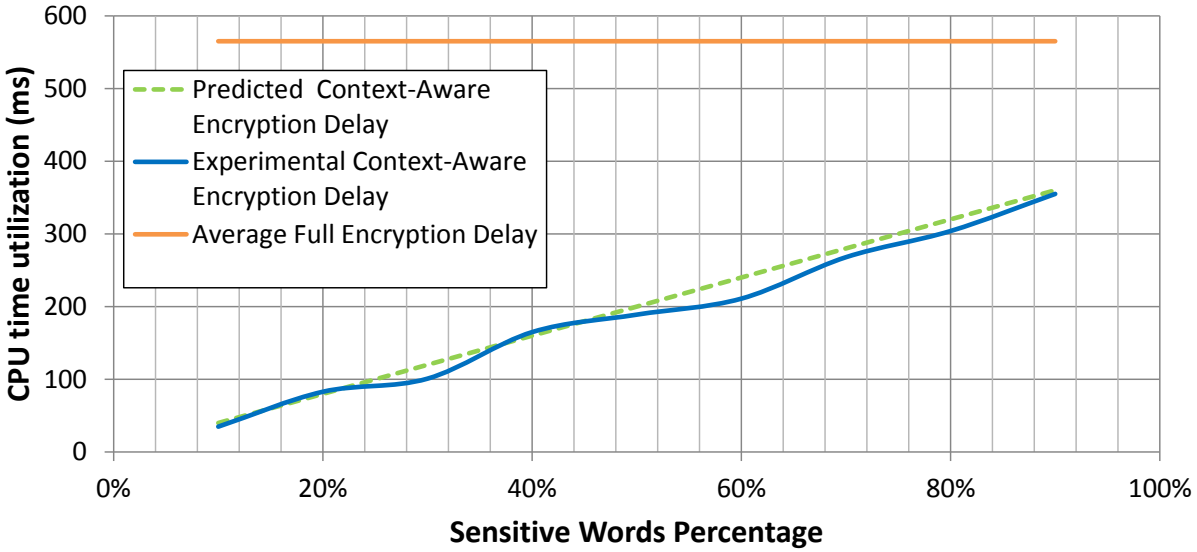


FIGURE 8.5. CPU time utilization when fully encrypting and context-aware encrypting.

approach to select the sensitive information and utilize it in a different environment in which, on an online phone call, in a more resource constrained mobile phone and use them to apply encryption algorithms. So that the data can be secured while conserving mobile phone resources and energy utilization.

An input audio data stream comprising speech is processed by an automatic censoring filter in either a real-time mode, or a batch mode, producing censored speech that has been altered so that

undesired words or phrases are either unintelligible or inaudible. The automatic censoring filter employs a lattice comprising either phonemes and/or words derived from phonemes for comparison against corresponding phonemes or words included in undesired speech data. If the probability that a phoneme or word in the input audio data stream matches a corresponding phoneme or word in the undesired speech data is greater than a probability threshold, the input audio data stream is altered so that the undesired word or a phrase comprising a plurality of such words is unintelligible or inaudible. The censored speech can either be stored or made available to an audience in real-time.

A work from a collaboration of scientists from King Saud university, Indiana University and University of Illinois at Urbana-Champaign worked on the title Context and Location-Aware Encryption for Pervasive Computing Environments. The paper begins with description about what is context-aware computing and its applications. The researchers approach was to decouple the context from the identity such a way that it can incorporate additional security features, value-added services and efficient key management for group communication. Their approach was to identify the location of a node (server or client) using Bluetooth and then authorize the user to access a resource. ARE with 128-bit and 265-bit key sizes were used to encrypt the data. [5]

Another example of context-aware computing was demonstrated by the MIT Media Lab under the paper title AreWeThereYet? - A Temporally Aware Media Player. The system provides an intelligent audio player that can calculate and estimate the available listening time by adjusting the listening rate or selecting play-list using listeners current location and their predicted destination to. [3]

“Context-Aware Computing with Sound” was a title of a work done by a team at the computer laboratory of University of Cambridge. They also discussed about context aware audio processing. This work is about, audio networking that uses available sound hardware (i.e. speakers, sound-cards and microphones) for low-bandwidth, wireless networking. The method is unique in a way such that it uses a data transmission system via audio signals. [62]

A patent from Carey Nachenberg explains about content aware data encryption. The patent claims for content aware encryption of text or audio data. If the data is audio, the author claims a

speech-to-text approach to obtain the text and then apply the encryption as for the text data. An encryption policy is determined by the contents. Based on the content and the policy that apply the content will determine the encryption method. The methods of encryption key claims were Advanced Encryption Standard (AES) key, a Blowfish key, a Data Encryption Standard (DES) key, a Bluefish key and an IDEA key, and the public key is as a Diffie-Hillman encryption key, an RSA encryption key or an Elliptic Curve encryption key. [67]

Another patent form the inventor L. Y. Gomez claims that ”receiving context information relating to the context of the applications; generating a key pair using the provided context information from the applications; and sending the generated key pair to the at least two applications.”. Here the inventor claims the content information as some location information such as environmental, temporal, user or computing information. Here key pair generation involves context information and the idea behind this is, context based keys can provide additional security. [42]

In an article about VoIP security in Security Management magazine, J. Wagley explains about how vulnerable close VoIP system are. He claims that ”Eurojust, announced that it was considering spearheading an effort to help its 27 constituent countries find a way to listen in on encrypted VoIP”. According to the author one of the popular VoIP provider Skype also suspicious in providing back door to the law enforcement. Phil Zimmerman, the inventor of Pretty Good Privacy (PGP) crypto system, invented and an open source VoIP software solution named Zfone. ”Making a product open source is one of the best ways to ensure that it doesnt have any backdoors, says Peter Eckersley, staff technologist at the San Francisco-based Electronic Frontier Foundation.”. [98]

In the work ’P3: Toward Privacy-Preserving Photo Sharing’ of R. Ra et al. describes how an important section of an image can be encrypted while leaving sufficient unencrypted data so that it can be utilized in cloud file sharing services [83]. In general, when an image is encrypted, it cannot be used or none of its meta information is visible until its being decrypted. Thus, such encrypted images cannot be rendered in a cloud file sharing services such as FaceBook, Google+, Flickr etc. unless they are decrypted. The novel idea in R. Ra et al. is that the image is split into

two section called public and private where public image contain only the vital meta information about the picture sufficient to identify, render and resize in a cloud file sharing server while hiding all the vital content information. The secret image contains the hidden vital content information and if the image needed to be viewed, this secret image has to be merged with the public image to reconstruct the full image. The public and secret image division was performed based on the AC and DC components of the image threshold at a ”sweet spot” [83].

It is very important to note that, the partial encryption described in this algorithm does not refer to encrypting cropped section of an image. Rather it is the division of meta information and its content. Though the algorithm described in the paper relates to this work, there are several fundamental dissimilarities between these two projects. In the partial photo encryption, image is being split into two parts regardless of its content, compared to that of in this project, the audio files are segmented based on its content. Further, both images needed to put back the original image while in this project encrypted and unencrypted sections are independent.

The Symantec corp. introduces a secure way of communicating through messages with ”Symantec Content Encryption for Symantec Messaging Gateway” [96]. The Sysmantec Content Encryption will monitor any outgoing email and its content for user defined sensitive information. User specify the sensitive information via policy management. Once sensitive information is detected it will encrypt the message before transmitted to the intended recipient.

The key similarity in this project and Symantec model is that, both methods encrypt the data based on the content. However, the Symantec model encrypts the whole document when sensitive data is detected while in this project only the sensitive information is encrypted. Further, the obvious contrast is that text document (emails) vs audio data where audio data are streaming data while emails can be considered as static data.

8.7. Future Work

In the demonstrated model sensitive words are predefined. This can be improved by storing each sensitive words in a tree structure, each branch related to certain context. This will enable the app to retrieve sensitive words dynamically based on the uttered context.

In a future version, sensitive words will be hashed and encrypted before storage to prevent certain attacks.

Elliptic curve cryptography (ECC) attracts mobile security providers due to its less computational complexity and encryption strength. I will replace AES with ECC for enhanced performance and better security.

8.8. Summary

It is also possible to toggle the encryption manually providing an encryption on/off switch to the end user. User can flip the switch manually as desired. However, it is possible that the user can slip a word mistakenly with sensitive information and it can be too late by the time the user realizes it. With this model it automatically detects the sensitive words without the user's explicit intervention to protect the user's privacy and security. User-intention is utilized to achieve this capability. This gives an added security and peace of mind in telephone conversations while saving energy by avoiding unnecessary encryption.

One can also use this method to simply block out the sensitive words from being transmitted out of the mobile phone. I.e. once a sensitive information is detected, instead of encrypting the data and transmit it to the other end, simply block that word being transmitted. This way, the model can protect the user from unintentional utterance of sensitive or bad words. In this way it can address the security vulnerabilities which occur due to unintentional speech.

In conclusion, the overall model gave a better performance in encrypting selected sensitive speech data in real time. Also, it can be used as a shield to protect and block sensitive information before sending it to the recipient. Thus, it can be used as a sensitive word filter in a phone conversation. Overall, this section describes a VoIP data encryption protocol that could be applied in a mobile platform which can save processing time and in turn battery life and other resources. Further, I assume that this will be beneficial for all mobile phone users who are concerned for their privacy and security during a phone call, and while saving battery life to provide an increase in conversation time as well.

CHAPTER 9

FUTURE WORK: EEG BASED USER-INTENTION IDENTIFICATION FOR AUTHENTICATION

Identifying the intention is a key aspect in this work, whether it is the intention of the app or the intention of the user. In the recent past, EEG technology has advanced in a way that there were many portable EEG devices produced to the market for an affordable price. These devices can be paired with a smart phone to connect human brain signals to the phone. See appendix A for how to obtain EEG data into an Android smart phone. With the right equipment, these data can be processed to obtain many vital information about the users mind, including his or her intention. That is the user-intention when an app is used by a user. What if I can identify the user-intention by users brain feedback, without the users voluntary involvement in reporting his/her perception about the app? As discussed in previous chapters, user-intention could be utilized in different ways to provide security for a mobile system. In this chapter, I emphasis the idea of EEG based user-intentions identification, rather than looking into its applications.

First, to understand the concept of EEG, let's take look at where it originates from.

9.1. Structure of the Human Brain

User motor activities are closely bound with brain activities. In L. King's book, Experience Psychology, the structure of the brain and its functions are described in detail. In chapter 2, King describes an individual's LEFT prefrontal cortex is more active than the right when the subject is stimulated with activities such as watching an amusing video. In this particular experiment, researchers used videos like a puppy playing with a flowers, and monkeys taking bath as amusing stimuli. On the other hand, clips that provokes fear or disgusts such as a leg amputating showed generally more activities on the right prefrontal area of the brain. [54]. This observation implies that not only does the human brain divide its task into the left and right sides of the brain, but also that each side processes different types of tasks.

From birth, the human brain is basically divide into three parts, Forebrain, Midbrain and Hindbrain. Hindbrain is located at the rear-most part of the brain and consists of the medulla,

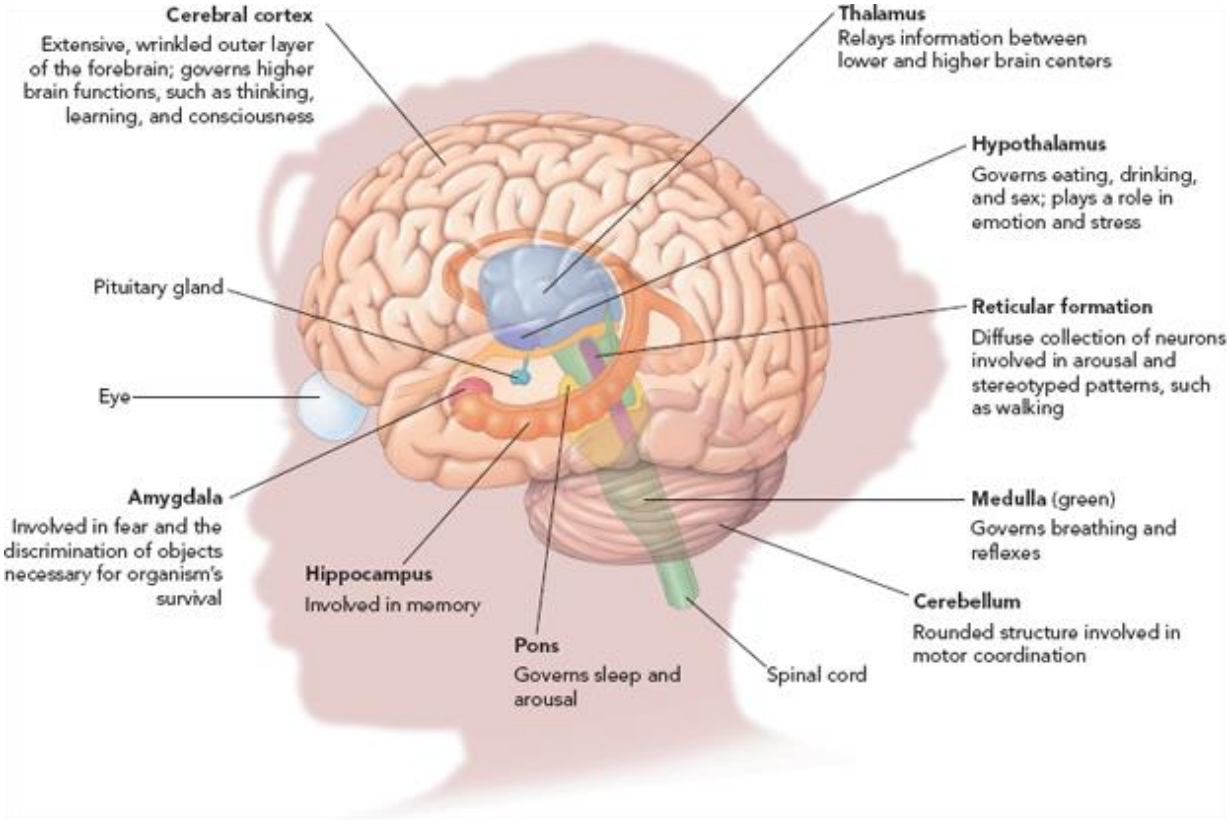


FIGURE 9.1. Anatomy of the brain and their tasks in brief. Pic. by L. King. [54]

cerebellum, and pons [54]. See Figure 9.1. Midbrain is literally the middle part of the brain, located between hindbrain and the forebrain. Midbrain consist of the reticular formation system and a system of neurons that consist of some specific neurotransmitters. The frontal part of the brain known as the forebrain is the larger portion of the brain, and consists of structures like the limbic system, thalamus, basal ganglia, hypothalamus, and cerebral cortex. The functionality of each of these structures and their location in the brain are briefly explained in the Figure 9.1.

Based on the above structure, forebrain activities are the center of attention in this research. The human brain is a very complex system. One motor activity of human behavior could activate many parts of the brain and coordinate by itself to perform various tasks. However, with the above description, one can realize that different parts of the brain are responsible for different tasks. Therefore, it can also be studied on by looking at its functionality. In such a division, the cerebral cortex (the wrinkled surface) consists of two hemispheres. Each hemisphere is divided into four lobes named as occipital, temporal, frontal, and parietal. See Figure 9.2. Each of the lobe is

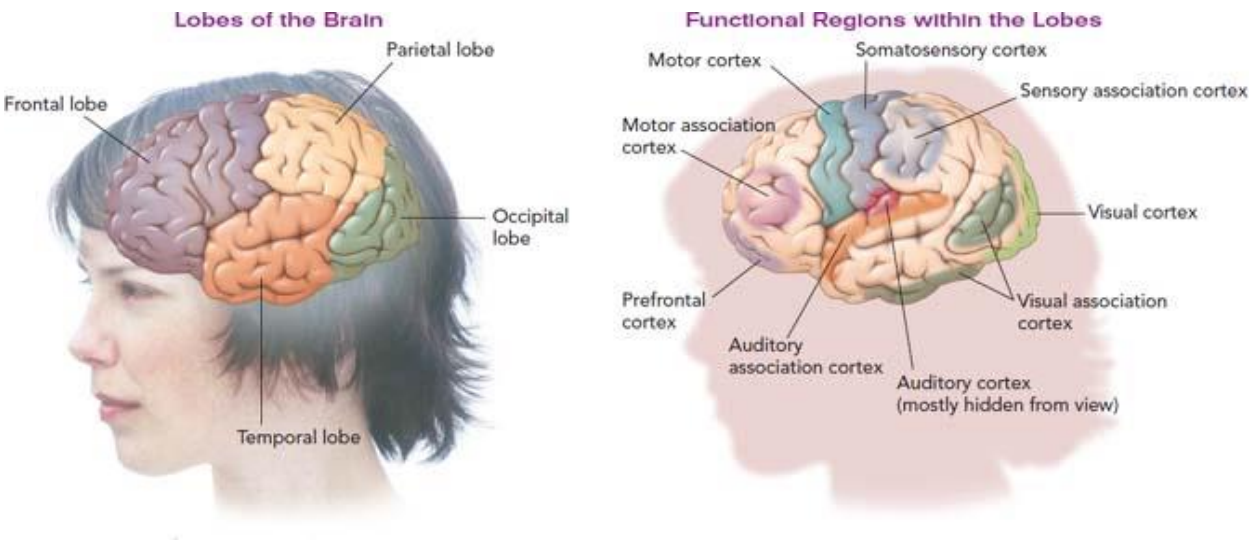


FIGURE 9.2. Division of the cerebral cortex based on functionality of the brain. Pic. by L. King. [54]

further divided into subsection based on its functionality. Following is a brief description about the functionality of each section.

The occipital lobes: respond to visual stimuli.

The temporal lobes: Involved in hearing, language processing, and memory.

The frontal lobes: Involved in personality, intelligence, and the control of voluntary muscles.

The parietal lobes: Involved in registering spatial location, attention, and motor control.

However, these are not hard cut localizations; rather they are the most common and general areas of the brain that perform these designated activities.

In this work, I am interested in capturing the human motor activities such as moving the finger, swiping, etc. Thus, it is interesting to see the region of the brain that performs motor activities in detail. Two sub regions of the brain that preforms motor activates are the somatosensory cortex and the motor cortex. The body sensation information are processed at the somatosensory cortex while the information regarding voluntary movements are processed in the motor cortex [54]. According to the research performed by W. Penfield, Figure 9.3 depicts the parts of the body that associate with the sections of the somatosensory and motor cortexes [74].

In this problem, no matter how many parts of the brain are activated, I am only interested in the designated activity that the subject performs at the end. Most of these activities are either

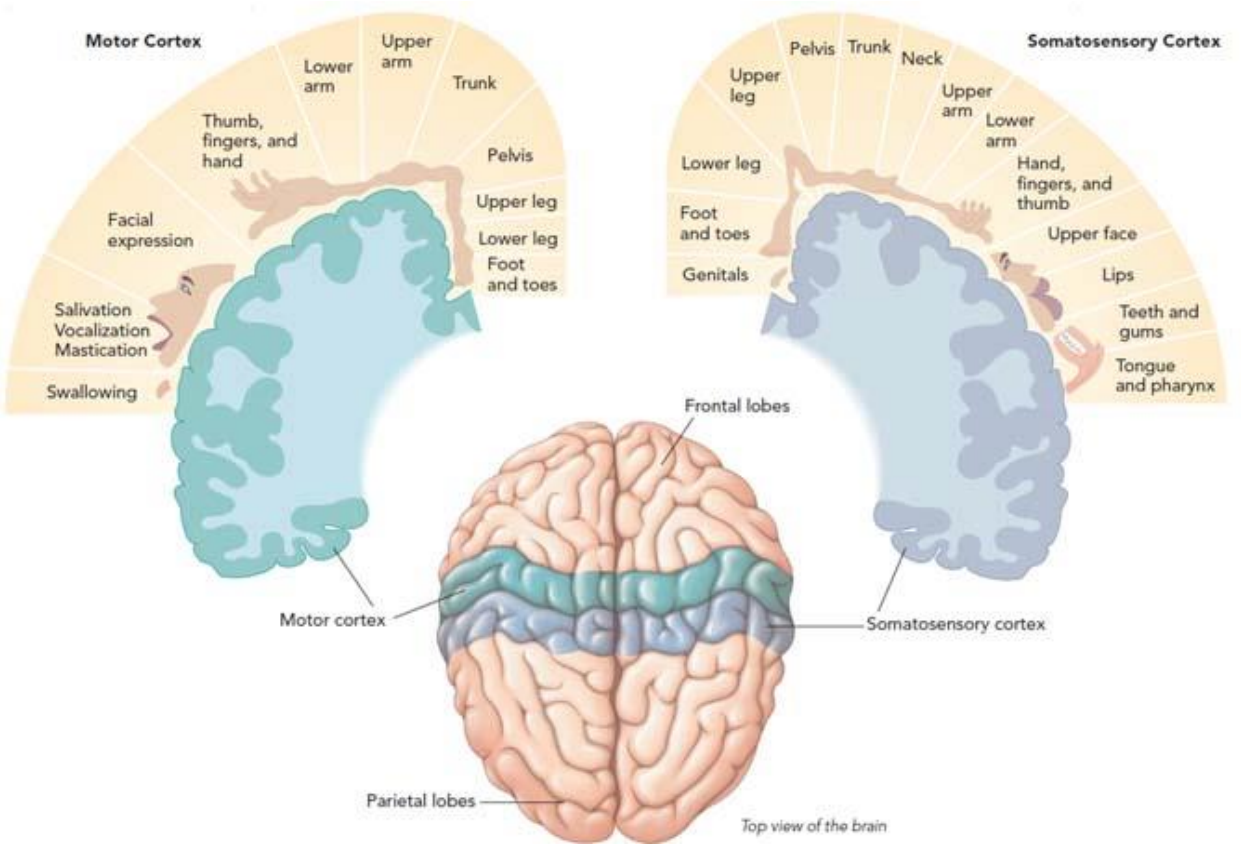
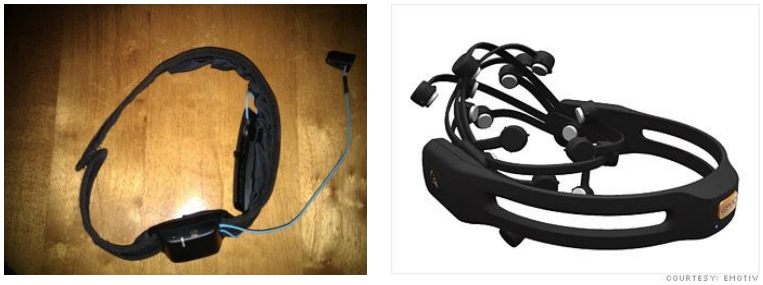


FIGURE 9.3. Parts of the body associated with the parts of the motor and somatosensory areas of the cortex. Pic. by L. King. [54]

motor activities, like moving the finger, hand etc., or sensations such as touch, vision, and audio. Therefore, EEG signals coming from these regions will determine what type of tasks the subject performed. Thus, in this research, I used EEG signals from these regions to build the task classifiers.

9.2. Electroencephalography (EEG)

Electroencephalography (EEG) is the signal that is recorded by observing scalp electrical activity which is generated by the firing of neuron brain cells in the brain [106]. EEG is also referred to as recording of the brain's spontaneous electrical activity over a short period of time, as recorded from multiple electrodes placed on the scalp [106]. EEG was originally used in neurology to diagnose and/or study epilepsy, comas, encephalopathies, brain deaths, etc. There were also other uses, such as diagnosing tumors, stroke, and other brain disorders, which have faded away due to substitute technologies.



(a) (b)

FIGURE 9.4. EEG measurement devices. (a) Neurosky Mind-Band. (b) Emotiv EPOC.

9.3. EEG Monitoring Devices

There are two brands of BCI devices which are used in experiments, namely The Neurosky and The Emotiv as shown in the figure 9.4. The Neurosky model is equipped with two electrodes, where one is used as a reference. The electrodes are dry sensors. The connection between the computer and the device was established via Bluetooth. The data collection software was same for the both devices. Mind-Band is more flexible and has more freedom in placing the electrode. To improve the quality of the contacts, an electrode gel was used, though none of these models were wet sensors. The NeuroSky not only produces EEG data, but also it calculates the power spectrum in addition to Meditation and Attention levels of the brain.

Emotiv Epoc is a 16 electrode, 16 channel, wet sensor based BCI device that is equipped with a gyroscope. It can also detect head motions as well. This device is uses Bluetooth for computer-device connection.

9.4. User-Intention Using EEG

Identifying the user-intention when a user accessing a mobile app using EEG signals is the key motive here. It is easy to identify basic events such as tapping, swiping, pinch, etc. using EEG. Once such events are identified, it can then be utilized to identify much complex user behaviors, or user-intentions. Thus, it is essential that I develop a system which is capable of identifying basic activities using EEG. I call these basic events atomic events.

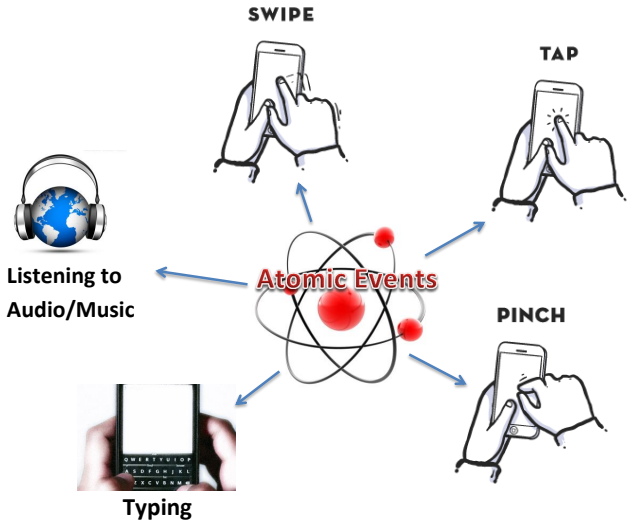


FIGURE 9.5. Atomic events considered in the work.

9.4.1. Atomic Events

An 'atomic event' is a simple and single event - performed while recording the EEG. Thus an atomic event takes place for a relatively small duration. For instance, tapping, pinching, and swiping on the mobile phone screen are considered to be atomic events. A series of atomic events can be combined serially or in parallel to form a larger, composite event. I call this a task. For example, to unlock a mobile phone screen, tap on the unlock icon and swipe it across the screen. This example involves two atomic events. Therefore, if I can identify atomic events using EEG, I may be able to identify more complex tasks by combining atomic events. I used a divide and conquer approach to tackle the concept of identifying atomic events and using this knowledge to identify more complex tasks.

To capture the atomic event using EEG, I only recorded the EEG while performing an atomic event. Since these are simple and short events, the EEG recordings are also in the form of short recordings. In this work, I try to identify a set of atomic events that are common to smart phone users. In this experiment, only following the atomic events were considered, though there are certainly other additional atomic events.

(1) Listening to music,

(2) Pinching the screen to zoom out,

(3) Tapping the screen to press a button,

(4) Swiping the screen,

(5) Typing some words on the screen.

See figure 9.5.

9.5. Methodology

For the purpose of identifying atomic events performed on a smart phone, I utilized a machine learning approach. The idea here was to train a model with known atomic events, and then test it with a set of unknown events to see whether the system can recognize the unknown events with the learned knowledge. If the system can recognize atomic events with better accuracy, it will prove that the atomic event identification using EEG is not only repeatable (can recognize the same atomic even multiple times) but also reliable (identified atomic event is the actual performed atomic event).

Developing a system to recognize atomic events can be utilized in identifying more complex user activities on a mobile phone. It can be also extended to identify the user intention on the mobile phone. Since this proposed model uses a machine learning model, it needs to be trained and tested. Thus, a good set of features are required. Since EEG signals are used in this approach, features must be extracted from these signals. Generally EEG signals are recorded in the time domain. Many EEG analyses suggest that it is better to identify EEG features in the frequency domain than in the time domain. Thus, for feature extraction, the collected EEG signals are converted to the frequency domain before features are extracted. Feature extraction is discussed in a later section in this chapter. Once the features are extracted, supervised machine learning models are trained and tested with untrained atomic events.

9.6. Experimental Setup and Data Collection

The 14-electrode EMOTIV EPOC device was used to record EEG. All the channel data were used in the training and testing. As usual, first the base line EEG data were recorded while the subject was resting both with eyes closed and while eyes were opened. Then the subject was asked to be seated in front of a table with a smart phone presented. In the standby position, the

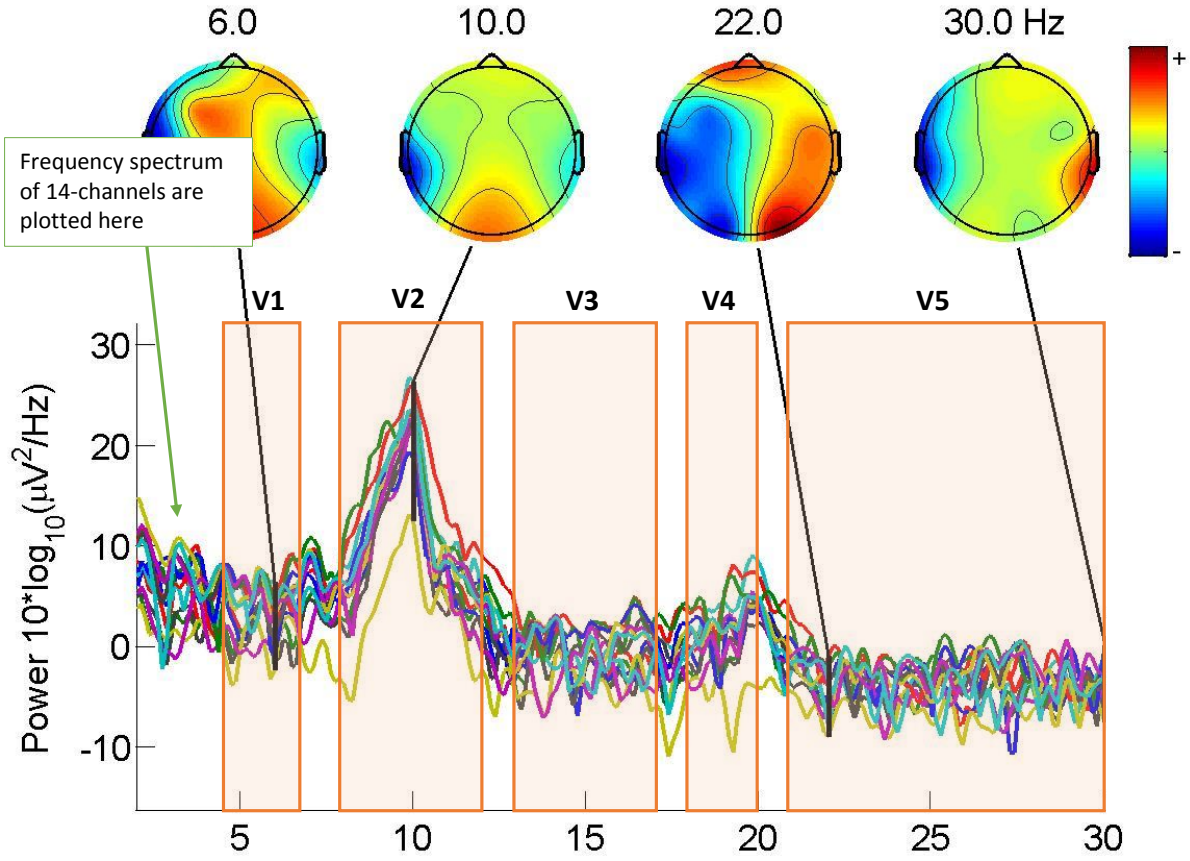
phone was placed on the middle of the table and the subject was asked to rest his hands on the either side of the phone. Once a command is given, the subject performed an atomic event on the mobile phone's screen and brought back both hands to the standby position. The EEG signal was recorded only when the subject was performing the atomic event. This way the EEG for the atomic event was captured. The subject was asked to perform each of the atomic events shown in the table 9.1 one at a time. Each atomic event EEG recordings were repeated 10 times.

9.7. Feature Extraction

Lets consider a sample EEG recording to understand the above process of forming the full feature vector. Each recording has 14-channels of EEG records. Each channel was processed identically and combined the results of each channel to form the feature vector. First, each EEG channel was converted to a frequency-time spectrum. Then the window size was increased to the whole EEG reading duration, so that it would eliminate the time axis of the frequency-time spectrum plots and only produce the frequency spectrum. Thus, in this case, the time window of the FFT is the total EEG recording duration. This is acceptable as I am performing only atomic

Atomic Event	**Number of recordings**
Baseline (EC)	3
Baseline (EO)	3
Listening to music (EO)	11
Pinching the Screen (EO)	10
Swiping the Screen (EO)	10
Tapping the screen (EO)	10
Typing "this is nsl" (EO)	10

TABLE 9.1. Description of EEG data collection. All pinching, tapping, and swiping were performed when using the right hand. For tapping and swiping, only the right index finger was used. Both hands were used while typing. [EC: Eyes Closed, EO: Eyes Open]



V1 - Average power of Theta (4-7 Hz) for each channel (14 values)
V2 - Average power of Alpha (8-12 Hz) for each channel (14 values)
V3 - Average power of Low Beta (13-17 Hz) for each channel (14 values)
V4 - Average power of Mid Beta (18-20 Hz) for each channel (14 values)
V5 - Average power of High Beta (21-30 Hz) for each channel (14 values)

Full Feature vector, FV = [V1, V2, V3, V4, V5]

n(FV) = 14 * 5 = 70

FIGURE 9.6. Extracting features from frequency spectrum to form feature vector.

event in each reading. This process is illustrated in Figure 9.7.

Once they were converted into frequency domain, each channel frequency was divided into 5 segments based on the standard EEG frequency ranges; Theta (4-7Hz), Alpha (8-12Hz), Low Beta (13-17Hz), Mid Beta (18-20Hz), and High Beta (21-30Hz). See Figure 9.6 for frequency ranges of the standard EEG bands. The power values in each window were averaged to produce

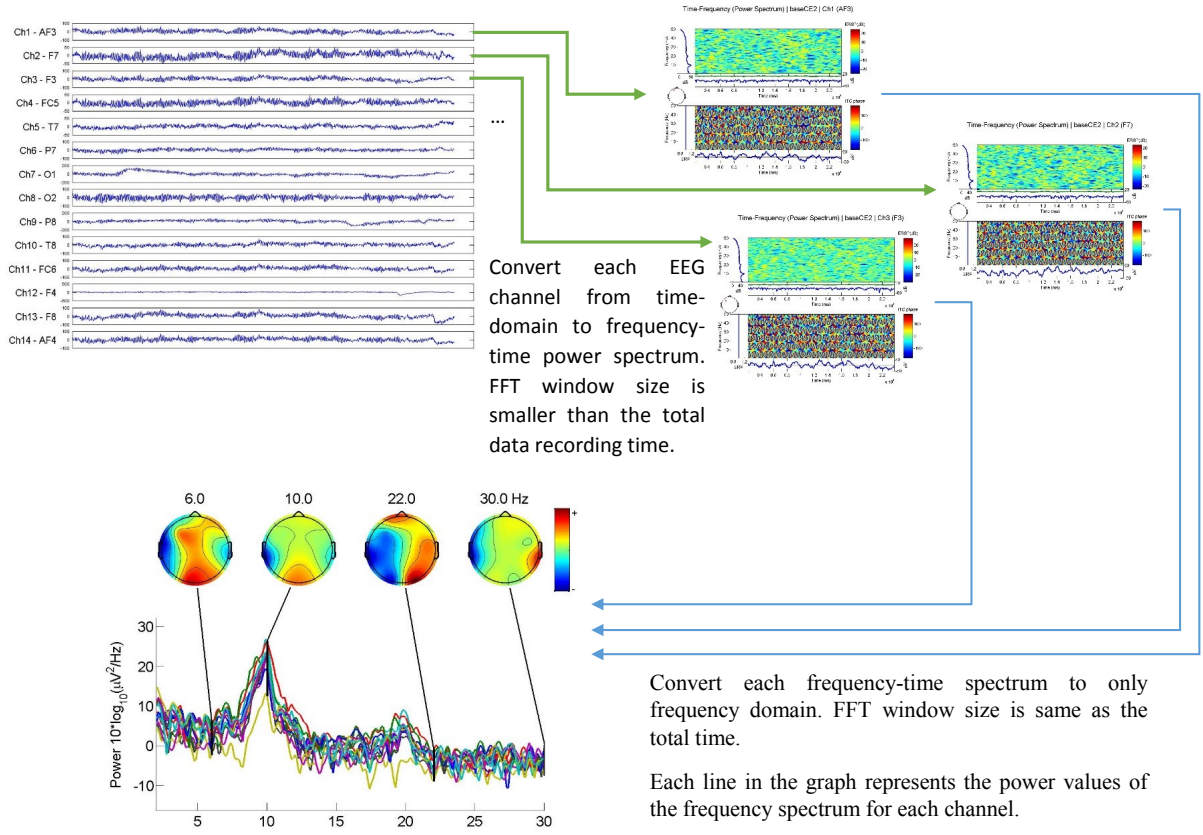


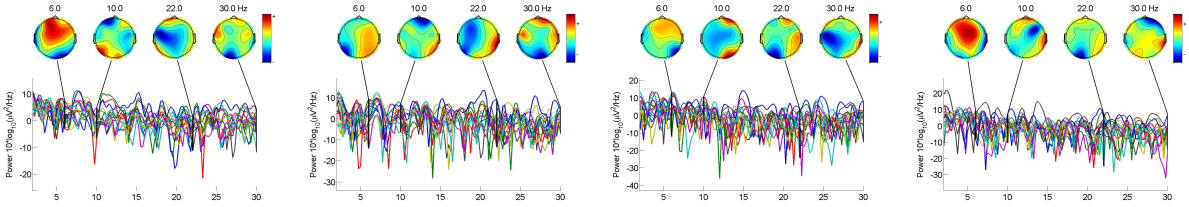
FIGURE 9.7. Converting EEG signals of all 14-channel to the frequency domain.

the feature value. Since there were 14 channels, 14 feature values were produced for one frequency band. Thus, for each 5 bands, it produced $14 \times 5 = 70$ values in the final feature vector for each reading. This process is illustrated in Figure 9.6. Once the feature vectors were extracted from each EEG reading, they were organized into an ARFF file format for the Weka [45] tool to process in its various machine learning models.

9.8. Readings

This section depicts the power spectrum of the EEG signals obtained while performing atomic activities on the phone. A single EEG channel can produce a power spectrum. Since the EEG device produced 14-channels of EEG data per reading, 14 power spectrums were produced per reading. In other words, for each recording of 14-channel EEG data, 14-power spectrums were generated. To simplify the illustration, only the first activity type (atomic event) is presented with two types of readings sets: 1. the power spectrums for each channel with the time axis (FFT

is applied for small windows of time) and 2. the power spectrum for the whole window (time is eliminated by applying FFT for the whole time window). Since the latter is produced by the previous plots, for the rest of the atomic events only the second set of power spectrums(power spectrum generated by applying FFT for the whole time window) were depicted.

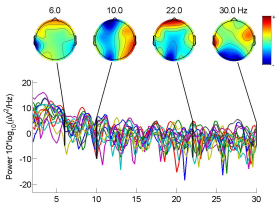


(a) Listening to Audio/Music

(b) Pinching the screen with the right hand index finger and the thumb

(c) Swiping the screen with the right index finger

(d) Tapping the screen with the right index finger



(e) Typing on the soft keyboard with both thumbs

FIGURE 9.8. Power base of all 14 channels per event.

This collection generated a large amount of data. For illustration purposes, I present one set of readings from each activity type. Each sub-figure in Figure 9.8 illustrates the frequency spectrum of all the 14-channels per given event.

Listening to Audio/Music: See Figure 9.9 for power spectrums with time and Figure 9.10 for power spectrum for the whole reading. This is to indicate that that figure 9.10 is generated from figure 9.9 by cumulating through the time axis. The time varying frequency spectrum is not required in this scenario as all these plots are referring to atomic events. It is assumed that, during an atomic event there were no significant variation in the power spectrum. Further, to depict the rest of the events, only power spectrums generated form the whole readings are considered.

Pinching: See Figure 9.11 for power spectrum for the whole reading.

Swiping: See Figure 9.12 for power spectrum for the whole reading.

Tapping: See Figure 9.13 for power spectrum for the whole reading.

Typing: See Figure 9.14 for power spectrum for the whole reading.

9.9. Preliminary Results and Observations

Since this is a classical classification problem, I present the results in a confusion matrix. The accuracies and F-measures were calculated to determine the performance of the system. Since, in the previous experiments, Multilayer Perceptron (MP) depicted the better performance, in this model only this classifier is used. The data set was split into two sections, one being the train set and the other being the test set. The train set consisted of 50% of the data while the test set consisted of the other 50%. I trained each machine learning model with the train set and tested with the test set. The results are shown in Tables 9.2 and 9.3. The WEKA performance summary is shown below.

```
=== Evaluation on test set ===
=== Summary ===
Correctly Classified Instances          24               85.7143 %
Incorrectly Classified Instances         4               14.2857 %
Kappa statistic                          0.8255
Mean absolute error                      0.0713
Root mean squared error                  0.184
Relative absolute error                 29.7309 %
Root relative squared error             53.3957 %
Total Number of Instances               28
```

9.10. Summary and Future Work

Even with a totally new data set collected at a different points of time and with both more data and a new set of events, the model performed to produce an overall accuracy of 85.7%. This accuracy further can be improved by filtering EEG data by removing artifacts. This concludes that atomic event classification is feasible and highly accurate.

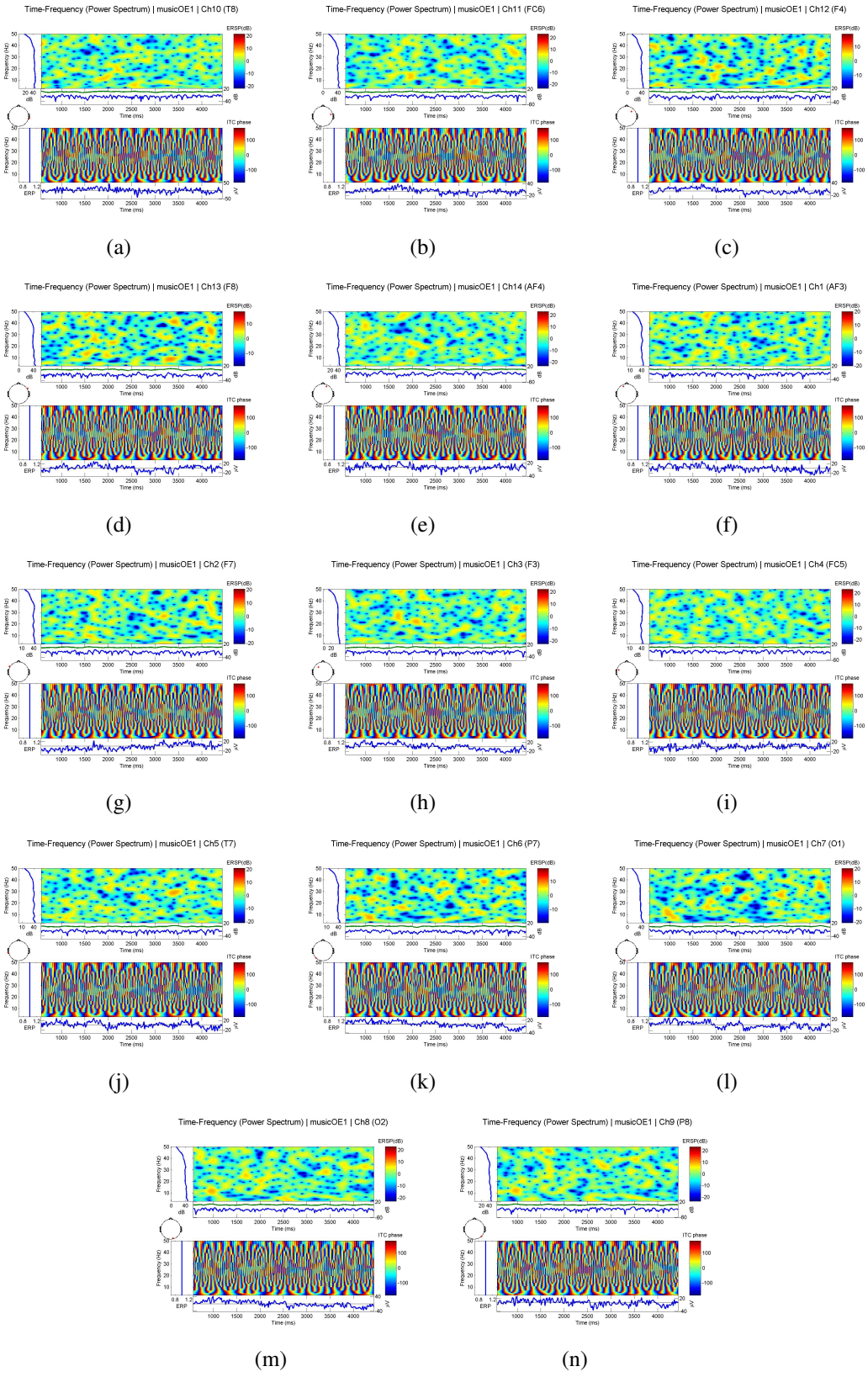


FIGURE 9.9. Listening to Audio/Music - frequency spectrums of all 14 channels.

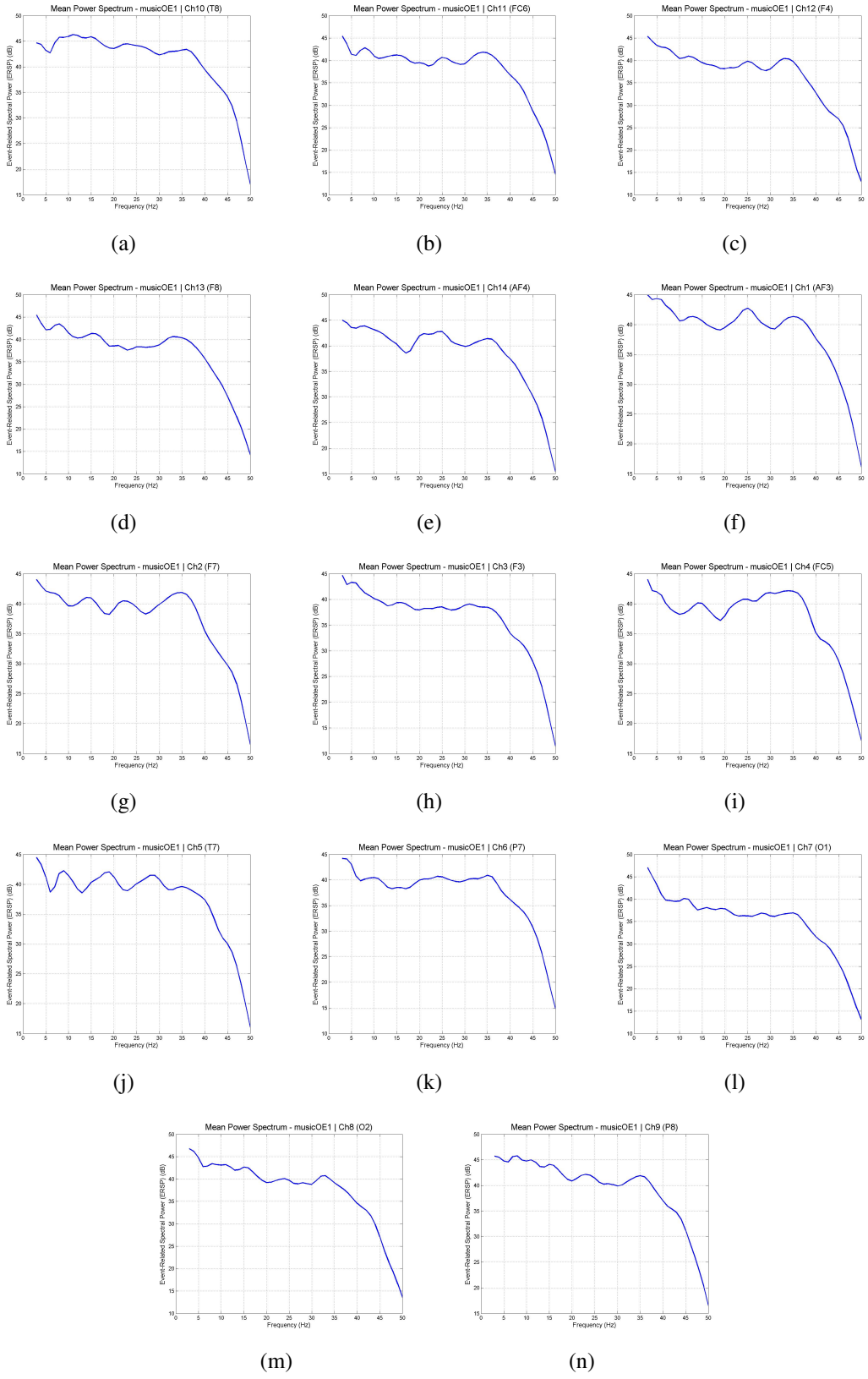


FIGURE 9.10. Listening to Audio/Music - Power base of all 14 channels.

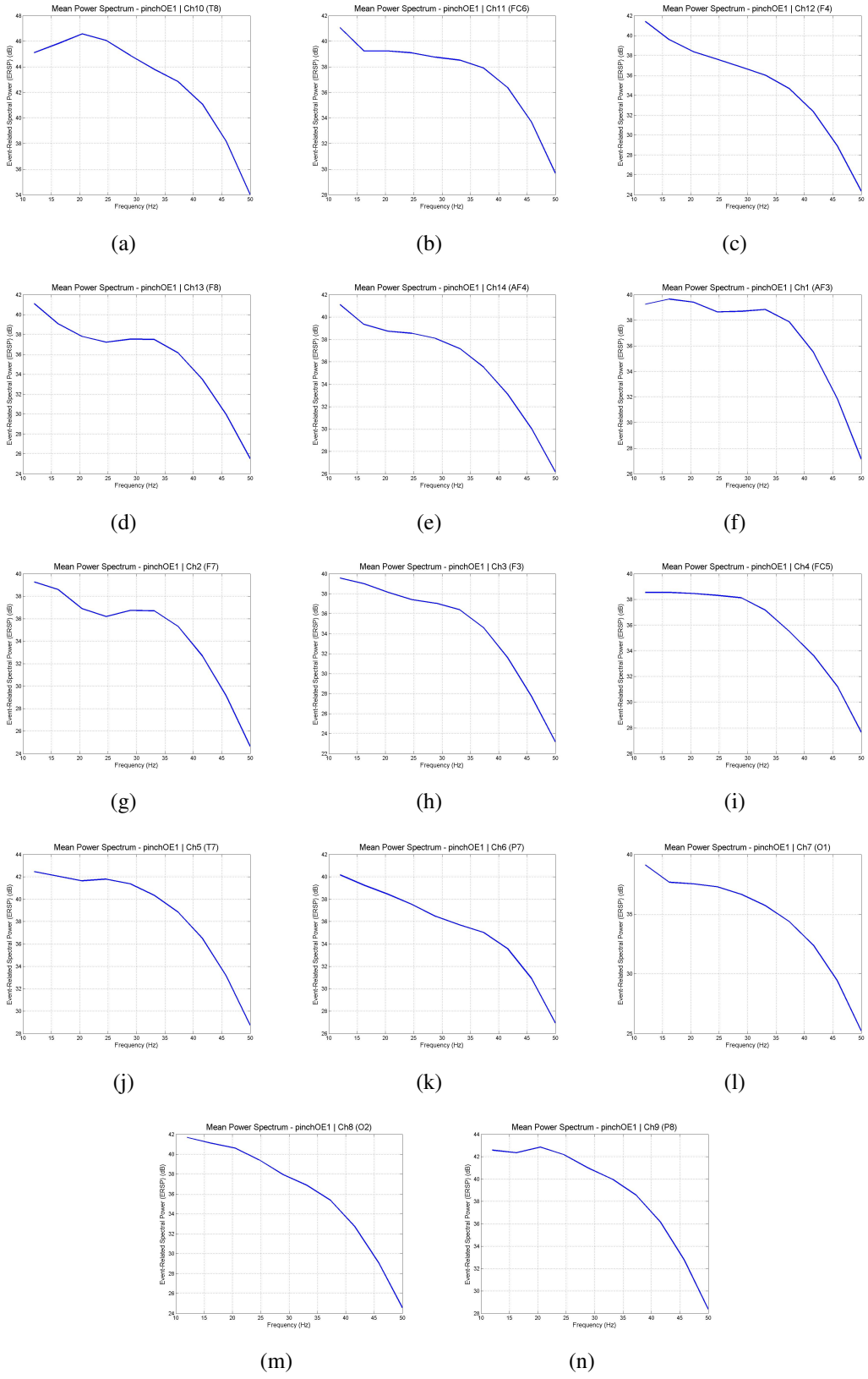


FIGURE 9.11. Pinching the screen with the right hand index finger and the thumb - Power base of all 14 channels.

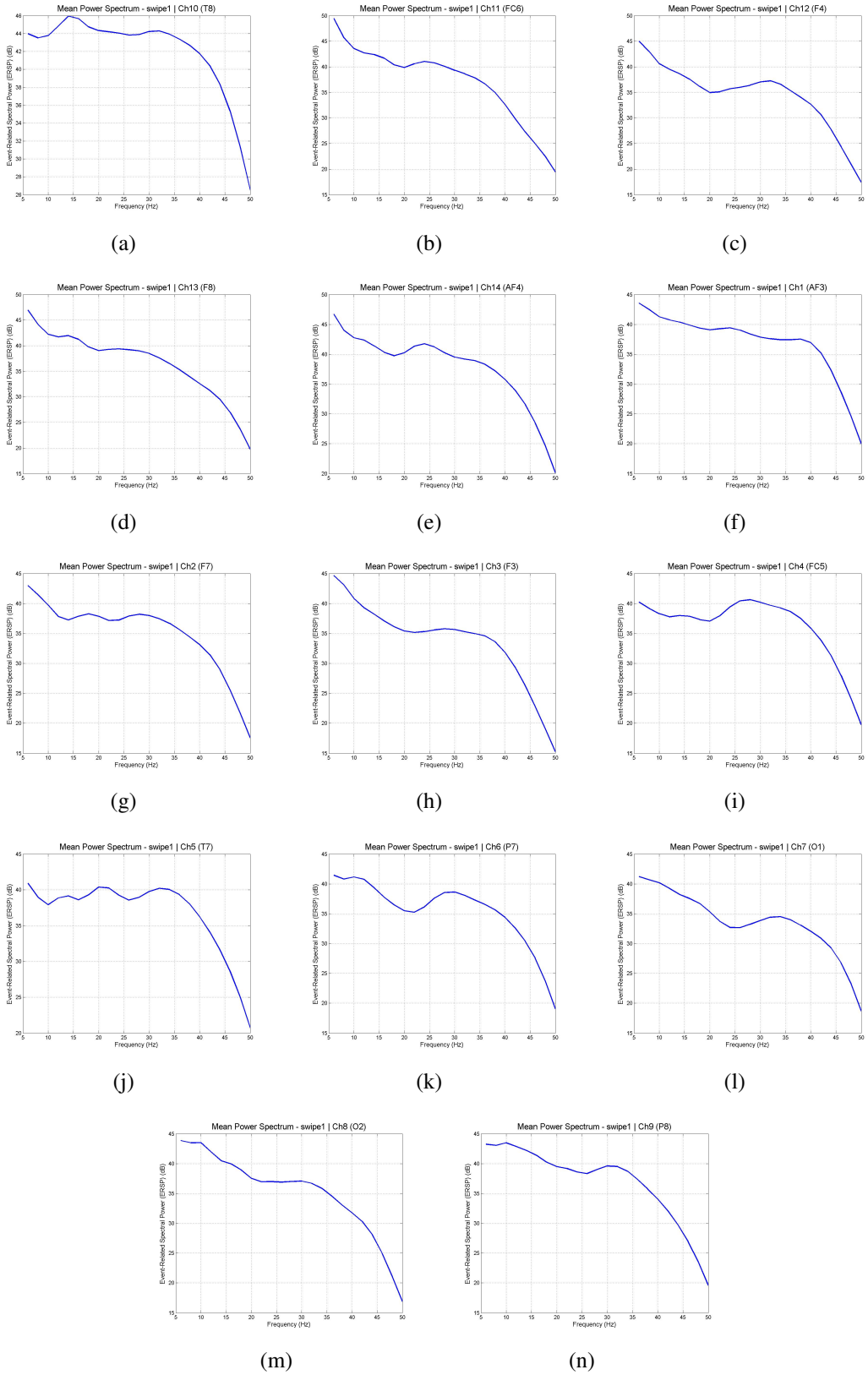


FIGURE 9.12. Swiping the screen with the right index finger - Power base of all 14 channels.

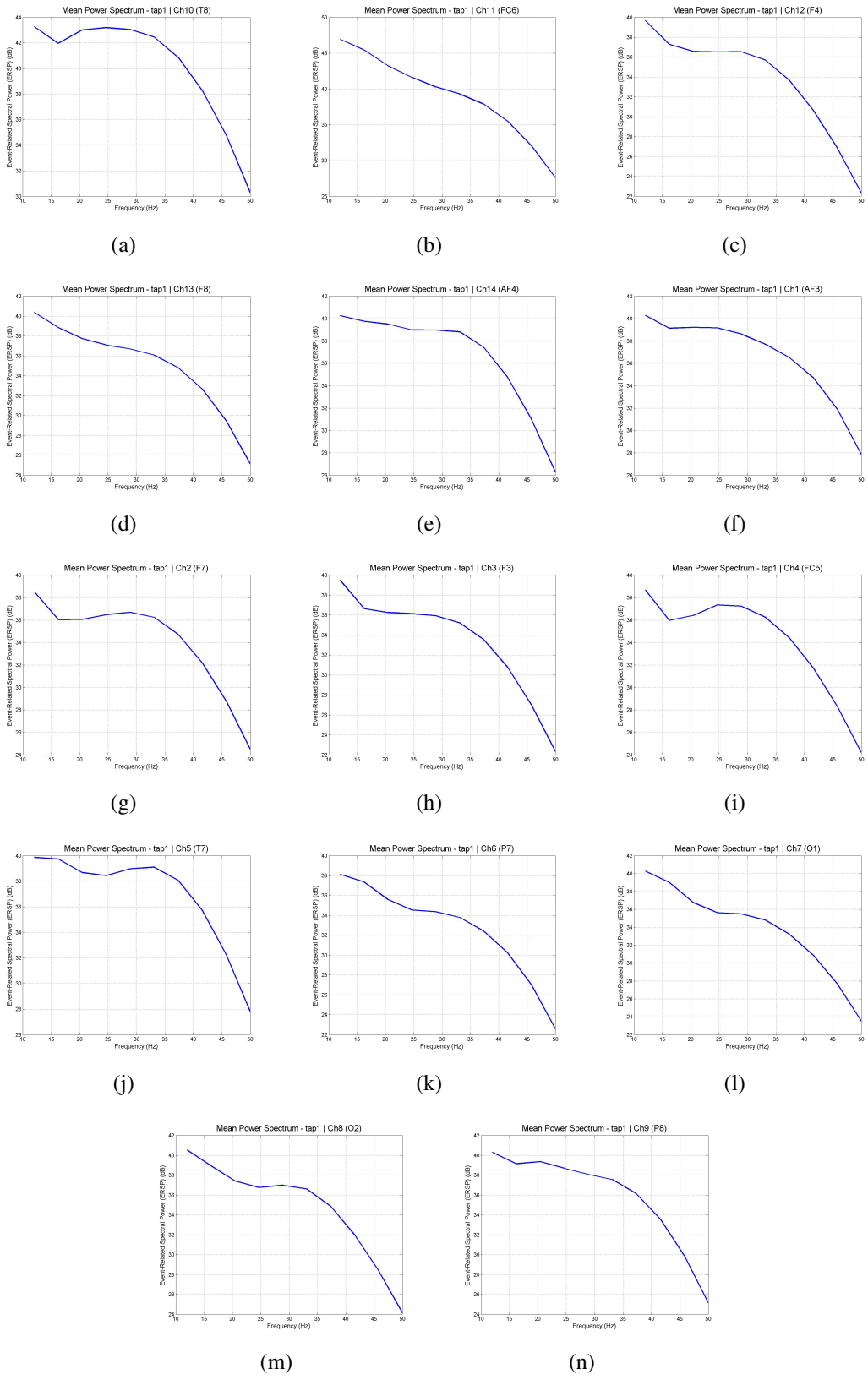


FIGURE 9.13. Tapping the screen with the right index finger - Power base of all 14 channels.

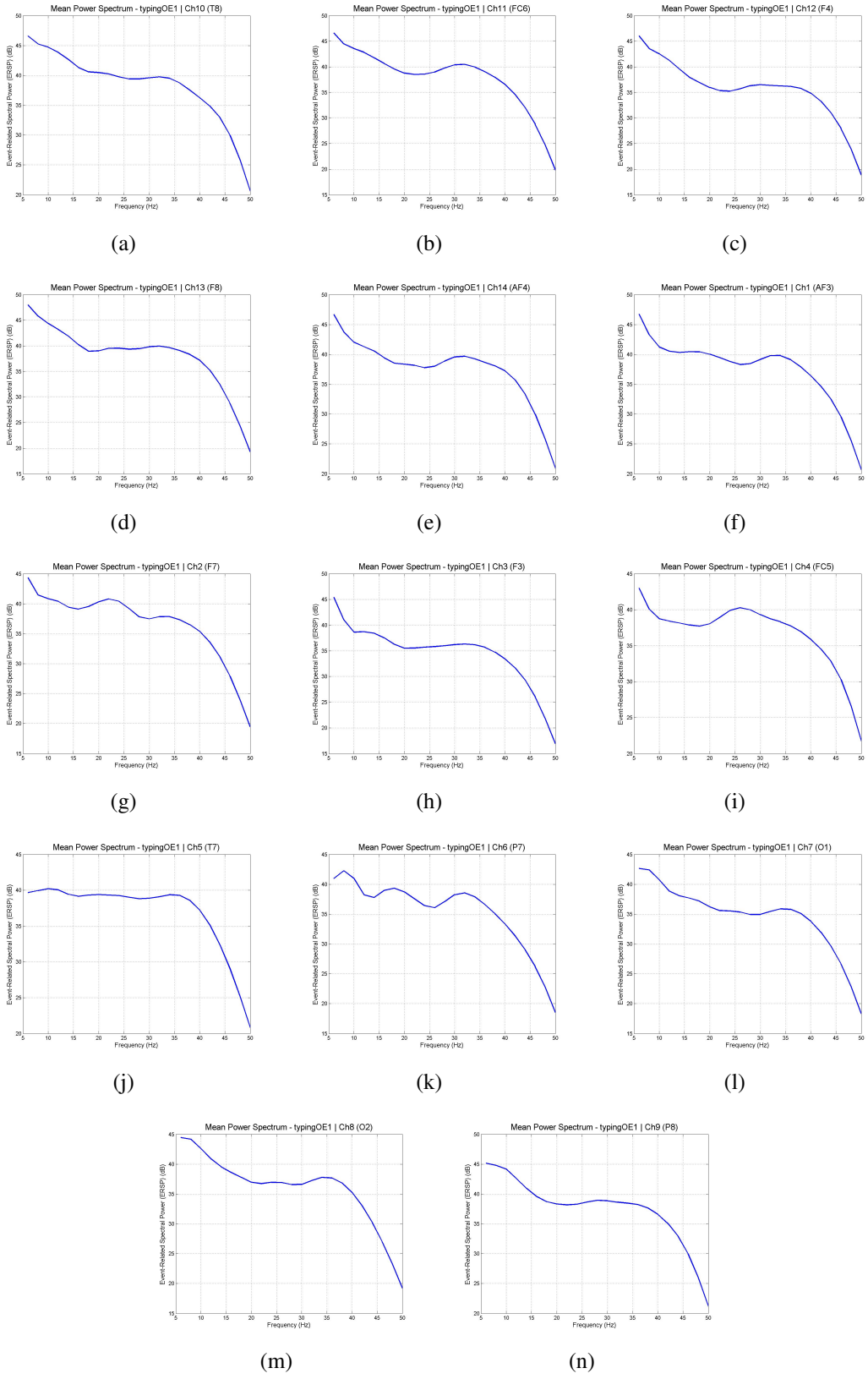


FIGURE 9.14. Typing on the soft keyboard with both thumbs - Power base of all 14 channels.

Confusion Matrix

Classified As							
a	b	c	d	e	f	g	
1	0	0	0	0	0	0	a=baseCE
0	0	0	0	1	0	0	b=baseOE
0	0	6	0	0	0	0	c=musicOE
0	0	0	4	1	0	0	d=pinchOE
0	0	0	0	4	1	0	e=swipe
0	0	0	1	0	4	0	f=tap
0	0	0	0	0	0	5	g=typingOE

TABLE 9.2. Confusion matrix for multilayer perception.

TP Rate	FP Rate	Precision	Recall	Recall	F-Measure	ROC Area
0.857	0.031	0.833	0.857	0.857	0.844	0.962

TABLE 9.3. Performance of the multilayer perception. TP:True Positive, FP:False Positive, ROC:Receiver Operating Characteristic.

In the future, these atomic events will be combined to perform more complex tasks. Also, when EEG signals are recorded as a sequence, a windowing technique is proposed to identify a sequence of atomic events. Then, based on the sequence of atomic events performed, the overall activity could be determined. Eventually this could lead to identify a user based on the unique atomic event sequence signature. In the future I hope to open up a research area with a concept called ‘atomic event detection using EEG’ to identify user’s intention.

Also, in the future this model will be studied to perform the same classification with a smaller number of electrodes. This would localize the most common atomic events to a certain part of the brain. This can be performed by using statistical feature selection algorithms in order to identify the most potential area of the scalp and then decide which electrodes need to be eliminated or vise versa. The idea here is to simplify the EEG collection by reducing the number of electrodes,

while also identifying the most ideal location of the scalp to place the limited number of electrodes.

This work can be extended to identify different users by producing unique EEG signal for different atomic events. This can be used to continuously authenticate the users in a mobile phone even after the initial authentication. This way, even if the mobile device is acquired by another user (due to stealing, coercion, etc.) and broke the password, still the attacker cannot use the mobile device as the owner's EEG is required to use the device.

CHAPTER 10

DISCUSSION AND CONCLUSION

This section includes the overall interpretation of my dissertation work and how it addresses the thesis. I discuss my proposal to improve the security of mobile phones. This includes:

- Dynamically recognize malware and trolls through modeling behavior patterns.
- Identify sensitive words to optimize encryption in a voice conversation.
- Create a dynamic, powerful, unable to be falsified or mimicked authentication system that uses unique information that is constantly provided by the user.

10.1. Limitations of Current Models and How They Can be Addressed in Future Work

10.1.1. Malware detection limitations

There are several limitations when attempting to address malware detection. The first is that there are malware which are designed to evade being detected in their permission requests, including repackaged applications which are challenging to detect. Additionally, once malware is detected... what is the next step? It is difficult to quarantine something that is designed to resist being quarantined. And finally, we could improve the benign application grouping by extracting additional features.

Lessons Learned and Limitations

With this work, I came to an understanding that the intention of an application plays a vital role in determining its behavior in a system. I introduced a novel approach to identify potential Android malware applications by identifying its intention and observing its permission requests. This system utilizes several machine learning models, which indicates that it can be trained to perform better. I also identified several limitations in this approach, such as consequences of mislabeling the classes, dependency on the permission list, and issues in reverse engineering the Android app due to Java Native Interface calls. I also provided solutions for problems like class mislabeling by introducing unsupervised machine learning models. However, other problems are left to be addressed in future work. Further, one of the feature extraction methods adopted a string

extraction approach, which could also pertain to certain limitations. When the user language of the app is not English, these dictionaries could fail to extract the correct features and this could leads to poor performance. Nevertheless, this could be resolved by constructing dictionaries with multiple languages and this model is capable of such scalability.

Zero-day Malware Detection

Zero-day malware apps are emerging threats previously unknown to the malware detector system [24]. This approach does not follow any payload signature based malware detection. Therefore this system does not need to be trained with malware payload signatures. I only train this system with the characteristic features of the benign apps. I detect malware apps when they try to obtain unrelated permissions from its task-intention I-Shape. Therefore, the model can be used to easily identify any Zero-day malware samples that do not have the right intention. Further, because of I did not use any malware samples to train the system, all the detected malware samples are Zero-day malware detections.

10.1.2. Social network limitations

One major limitation inherent to statistical base solutions is the completeness of the dataset. It is important but hard to get a hold of a complete and comprehensive dataset. To obtain a reasonable sample size, more crawling is needed to collect information about the network. Further, social graphs are dynamic and relationships change with every single human interaction. The approach needs to be improved to have a model which constantly evaluates the social graph, because spammers/lurkers/trolls evolve over time.

10.1.3. Phone Encryption limitations

Phone hardware is limited and could benefit from improving battery power, CPU, and even adding dedicated encryption chips. Further, speech engine / voice recognition engine used in the model is not very accurate. It needs to be trained further for more robust recognition. Further, it is important to have a stand-alone speech recognition system that does not depend on any external resources like Google speech engine. Sending sensitive words to a cloud to recognize will defy the main purpose. Because if they are not encrypted it can be eavesdropped by a man in the middle.

Encrypting each word before sending to a cloud for sensitive word recognition again will not make any sense in this application as the main goal of recognizing sensitive words is to perform the encryption.

10.1.4. Limitations of using EEG with a phone

There are a number of limitations with EEG. The first is that EEG is not perfect yet needs to be improved on handheld devices. Another limitation is that EEG has to be worn and the current design is uncomfortable and inconvenient. Despite this, EEG is extremely useful and is probably the direction of the future if it can be simplified.

Another limitation is that my study only identified different types of activities of a single user using EEG, so more studies need to be done in order to use this as an authentication model. Again, my work here was a preliminary study, so further experiments should be conducted with a larger dataset and multiple users. This behavior can then be used to create signatures for users. These signatures could then be used to authenticate user logins on their phones.

10.2. Conclusion

Importance of identifying the intention of an agent to provide security for mobile system was studied in this dissertation. Intentions can be of different forms. To begin with, different types of intentions were studied. Two identified major different intentions were the user-intentions and the application-intentions (app-intentions). Application-intentions were further subdivided into four types; task-intentions, alternate-intentions, malicious-intentions, and benign-intentions. User-intention was useful in identifying malicious behavior of a user in a system while app-intention was useful identifying malicious behavior of an application.

Later in the document, four useful security applications were discussed. First, a malware identification system in the Android system was studied. It was one of the direct applications of app-intention identification. At the inception of the approach, task-intention of an Android app was identified using machine learning models. Later, permission requests of such Android apps were extracted and I-shapes were constructed for each different types of task-intention groups. I-shapes represent the probability distribution of the permission requests of given Android app

task-intention group. Then, by using the task-intention, permission requests and the I-shape of an app, its maliciousness or the benign-ness was determined. This approach produced an accuracy of 89% in identifying malware samples in the database. In terms of specificity, this approach performed better than traditional anti-malware tools such as AVG and Norton.

Under applications of user-intention identification, three examples were discussed; user role identification in a social media, context-aware encryption in voice communication, and EEG based behavior identification and user authentication.

A Twitter social network dataset was used in user role identification. In this problem, two approaches were demonstrated; context-dependent and context-independent approaches. In both approaches, different machine learning models were utilized. These user role identification demonstrated how to use user-intention to identify malicious users such as spammers, spam-bots, lurker etc.

In the context-aware encryption application, another usage of user-intention identification was explored. User-intention in a telephone conversation is used to improve the performance of existing encryption protocols. The idea was to monitor the phone conversation with speech recognition system in real time to identify any sensitive information. Once such information is detected, user-intention is identified to be 'transmit sensitive information'. Thus, the encryption mechanism encrypts the data with higher strength. Otherwise it is not encrypted. A sample model was implemented to prove the practicality of this model and feasibility of the model was studied.

Finally, user-intention was explored further with EEG brain signals of a phone user. The idea was to introduce a novel platform to identify user intention through EEG by identifying user's fine grained behavior known as atomic events. Many potential mobile security solutions could be pointed out with this approach such as continuous user authentication, cryptographic key generation, user misusing mobile apps. Though none of the potential applications were implemented under this section, it was left open for the future references to explore further on this.

The intention identification and improving the mobile security by utilizing those identified intentions are the main contribution of this dissertation.

10.3. Summary

Since mobile devices are becoming very popular, security is a very serious issue to be addressed. Attacks are evolving from all directions. Therefore it is an important task for future security researchers to keep up with these trends.

APPENDIX A

READING EEG DATA IN A MOBILE PHONE

In this chapter, I discuss on how to integrate the Emotiv EEG device to an Android device and communicate with the headset to obtain raw data, visualize it, save it, and process it. I also discuss about challenges faced by other researchers and how they overcame these challenges.

(a) Emotiv USB wireless data Receiver (USB Dongle)

(b) Emotiv EEG recording, wet sensor electrode headset

FIGURE A.1. Emotiv EEG recording device.

A.1. Introduction

The Emotiv EEG head sets are available with different packages. Though the hardware is the same on all the packages, the software bundle varies based on its functionality. The main difference in device software is the capability of obtaining the raw data from the device and recording it on a file. The Research and Enterprise editions are capable of handling the raw EEG for a higher price tag. Since the device is same for all the editions, the same raw data is encrypted when it is transmitted from the USB receiver to the PC. The previously mentioned exclusive software only poses the key to decrypt the raw data. However, the manufacturer has not officially published any technical information about the decryption or about the decryption key. The USB receiver connects to the headset automatically when the headset is in close proximity. The Emotiv USB receiver is shown in Figure A.1(a). Communication between this USB receiver and the head set is

established on a wireless medium according to the 802.11 frequency range (2.4Ghz). Therefore, this device does NOT use Bluetooth. The manufacturer's comment on this decision is "*EPOC uses a low-power chipset which is not Bluetooth - BT is horribly power hungry and we avoided it to make sure our battery lasts a while*". Also, this USB dongle is known to encrypt the EEG raw data before any data is sent to the PC.

A.1.1. Problem Definition

I wanted to create a software driver for the Android mobile device in order to communicate with the Emotiv USB communication device, called the Emotiv USB dongle. The Dongle will automatically connect to the headset and communicate with it to receive the raw data. Then I wanted to process the data in realtime, by using the developed software to obtain the raw data, and finally visualize the data and store it in the local SD card for later retrieval.

A.1.2. Methodology and Challenges

The following activities can be identified as the outline of solving this problem;

- **Create a communication bridge between the USB dongle and the Android device via USB connection:** Since the manufacture provided very less details about the Emotiv USB dongle, finding its USB interfaces and its EEG sensor details were difficult. Further, figuring out some critical USB dongle parameters like usage of each USB interface, type of the interface, direction of the end points, protocol of the data frame consumed reasonable time due to lack of documentation. Initially, this needs to be addressed to establish the communication.
- **Decrypting the raw data packets:** As mentioned earlier, the raw data frames are encrypted with AES-128 due to its commercial nature. With the aid of external sources, the USB dongle software was exploited to determine its encryption key. Thus, it was used to decrypt the packets. This is the most challenging part as the manufacturer generated the encryption key based on the serial number of the Emotiv USB dongle and it varies from device to device. Also, all the sources who exploited the device have cracked mainly on PC platforms with different languages like C or Python, rather than Java. Thus, to obtain

the cryptographic key, the dongle must be connected to one of those systems. Setting up such a platform is tedious and time consuming, especially due to extensive dependencies.

- **Constructing the data packet from the data frame:** Once the raw data frames are decrypted, the data will be in raw binary form. Constructing a data packet that represents meaningful information such as EEG node sensor values and built in gyroscope values is challenging, again due to the lack of documentation about data packets. Furthermore, although external resources have done this in other language models such as C or Python, porting them to Java/Andoid is challenging.

A.1.3. Literature

Two interesting projects can be pointed out on the use of the Emotiv headset with a mobile smart phone. However, none of these publications explain a direct connection to the Android platform.

Campbell et al., a research team from University of Dartmouth, published a paper on how to connect the Emotiv EEG device to a smart phone. They used the EEG signals to dial a call of a person in the contact list [15]. They then used this device to obtain the P300 [58] signal to classify the people in the contact list. The mobile device they used to process the data was an Apple iPhone. It is interesting to note the way they connected the Emotiv EEG headband to the mobile phone. As they mentioned in the paper, they acknowledged that Emotiv is a closed-source SDK and the raw data is encrypted. Thus, they used a laptop to obtain the raw data from the headset that runs the proprietary Emotiv SDK and relay that information to the mobile phone via WiFi. This means that they never connected the Emotiv headset directly to the phone in order to obtain raw data, as was suggested in this work. Furthermore, their technique had several disadvantages in the communication, such as high power consumption for Wifi communication and unreliability of external hardware (like a PC or a laptop).

Another research team from the Technical University of Denmark, Stopczynski et al. built a smart phone interface for the Emotiv EEG device. This group used a Nokia N900 phone which runs the Maemo 5 OS (an OS developed by Nokia for its phones) to interface the EEG device [95]. Their system directly connected to the Nokia device and decrypted the EEG raw data in the phone

in order to visualize it on the mobile screen. The software was implemented using Python, and the setup was used to ”demonstrate the ability to distinguish among emotional responses reflected in different scalp potentials when viewing pleasant and unpleasant pictures compared to neutral content.” [77]

According to a Blog post by G. Moro, a toolkit named ’Emokit’ from C and Python was implemented to communicate and obtain the raw EEG from the Emotiv device in a PC [66]. This PC software is independent from the Emotiv workbench toolkit. The ’Emokit’ is an open source toolkit which is freely available on GitHub under the following link `https://github.com/qdot/emokit`. This code and documentation was very useful in obtaining the encryption key of the USB dongle.

An interesting concept emerged in the market related to the brain mobile interfacing called neurocam form Japan. The idea was to use brain signals to determine your interests towards what you are looking at. A head mounted iPhone will trigger a video recording automatically if you show more than a certain level of interest [70, 69].

It is interesting to note that a similar approach is followed in this work, except all the implementations are on the Android platform. This has never attempted by any other group to this date. In fact, the Emotiv EEG device manufacturer also announced that their device does not support Android platform yet and they are in the process of implementing such APIs for the Android platform. Thus, I implemented the EEG to mobile phone integration due to the urgent need in my research.

A.2. Implementation

Once the USB dongle is plugged into the phone and the head set is switched on and brought into close proximity, the dongle will establish the Dongle-to-Headset connection automatically. I assume that this connection is established by default before implementing the rest of the software connection. The software of connecting the Emotiv USB dongle to the Android device was implemented in three stages. See Figure A.2. In the first stage, the Android app established the communication between Emotiv USB dongle through Android APIs. Once connected, the app was capable of accessing the raw data. Since the dongle encrypts these raw data, it needs to be

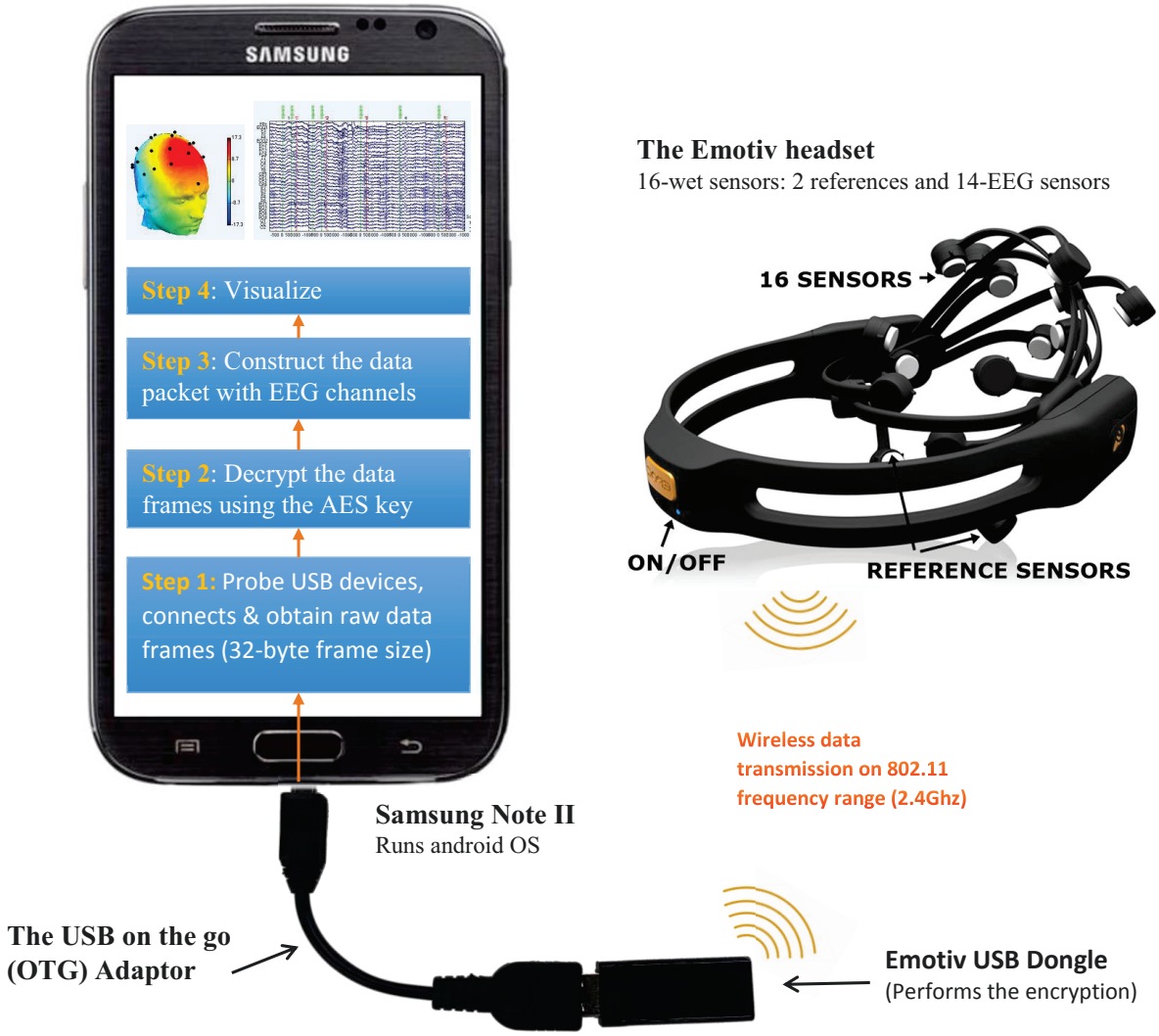


FIGURE A.2. Architecture of the interface.

decrypted before processing. Hence, at stage two, these raw data were decrypted. Then it was converted to proper EEG data packets in stage three. This produced the EEG data for each channel and other information. Once stage three is done, the raw EEG data was available for visualizing and processing. More details about this process are discussed in the following sections.

A.2.1. Android App to Emotiv USB dongle connection and obtain 32byte data packets

This was the first step in this data retrieval. Initially, a broadcast receiver was initiated to auto detect the USB device and obtained the user permission to connect to the USB dongle. Then, the app created a USB manager to probe all available USB devices connected to the mobile phone. Once it identified the plugging in of the Emotiv USB dongle, the app prompted the user (first time

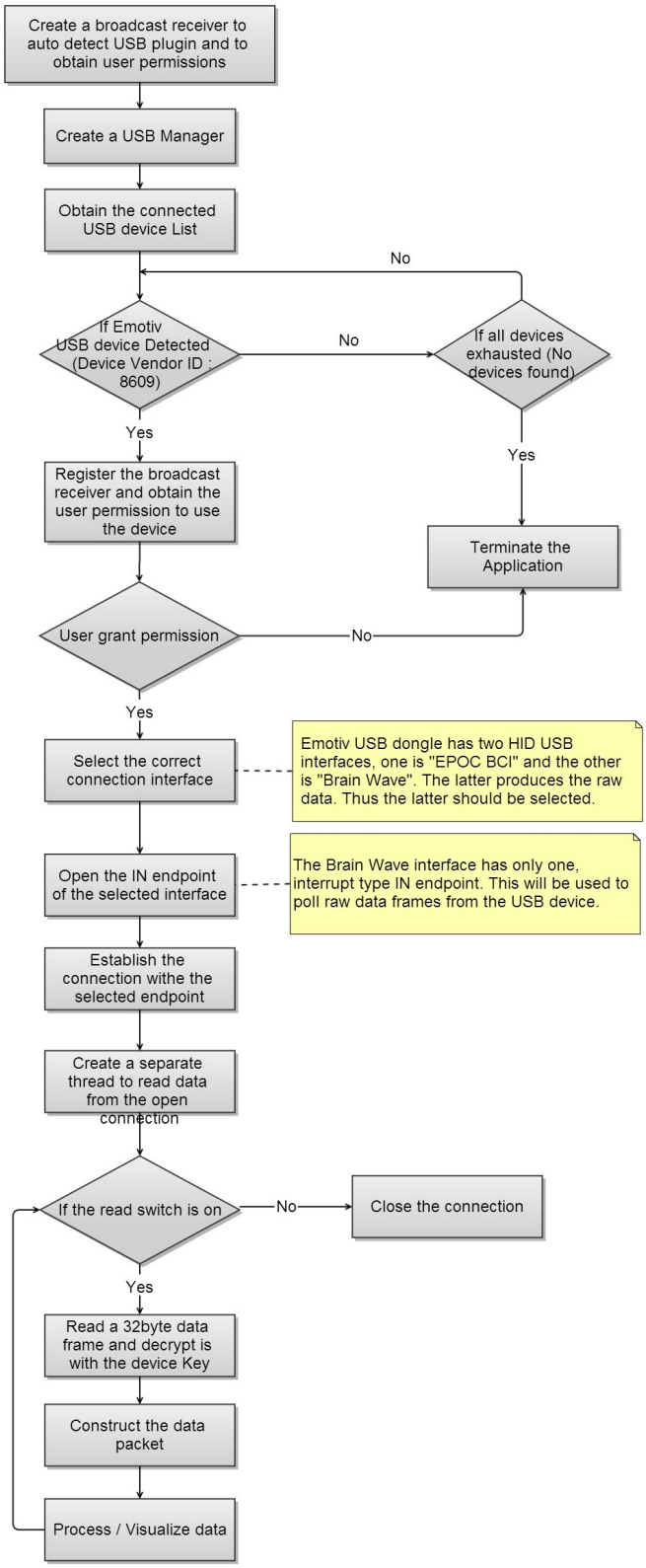


FIGURE A.3. Flow Chart of the data reading Android app.

only) to authorize this app to use the USB device. When its authorized by the user, it established the communication and started reading data frames from the dongle. This process is illustrated in the flowchart depicted by Figure A.3.

However, to establish the above explained connection, it is necessary to identify the parameters of the USB device which was plugged in. To obtain the parameters, one can use any USB probing tool to collect the dongle's information. Figure A.4 depicts the parameters of the Emotiv USB dongle. As mentioned earlier, this device has two interfaces and each interface consisted of one Human Interface Device (HID) interrupt type as an IN endpoint. Thus, data can be only received through this port and cannot send any data to the dongle or to the headset through this port. The vendor id of the device is 8609.

Once the data connection is established it can send 32-byte data frames at an average rate of 128Hz. The structure of this frame and how to construct the EEG and other data embedded in this frame are discussed in a later section.

A.2.2. Decrypting the packets

According to the Emokit tool documentation, the encryption is AES-128 bit with 128bit key [26]. Also according to this, the data emitted from the Emotiv headset is not encrypted and it is encrypted at the USB dongle. In early batches of the USB dongles encrypted the data with a common key. Since this key was breached, the manufacturer produced later batches with unique encryption keys based on the serial number of the USB dongle. Thus Emokit documentation describes how to generate this key from the dongle.

Emokit toolkit was developed in C and Python languages to extract the key from the USB dongle. The toolkit was capable of receiving raw data from the Emotiv device. I used the Python version of this toolkit to extract the cryptographic key of the dongle. After the key was extracted, it was used with the Android Crypto package to decrypt the packets. Due to the proprietary nature of the cryptographic key, it is not included in this document. One can refer to the Emokit toolkit documentation for further details.

```
Device Info Device Path: /dev/bus/usb/002/042
Device Class: Use class information in the
              Interface Descriptors (0x0)
Vendor ID: 21a1 Vendor
Name: Emotiv Systems Pty. Ltd.
Product ID: 07fa Product
Name: not in db Interfaces

Interface #0
    Class: Human Interaction Device (0x3)
    Endpoint: #0
        Address : 129 (10000001)
        Number : 1
        Direction :
        Inbound (0x80)
        Type :
        Intrrupt (0x3)
        Poll Interval : 1
        Max Packet Size: 8
        Attributes : 000000011

Interface #1
    Class: Human Interaction Device (0x3)
    Endpoint: #0
        Address : 130 (10000010)
        Number : 2
        Direction : Inbound (0x80)
        Type : Intrrupt (0x3)
        Poll Interval : 1
        Max Packet Size: 32
        Attributes : 000000011
```

FIGURE A.4. Android USB probe results on Emotive USB dongle

A.2.3. EEG and other data construction form the 32 byte data packets

Once I obtained the key, it was used to decrypt and obtain the 32byte data frames. According to Emokit documentation [26], these frames were segmented as follows;

- Sensor Data - 128hz
- Gyro Data - 128hz
- Battery - 1hz
- Sensor Quality - 1hz-16hz (Depends on the sensor)

Bit Indexes	0:07	8:21	22:35	36:49	50:63	64:77
Used for	Counter/Battery	F3 Data	FC5 Data	AF3 Data	F7 Data	T7 Data
Bit Indexes	78:91	92:105	107:120	121:133	134:147	148:161
Used for	P7 Data	O1 Data	Connection Quality	?	O2 Data	P8 Data
Bit Indexes	162:175	176:189	(Rotating) 190:203	204:217	218:231	233:239
Used for	T8 Data	F8 Data	AF4 Data	FC6 Data	F4 Data	Gyro X
Bit Indexes	240:247	248:255				
Used for	Gyro Y	?				

TABLE A.1. The bit ranges for each raw data.

Since each data frame has 256bits (i.e 32bytes), different ranges of bits represent different values. Emokit documentation describes this in detail. Table A.1 explains the EEG raw data representation in the data frame. A special mechanism was followed to obtain the battery power and EEG sensor contact quality as explained in the Emokit documentation.

A.2.4. Saving the Raw data

The Emotiv device produced 32 byte raw data frames at a rate 128Hz. Each data packet consisted of the basic following information.

- Packet Counter
- Battery Level
- Contact Quality
- Contact EEG Sensor Readings
- Gyro Sensor Readings

To save the EEG data to a file, I created a comma separated text file (CSV) that consisted of fields for the above data list. Since the data packets were produced without a time stamp from the device, a time stamp was created by using the nano time of the mobile phone. Therefore, in addition to the above fields, the nanotime was also added at the first column for time stamping. A sample of the recoded file is shown in the figure A.5.

```
tm cnt batt gyrox gyroy AF3 AF4 F3 F4 F7 F8 FC5 FC6 O1 O2 P7 P8 T7 T8 q_AF3 q_AF4 q_F3 q_F4 q_F7 q_F8 q_FC5 q_FC6 q_O1 q_O2 q_P7
q_P8 q_T7 q_T8
1.93E+13 0 93 0 0 7630 8483 9006 7662 8780 9652 8773 7678 8569 9202 8518 8554 8451 8363 998 1014 984 950 1030 962 992 986 1064 908
1072 728 1044 940
1.93E+13 4 93 0 -1 7659 8510 9030 7692 8818 9659 8799 7706 8607 9222 8595 8604 8477 8401 998 1014 984 950 1030 962 992 986 1064
908 1072 728 1044 940
1.93E+13 5 93 0 -1 7649 8477 9018 7678 8822 9661 8787 7699 8605 9242 8601 8587 8482 8399 998 1014 984 950 1030 962 992 986 1064
908 1072 728 1044 940
1.93E+13 6 93 1 -1 7659 8484 9024 7690 8814 9688 8795 7715 8596 9226 8604 8576 8506 8420 998 1014 984 950 1030 962 992 986 1064
908 1072 728 1044 940
1.93E+13 7 93 1 -1 7656 8503 9031 7697 8790 9696 8796 7721 8570 9209 8576 8593 8493 8426 998 1014 984 950 1030 962 992 986 1064
906 1072 728 1044 940
1.93E+13 8 93 1 -1 7656 8500 9036 7697 8819 9696 8801 7724 8554 9213 8567 8602 8477 8433 998 1014 984 950 1030 962 992 986 1064
906 1072 726 1044 940
1.93E+13 12 93 1 -1 7661 8478 9024 7690 8830 9681 8799 7692 8581 9217 8602 8581 8497 8407 998 1014 984 950 1030 962 992 986 1064
906 1072 726 1044 940
```

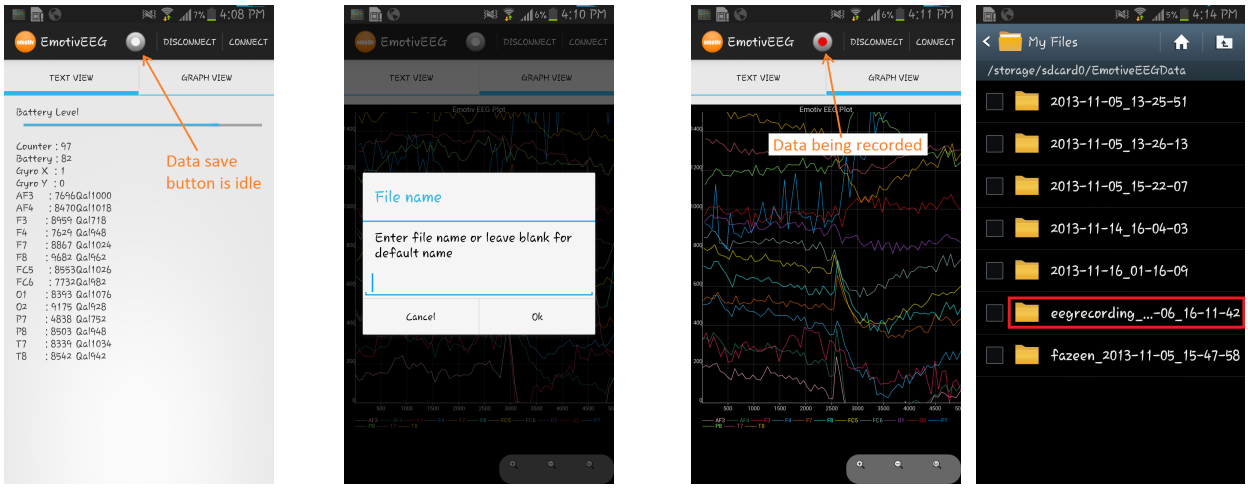
FIGURE A.5. Sample of a recorded csv data file.

An action bar button is introduced to handle data recording. This is a toggle button where it initiate and ends data recording. When the system is not recording to a file, the button will show a grey color that depicts idle. See figure A.6(a). Once this button is pressed, it will bring up a dialog box to enter a file name. See figure A.6(b). The user can enter a file name or the field can be left blank for a default file name. This is convenient when instant data recording is required. Once it starts recording, the button will turn into a red color to indicate that the data is being recorded to a file. See figure A.6(c). This button can be pressed again to stop recording. The recorded file will be placed in a folder under the provided name with the current date and time. See figure A.6(d)

A.2.5. Realtime Visualization in the Mobile Application

EEG data visualization on the mobile phone is implemented using a tool called "AChartEngine" for Android [2]. The tool provides APIs to implement different types of charts for Android devices and I have utilized this open source tool to implement and visualize EEG graphs in the app. The chart type used in this work is a line chart. See figure A.7(b) for the data visualizing of the app in real time.

Data were presented in two different views: one as a numerical view and the other as a graphical view. Each view is accessed by a user interface (UI) tab. Initially the 'Text View' tab



(a) Application is idle and not recording. Check the color of the record button.

(b) When record button is pressed, bring up this dialog to enter a file name. If the file name left blank the system will use a default name

(c) Data being recoded. Note the data visualization.

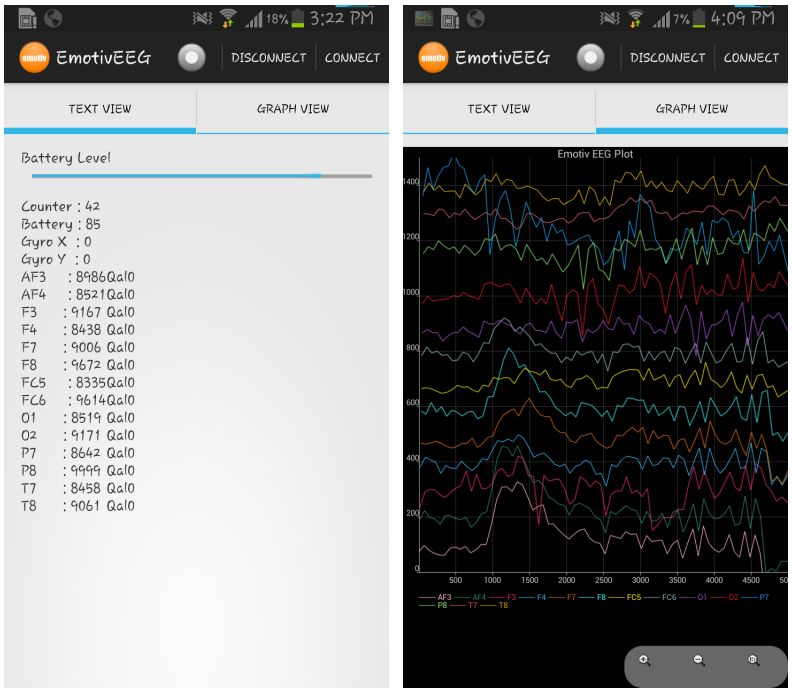
(d) Recorded data file folder in the mobile SD card. Browsed by the file viewer

FIGURE A.6. Screen shots on recording EEG data to a file.

is displayed and it will depict the numerical values of the raw data. See figure A.7(a). When the 'Graph View' tab is pressed it will initiate the graphical view of the data and plots all 14-channel EEG data in real time. See figure A.7(b). When this tab is selected, the system will disable the background updating of the numerical view. However, if the 'Text View' is selected, the system will update the graphical view in the background. Otherwise, graphical view will lose some plotting data during the text view is selected.

A.2.6. Fast Fourier Transformation (FFT) in the Mobile Phone

Performing the frequency domain analysis on EEG data is very important in processing EEG data. Hence, the mobile app should be capable of performing the FFT on EEG data in realtime. To perform this, I utilized an external FFT java library called JTransforms. This library is originally designed to perform FFT on the regular pure Java platforms [101]. Porting this library to the Android platform was not a difficult task. Only extra memory needed to be allocated in the Android toolkit in eclipse for this library to compile properly. However, applying FFT for EEG data in a mobile phone was challenging due to its computational complexity. Since EEG produces a large amount of data it is important to determine whether the FFT can be applied in a feasible



(a) Text view of the EEG data (b) Graphic view of the EEG data

FIGURE A.7. Screen shots on EEG data visualizing.

amount of time.

By integrating the JTransform tool, I have obtained the FFT of a one second window of each channel EEG recording. For each second, the mobile phone performed FFT for all fourteen channels. According to the performance statistics, the Samsung Note II performed this within about 8.9ms. See the table A.2 for FFT performance results. This is feasible enough to perform any other derived calculation after applying FFT on the EEG data. Since this app is capable of obtaining frequency bands in an efficient time frame, one can use this to train machine learning models on the mobile platform for atomic events.

A.3. Emotive Data Comparison: PC vs Mobile

The data recorded from the mobile device was exactly the same as that of a PC except the file format to which they have been stored. Based on EEG studies, it is a known fact that meditation or baseline recording while the eyes are closed significantly depicts the Alpha band frequency (i.e. it will show higher activity in the frequency band of 8-12Hz). So I performed this experiment to capture the Alpha frequency from a PC recording and from a mobile recording. Then they can be

		Time in (ns)	Time in (ms)
I/System.out(2312)	Time for FFT	8140124	8.14
I/System.out(2312)	Time for FFT	4953792	4.95
I/System.out(2312)	Time for FFT	3479958	3.48
I/System.out(2312)	Time for FFT	3709083	3.71
I/System.out(2312)	Time for FFT	4177166	4.18
I/System.out(2312)	Time for FFT	3847792	3.85
I/System.out(2312)	Time for FFT	3951500	3.95
I/System.out(2312)	Time for FFT	4715917	4.72
I/System.out(2312)	Time for FFT	6817833	6.82
I/System.out(2312)	Time for FFT	8554666	8.55
I/System.out(2312)	Time for FFT	15673083	15.67
I/System.out(2312)	Time for FFT	6110916	6.11
I/System.out(2312)	Time for FFT	3768000	3.77
I/System.out(2312)	Time for FFT	4966166	4.97
I/System.out(2312)	Time for FFT	3583166	3.58
I/System.out(2312)	Time for FFT	5005209	5.01
I/System.out(2312)	Time for FFT	16232833	16.23
I/System.out(2312)	Time for FFT	3643917	3.64
I/System.out(2312)	Time for FFT	3400625	3.4
I/System.out(2312)	Time for FFT	39166167	39.17
I/System.out(2312)	Time for FFT	16419833	16.42
I/System.out(2312)	Time for FFT	20538500	20.54
I/System.out(2312)	Time for FFT	18531417	18.53
I/System.out(2312)	Time for FFT	4092707	4.09
	Average	8895015.41666667	8.9

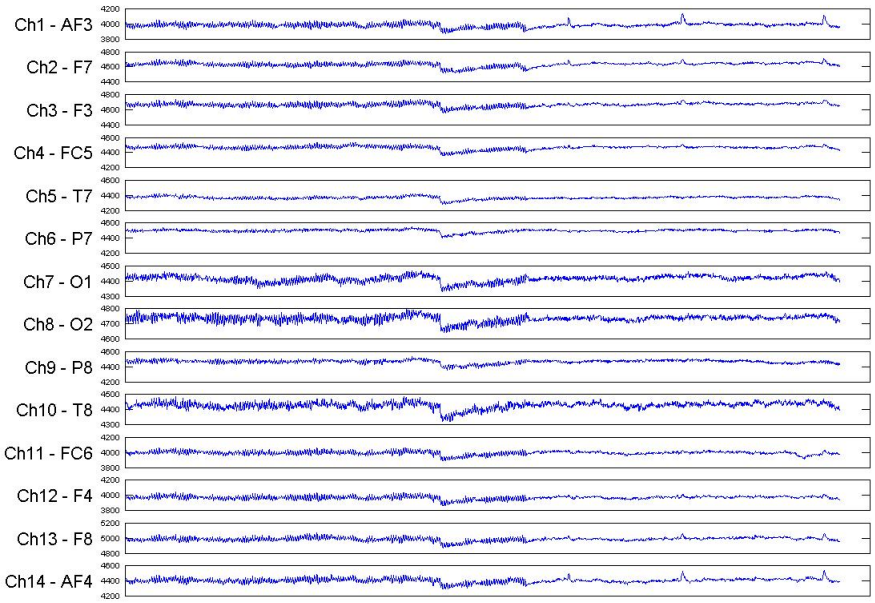
TABLE A.2. FFT performance results - Here each record represents how long it took the Samsung Note II phone to perform 1sec worth of all 14-channel EEG data. The data sampling rate is 128Hz.

compared to see the similarity of the data recording.

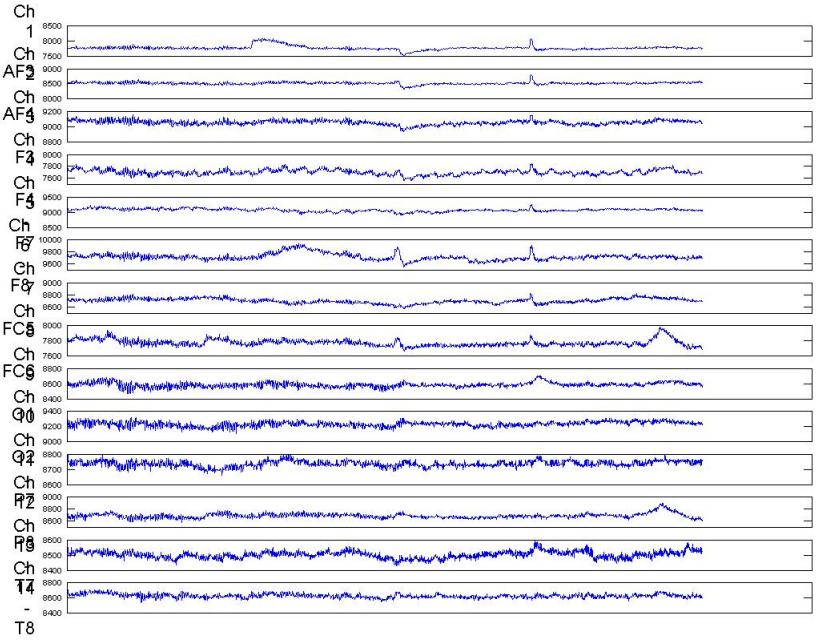
To capture the Alpha waves on both devices, a subject was asked to perform the following activity. First the subject was asked to rest (while their eyes were open) for 10 seconds. Then the subject was asked to close their eyes and rest for another 10 seconds. The EEG activities are recorded on both PC and mobile devices in two different instances while performing this activity. Two different instances were used due to both PC and mobile cannot record data from a single head set at the same time.

Results are shown in figure A.8(a) and A.8(b).

The results show that both graphs were similar (both depicted the expected alpha waves



(a) Data recorded from the PC application



(b) Data recorded from the Mobile application

when the eyes were closed) and this was consistent on multiple instances as well. Therefore, the EEG raw data recordings on the mobile device with my application were the same as the data recorded from the PC software which was provided by the manufacturer.

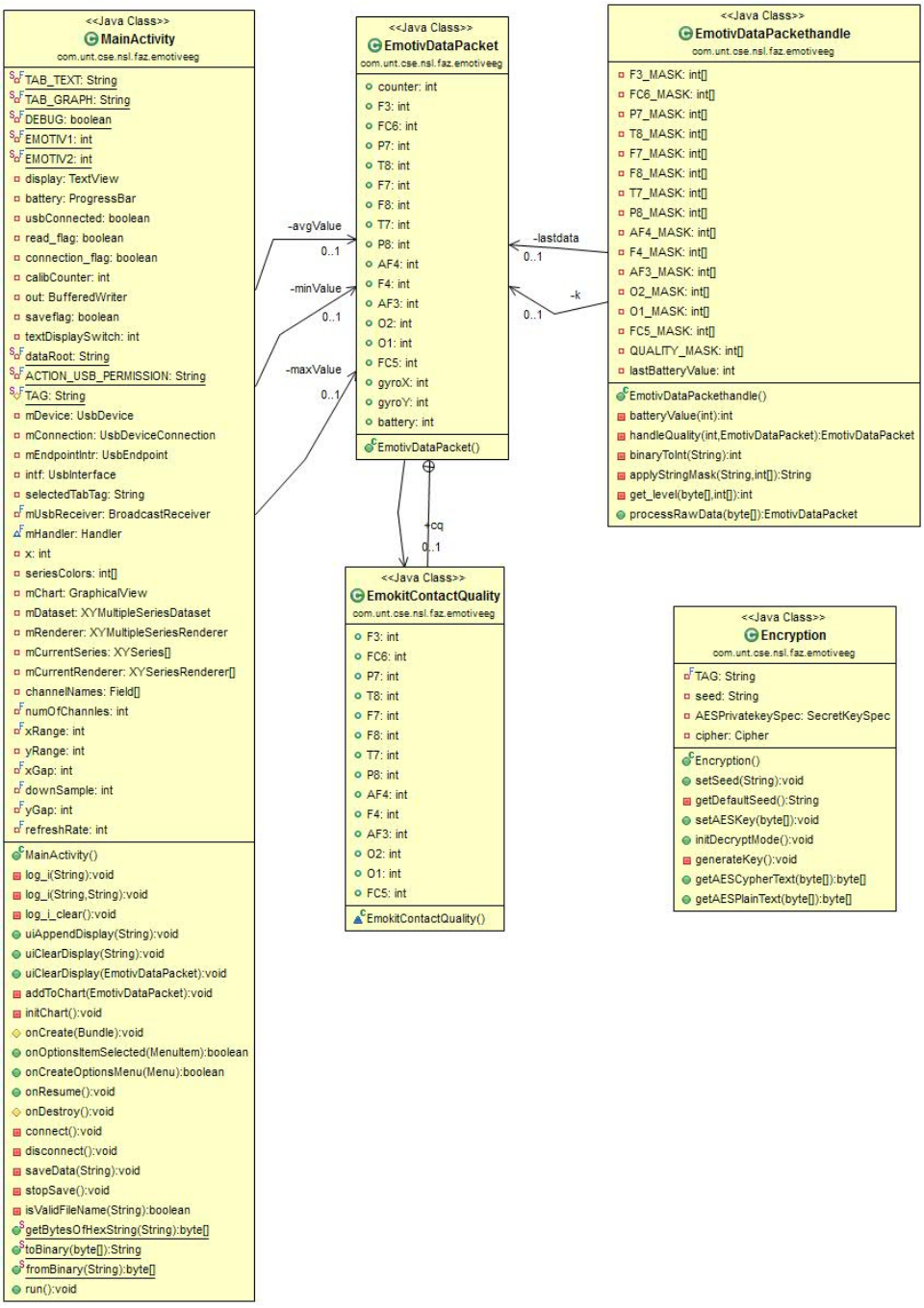


FIGURE A.8. Class diagram.

A.4. The Mobile App and the Setup

The experimental setup and the developed app is shown in figure A.9. Some screenshots of the app is shown in figure A.6. The class diagram of the application is shown in figure A.8. This application enabled multichannel EEG data recording and processing in an Android mobile device. This interfacing could be done with a minimal hardware and software requirements. Only additional required instrument was an 'On the Go' (OTG) adapter other than the main EMOTIV instrument.

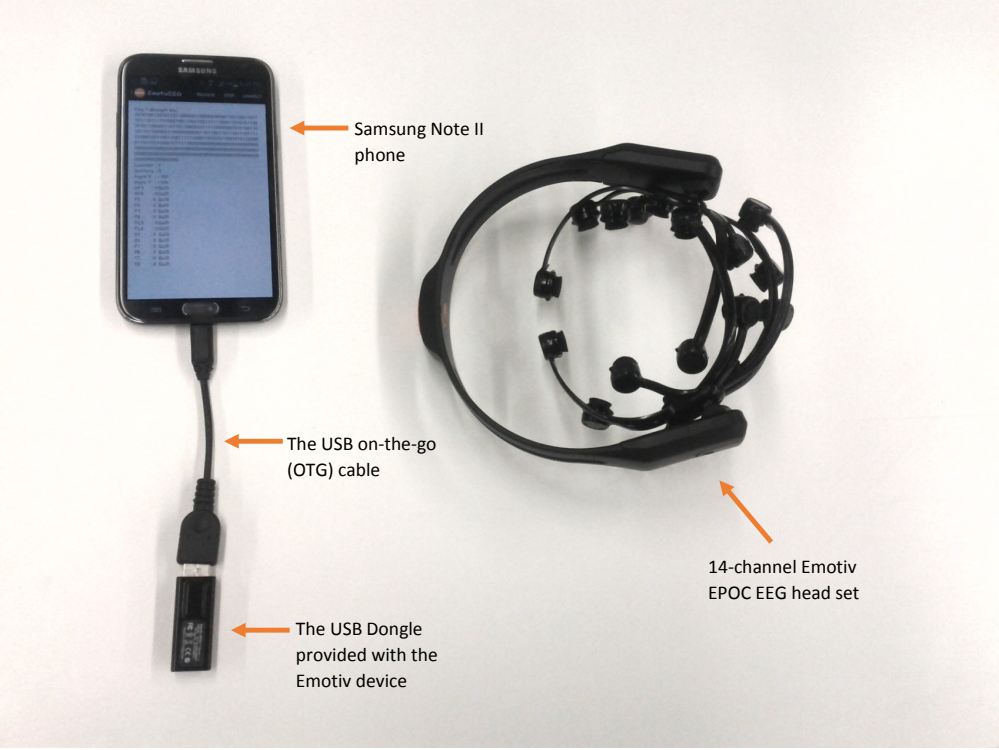


FIGURE A.9. Experimental setup to read data from the EEG headset.

APPENDIX B

MOBILE PLATFORM RESOURCES AND HARDWARE

B.1. Introduction

Currently, smartphones are equipped with various kinds of sensors. The main usages of these sensors vary from recognizing the orientation to reading the finger print of the user. Most of these sensors can be utilized in more innovative ways than its intended actual purpose [52]. For instance, the intended task of the accelerometer is to orient the content of the phone screen based on its orientation. However, P. Mohan et al., utilized this accelerometer to sense the road and traffic conditions [65]. J. Eriksson et al. also described on how to use the same accelerometer to monitor potholes in the road [33]. Similarly, all the other equipped sensors can be utilized in more productive and innovative applications and W. Z. Khan et al. summarized some of these interesting applications [52]. When looking at these capabilities, sensor data can cause many security and privacy issues. I have published an abstract paper with C. Claiborne, and R. Dantu on such security and privacy issues with these sensors [22]. The paper also proposed a framework to anonymize the sensor data when needed. Thus, it is necessary to understand some of the major sensors in a smartphone and be aware of their capabilities when providing security for these systems. This section presents several smartphone sensors and their capabilities.

In this research, I used several Android devices ranging from Google Nexus One to Samsung Note II. The table B.1 summarizes some of the devices used in this research and their available sensors. Some of these sensors and its performances are discussed in this chapter.

B.2. Accelerometer

Acceleration is the rate of change of velocity in time 12. The SI unit for acceleration is ms^2. The accelerometer sensor measure the acceleration in x, y, z axes, this is called the 3-axis accelerometer. According to android developers, generally the coordinate system of most of the sensors including the acclerometer sensor is defined as shown in figure B.1 [43]. This coordinate system is set relative to the orientation of the mobile phone's screen.

Device Name	Manufacturer	Processor	Available Sensors
Google Nexus One	HTC Cooperation	1 GHz Scorpion	Accelerometer, Proximity, Compass, GPS, Bluetooth, Wi-Fi, Touchscreen, Camera
Google Nexus S	Samsung	1 GHz Cortex-A8	Accelerometer, Gyroscope, Proximity, Light, NFC, Compass, GPS, Bluetooth, Wi-Fi, Touchscreen, Camera
Google Galaxy Nexus	Samsung	Dual-core 1.2 GHz Cortex-A9	Accelerometer, Gyroscope, Barometer, Proximity, Light, NFC, Compass, GPS, Bluetooth, Wi-Fi, Touchscreen, Camera
Google Nexus 4	LG Electronics	Quad-core 1.5 GHz Krait	Accelerometer, Gyroscope, Barometer, Proximity, Light, NFC, Compass, GPS, Bluetooth, Wi-Fi, Touchscreen, Camera
Samsung Note II N7100	Samsung	Quad-core 1.6 GHz Cortex-A9	Accelerometer, Gyroscope, Barometer, Proximity, Light, NFC, Compass, GPS, Bluetooth, Wi-Fi, Touchscreen, Camera

TABLE B.1. Android smart phone devices used in this work.

$$\text{acceleration}(\alpha) = \frac{\Delta v}{\Delta t} = \frac{\text{velocity change}}{\text{time difference}} \tag{12}$$

The Google Nexus One is equipped with BMA 150 digital triaxial accelerometer sensor. It has a sensitivity range of 2g/4g/8g with a max axial refresh rate of 3300 Hz. The limitations of the refresh rate and software integration yield a usable refresh rate around 2530 Hz [35, 44].

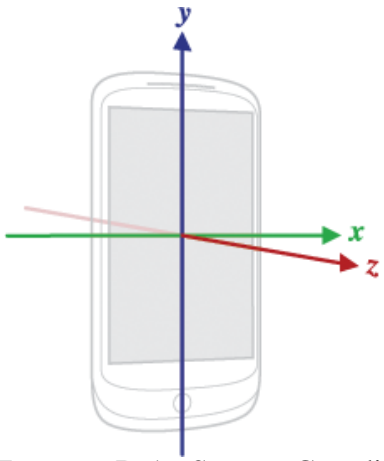


FIGURE B.1. Sensor Coordinate System used in Android API. [43]

B.2.1. Accelerometer Accuracy

None of the sensors are accurate. Sensors are incorporated with intrinsic errors and these must be minimized by adding error correction to the measurements. The accelerometer sensor accuracy was tested in a lab experiment to verify how accurate it

(a) Phone Attachment

(b) Experimental Setup

FIGURE B.2. Experimental setup for accelerometer accuracy test.

is compared to the actual acceleration.

Comparing with the Ground Truth: A series of tests was performed which aimed at testing the accuracy of the Bosch BMA150 accelerometer in the Nexus One. To test the accuracy of the phone, I utilized a bicycle-like wheel that has very good bearings. This provides little friction but is still present as the wheel eventually comes to a complete stop. I attached the phone to the end of the wheel parallel to the rotation of the wheel which is clamped down and immobile. The phone is in the same holster that is used for the testing inside the vehicle. I spun the wheel and set the phone to record accelerometer values. Figure B.2 illustrates the experimental setup used for the accuracy test[35, 44].

The experiment was recorded in which the timings per phone revolution around the wheel were analyzed. To compare with the accelerometer data from the phone, I calculated the averaged angular acceleration at different revolutions around the wheel in which the phone experienced. Since I used a wheel apparatus, centripetal acceleration or force was calculated in the y-axis and then compared with a calculated value. To describe the experiment, a sensor recording application was started on the phone and placed in the holster which was attached to the outside of the wheel by velcro. At the same time, a camcorder began recording the experiment which was later used to obtain the time per phone/wheel revolution. The experimental acceleration a_c was calculated using the radius of the wheel (r), angular velocity (ω) and r=0.336

$$a_c = \frac{(v_t^2)}{r} = \omega^2 r \tag{13}$$

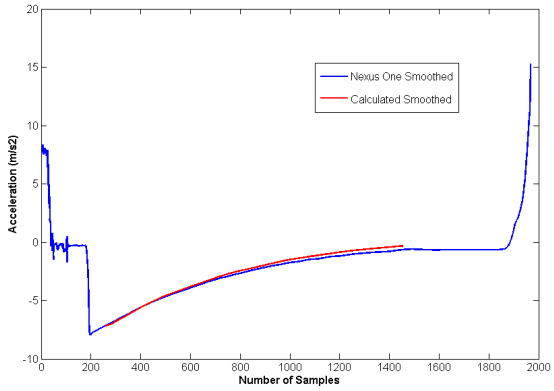


FIGURE B.3. Experimental and recoded acceleration data comparison.

After the centripetal acceleration was calculated using 13, it was compared against the z-axis data from the phone. The z-axis was used because I had the phone oriented parallel with the wheel and the direction of the centripetal acceleration is always inwards along the radius vector of the circular motion. Figure B.3 illustrates this analysis visually while I also calculated both the Spearman and Pearson correlation value between the two accelerations. Test 1 resulted in a Pearson correlation of 0.9997 and a Spearman correlation of 1 while Test 2 resulted in a Pearson correlation of 0.9978 and a Spearman correlation of 0.9979. These high correlation values convey that the phone is sufficiently accurate. [35, 44]

B.3. Gyroscope

The gyroscope is a sensor in the mobile phone that outputs the angular rotational velocity about the three axes shown in figure B.1. The units are in radians/second and this is a more reliable sensor to determine the orientation angle of the mobile phone. Gyroscope sensor model built in the Google Nexus S is K3G manufactured by STMicroelectronics. Its angular velocity ranges are in 250/500/2000dps. The sampling rates for the gyroscope is 100 Hz [10].

B.4. Near Field Communication (NFC)

Near Field Communication (NFC) is relatively a new type of sensor equipped in most of the new Android devices. Its main purpose is to make a short-range wireless communication (within about 4 cm apart). Google wallet is one of the main applications of this technology. Recently released Apple iPhone 6 and 6 plus are now equipped with NFC sensor for their Apple Pay services.

It is also used to read RF id tags. It can be used to share small payloads of data between an NFC tag and an Android-powered device, or between two Android-powered devices [43].

In hardware level, NFC uses an induction coil to transmit data from the induction current. To show the induction power, several experiments were performed with the NFC antenna. In this set of experiments, the electric potential induced by two audio speakers and the power supply of one of these speakers were measured and tested. The power supply of the speaker (AC to DC transformer) can induce a significant amount of electric potential in the NFC antenna.

B.4.1. Experiment - Measuring the Induced Voltage in NFC Antenna

The speakers I used in this experiment were a pair of small computer speakers (smaller speaker) and a mini hifi speaker (bigger speaker). Since the smaller speaker did not show any significant magnetic field fluctuations it was excluded in this experiment to measure the induced voltage. Though the bigger speaker was used in the experiment to measure the induced voltage it did not produce much of a voltage and the detected voltages were in micro volts. However, the power supply of the speaker induced a significant amount of voltage. And also this induced voltage increased with the volume of the speaker as it used more power for high volume.

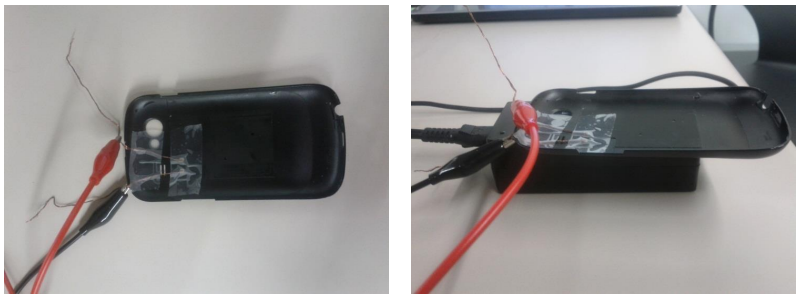
Experimental Setup

First, the mobile phones back cover with the NFC antenna was dismounted from the mobile phone and the two terminals of the antenna was connected with two metal wires. Then it was connected to a high precision volt meter. See figure B.4(a). Then the NFC antenna was placed on top of the power supply as depicted in figure B.4(b).

First the induced voltage was measured before turning on the speaker. The speaker had 10 different speaker volume levels. For each volume level a sound of 60Hz were played in the speaker and the induced voltage in the NFC antenna was measured.

Results

When the NFC antenna was placed on the speaker and played a sound of 60Hz with full volume, it only produced an induction voltage of 0.06 mV. But when it was placed on the power supply the induced voltage increased; figure B.5 depicts the results. The power supply of the



(a) NFC antenna terminal connected with two metal wires

(b) NFC antenna placed on top of the power supply

FIGURE B.4. Experimental setup for NFC sensor induction current testing.

speaker is a transformer and near this the NFC antenna can induce a significant amount of voltage. The bigger speaker used in here did not induce significant amount of voltage when it was played even on full volume. The higher the volume of the speaker, higher the induced voltage from its power supply. It is intuitive that higher volume draws higher energy and this resulted in inducing higher voltages.

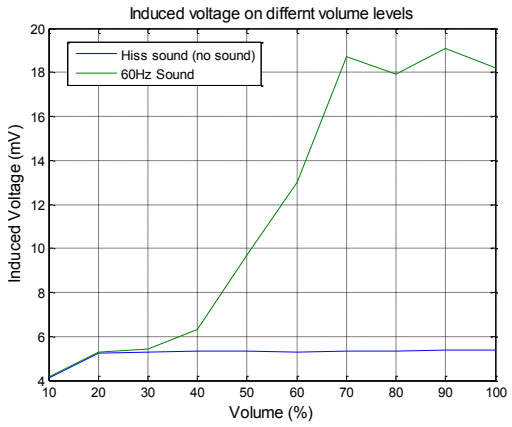


FIGURE B.5. Voltage induction results.

BIBLIOGRAPHY

[1] *CMUSphinx: The Carnegie Mellon Sphinx Project, URL: http://cmusphinx.sourceforge.net/html/cmusphinx.php*. 82, 85

[2] AChartEngine, *Achartengine,* `http://www.achartengine.org/`, Nov 6 2013. 126

[3] Matt Adcock, Jaewoo Chung, and Chris Schmandt, *Arewethereyet?: a temporally aware media player*, Proceedings of the ninth conference on Australasian user interface - Volume 76 (Darlinghurst, Australia, Australia), AUIC '08, Australian Computer Society, Inc., 2008, pp. 29–32. 88

[4] Emily Adler, *Social media engagement: The surprising facts about how much time people spend on the major social networks*, Bussiness Insider, Tech, `http://www.businessinsider.com/social-media-engagement-statistics-2013-12` (2014). 3

[5] J. Al-Muhtadi, R. Hill, R. Campbell, and M.D. Mickunas, *Context and location-aware encryption for pervasive computing environments*, Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on, march 2006, pp. 6 pp. –289. 88

[6] R. Anderson, M. Bond, J. Clulow, and S. Skorobogatov, *Cryptographic processors-a survey*, Proceedings of the IEEE 94 (2006), no. 2, 357–369. 82

[7] Bob Bailey, *What are typical human interaction speeds for reading, listening, speaking, keying, and handwriting?*, 3-day 1999 Annual User Interface Update Seminar, Insights from Human Factors International, Inc. (HFI), 2000. 78

[8] Garima Bajwa and Ram Dantu, *Abstract: Cryptographic key generation using electroencephalograms*, The Learning from Authoritative Security Experiment Results (LASER) workshop, `http://www.laser-workshop.org/prior-workshops/2013/program/bajwa-abstract/` (2014). 22

[9] David Barrera, H. Güneş Kayacik, Paul C. van Oorschot, and Anil Somayaji, *A methodology*

for empirical analysis of permission-based security models and its application to android, Proceedings of the 17th ACM conference on Computer and communications security (New York, NY, USA), CCS '10, ACM, 2010, pp. 73–84. 48

[10] Christopher Barthold, Kalyan Pathapati Subbu, and Ram Dantu, *Evaluation of gyroscope-embedded mobile phones*, SMC, 2011, pp. 1632–1638. 136

[11] Armen Khodaverdian Benjamin Davis, Ben Sanders and Hao Chen, *I-arm-droid: A rewriting framework for in-app reference monitors for android applications*, IEEE Mobile Security Technologies (MoST) (San Francisco, CA), May 2012. 50

[12] Christopher M. Bishop, *Pattern recognition and machine learning (information science and statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 26, 28

[13] L. Breiman, *Random forests*, Machine Learning 45 (2001), no. 1, 5–32. 62, 64

[14] K.H. Brodersen, Cheng Soon Ong, K.E. Stephan, and J.M. Buhmann, *The balanced accuracy and its posterior distribution*, Pattern Recognition (ICPR), 2010 20th International Conference on, aug. 2010, pp. 3121 –3124. 46

[15] Andrew Campbell, Tanzeem Choudhury, Shaohan Hu, Hong Lu, Matthew K. Mukerjee, Mashfiqui Rabbi, and Rajeev D.S. Raizada, *Neurophone: brain-mobile phone interface using a wireless eeg headset*, Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds (New York, NY, USA), MobiHeld '10, ACM, 2010, pp. 3–8. 119

[16] G. Canfora, F. Mercaldo, and C.A. Visaggio, *A classifier of malicious android applications*, Availability, Reliability and Security (ARES), 2013 Eighth International Conference on, Sept 2013, pp. 607–614. 51

[17] M. Cha, A. Mislove, and K. P. Gummadi, *A measurement-driven analysis of information propagation in the flickr social network*, Proc. 18th int. conf. on World wide web, 2009, pp. 721–730. 52

[18] Zheng Chen, Fan Lin, Huan Liu, Yin Liu, Liu Wenyin, and Wei ying Ma, *User intention modeling in web applications using data mining*, Wide Web: Internet and Web Information Systems (The WWW Journal, 2002, pp. 181–191. 5

[19] M. Cheong and V. Lee, *Integrating web-based intelligence retrieval and decision-making from the twitter trends knowledge base*, Proc. 2nd ACM workshop on social web search and mining (Hong Kong), 2009, pp. 1–8. 53

[20] John Chuang, Hamilton Nguyen, Charles Wang, and Benjamin Johnson, *I think, therefore i am: Usability and security of authentication using brainwaves*, Financial Cryptography and Data Security (AndrewA. Adams, Michael Brenner, and Matthew Smith, eds.), Lecture Notes in Computer Science, vol. 7862, Springer Berlin Heidelberg, 2013, pp. 1–16 (English). 8, 22

[21] Pablo Cingolani, *jfuzzylogic, open source fuzzy logic library and FCL language implementation*, *Available: http//jfuzzylogic.sourceforge.net/html/index.html* (2010). 57

[22] Cynthia Claiborne, Mohamed Fazeen, and Ram Dantu, *Android sensor data anonymization (back matter)*, Research in Attacks, Intrusions, and Defenses (SalvatoreJ. Stolfo, Angelos Stavrou, and CharlesV. Wright, eds.), Lecture Notes in Computer Science, vol. 8145, Springer Berlin Heidelberg, 2013, p. 469471 (English). 133

[23] Graham Cluley, *Revealed! the top five android malware detected in the wild*, nakedsecurity (2012). 3

[24] P.M. Comar, Lei Liu, S. Saha, Pang-Ning Tan, and A. Nucci, *Combining supervised and unsupervised learning for zero-day malware detection*, INFOCOM, 2013 Proceedings IEEE, April 2013, pp. 2022–2030. 113

[25] International Electrotechnical Commission, *Technical committee no. 6: Industrial process measurement and control. sub-committee 65 b: Devices. iec 1131-programmable controllers*, *Available: http://www.fuzzytech.com/binaries/ieccd1.pdf* (1997). 57

[26] Eguru Daeken, *Emokit documentation@ONLINE*, May 2012. 123, 124

[27] Wei Dai, *Crypto++ 5.6.0 benchmarks @ONLINE http://www.cryptopp.com/benchmarks.html*, (2009). 78

[28] Damon V. Danieli, *Automatic censorship of audio data for broadcast*, 10 2008. 86

[29] Ram Dantu, Sonia Fahmy, Henning Schulzrinne, and Joo Cangussu, *Issues and challenges in securing voip*, Computers & Security 28 (2009), no. 8, 743 – 753. 77

[30] Google Android Developer, *Introduction to android*, 2014. 15

[31] _____, *System permissions*, 2014. 16

[32] Allen B. Downey, *Think stats: Probability and statistics for programmers*, vol. 1.5.9, Green Tea Press, 2011. 17, 42

[33] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan, *The pothole patrol: using a mobile sensor network for road surface monitoring*, Proceedings of the 6th international conference on Mobile systems, applications, and services (New York, NY, USA), MobiSys '08, ACM, 2008, pp. 29–39. 133

[34] fsecure, *Mobile threat report q2 2012*, Tech. Report Q2 2012, F-Secure Response Labs, Auguest 2012. 2

[35] M. Fazeen, B. Gozick, R. Dantu, M. Bhukhiya, and M.C. Gonzalez, *Safe driving using mobile phones*, Intelligent Transportation Systems, IEEE Transactions on 13 (2012), no. 3, 1462–1468. 134, 135, 136

[36] Mohamed Fazeen, Garima Bajwa, and Ram Dantu, *Context-aware multimedia encryption in mobile platforms*, Proceedings of the 9th Annual Cyber and Information Security Research Conference (New York, NY, USA), CISR '14, ACM, 2014, pp. 53–56. 77

[37] Mohamed Fazeen and Ram Dantu, *Another free app: Does it have the right intentions?*, Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on, July 2014, pp. 282–289. 3

[38] Mohamed Fazeen, Ram Dantu, and Parthasarathy Guturu, *Identification of leaders, lurkers, associates and spammers in a social network: context-dependent and context-independent approaches*, Social Network Analysis and Mining 1 (2011), no. 3, 241–254 (English). 52

[39] Adrienne P. Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner, *Android permissions demystified*, Proceedings of the 18th ACM conference on Computer and communications security (New York, NY, USA), CCS '11, ACM, October 2011, pp. 627–638. 50

[40] Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee, *Polymorphic blending attacks*, Proceedings of the 15th Conference on USENIX Security Sym-

posium - Volume 15 (Berkeley, CA, USA), USENIX-SS'06, USENIX Association, 2006. 3

[41] Mario Frank, Ben Dong, Adrienne Porter Felt, and Dawn Song, *Mining permission request patterns from android and facebook applications (extended author version)*, CoRR abs/1210.2429 (2012). 48

[42] FR) Gomez, Laurent Y. (Le Cannet, *Context-aware based cryptography*, November 2007. 89

[43] Google, *Android developers*, 05 2013. xi, 133, 134, 137

[44] Brandon Gozick, *A Driver, Vehicle and Road Safety System Using Smartphones*, Masters thesis, University of North Texas, Denton, Texas, 5 2012. 134, 135, 136

[45] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten, *The weka data mining software: an update*, SIGKDD Explor. Newsl. 11 (2009), no. 1, 10–18. 33, 101

[46] Inquisitr, *Smartphone users fail to protect personal data,* `http://www.inquisitr.com/666421/smartphone-users-fail-to-protect-personal-data/`, 2013. 4

[47] Stylianos Karapantazis and Fotini-Niovi Pavlidou, *Voip: A comprehensive survey on a promising technology*, Computer Networks 53 (2009), no. 12, 2050 – 2090. 79

[48] Clare-Marie Karat, Christine Halverson, Daniel Horn, and John Karat, *Patterns of entry and correction in large vocabulary continuous speech recognition systems*, Proceedings of the SIGCHI conference on Human Factors in Computing Systems (New York, NY, USA), CHI '99, ACM, 1999, pp. 568–575. 78

[49] Martin Kennedy, Hrishikesh Venkataraman, and Gabriel-Miro Muntean, *Energy consumption analysis and adaptive energy saving solutions for mobile device applications*, Green IT: Technologies and Applications (JaeH. Kim and MyungJ. Lee, eds.), Springer Berlin Heidelberg, 2011, pp. 173–189. 79

[50] AD. Keromytis, *A comprehensive survey of voice over ip security research*, Communications Surveys Tutorials, IEEE 14 (2012), no. 2, 514–537. 3

[51] W. Khalifa, A Salem, M. Roushdy, and K. Revett, *A survey of eeg based user authentication schemes*, Informatics and Systems (INFOS), 2012 8th International Conference on, May 2012, pp. BIO–55–BIO–60. 22

[52] W.Z. Khan, Yang Xiang, M.Y. Aalsalem, and Q. Arshad, *Mobile phone sensing systems: A survey*, Communications Surveys Tutorials, IEEE 15 (2013), no. 1, 402–427. 133

[53] S. Kim and S. Han, *The method of inferring trust in web-based social network using fuzzy logic*, Proc. Int. workshop on Machine Intelligence Research, vol. 2, December 2009, pp. 140–144. 53

[54] Laura A. King, *Experience psychology*, 2nd edition ed., McGraw-Hill Higher Education, Jan 2013. x, 92, 93, 94, 95

[55] N. Klym and M. J. Montpetit, *Innovation at the edge: Social TV and beyond*, September 2008. 52

[56] Dave Lee, *Google glass hack allows brainwave control*, BBC News Technology (2014). 10

[57] Y. Lin, H. Sundaram, Y. Chi, J. Tatemura, and B. Tseng, *Splog detection using content, time and link structures*, Proc. IEEE Int. Conf. on Multimedia and Expo, July 2007, pp. 2030–2033. 53

[58] D. E. J. Linden, *The P300: Where in the Brain Is It Produced and What Does It Tell Us?*, Neuroscientist 11 (2005), 563–576. 119

[59] Hiroshi Lockheimer, *Android and security*, Google Mobile Blog (2012). 2

[60] M. Naslund E. Carrara M. Baugher, D. McGrew and K. Norrman, *The real-time transport protocol (rfc 3711)*, PROPOSED STANDARD, Network Working Group, Online. Available: http://tools.ietf.org/html/rfc3711 (2004). 82

[61] W. Ma, D. Tran, and D. Sharma, *A novel spam email detection system based on negative selection*, Proc. 4th Int. conf. Comp. Sci. and Convergence Information Technology (Seoul), November 2009, pp. 987–992. 62

[62] Anil Madhavapeddy, David Scott, and Richard Sharp, *Context-aware computing with sound*, IN PROCEEDINGS OF THE 5TH INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING, 2003, pp. 315–332. 88

[63] M. Mathioudakis, N. Koudas, and P. Marbach, *Early online identification of attention gathering items in social media*, Proc. 3rd ACM Int. Conf. web search and data mining, 2010, pp. 301–310. 53

[64] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, *Measurement and analysis of online social networks*, Proc. 7th ACM SIGCOMM conf. on Int. measurement, 2007, pp. 29–42. 52

[65] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee, *Nericell: rich monitoring of road and traffic conditions using mobile smartphones*, Proceedings of the 6th ACM conference on Embedded network sensor systems (New York, NY, USA), SenSys '08, ACM, 2008, pp. 323–336. 133

[66] Gianluca Moro, *Brain computer interface for alternative inpute @ONLINE*, October 2013. 120

[67] Carey Nachenberg, *Method and apparatus for content based encryption*, 06 2006. 89

[68] Neurogadget, *Mind-controlled virtual reality system combines the forces of emotiv epoc, oculus rift and razer hydra*, 2014. 10

[69] Neurowear, *neurowear "neurocam" concept movie*, October 29 2013. 120

[70] _____, *Projects / neurocam,* `http://www.neurowear.com/projects_detail/neurocam.html`, October 29 2013. 120

[71] O. Nhway, *An investigation into the effect of security on reliability and voice recognition system in a voip network*, Advanced Communication Technology (ICACT), 2011 13th International Conference on, feb. 2011, pp. 1293 –1297. 77

[72] A Pantelopoulos and N.G. Bourbakis, *A survey on wearable sensor-based systems for health monitoring and prognosis*, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 40 (2010), no. 1, 1–12. 10

[73] P. Pawar, A. van Halteren, and K. Sheikh, *Enabling context-aware computing for the nomadic mobile user: A service oriented and quality driven approach*, Wireless Communica-

tions and Networking Conference, 2007.WCNC 2007. IEEE, march 2007, pp. 2529 –2534. 78

[74] Wilder Penfield, *Some observations in the cerebral cortex of man*, Proceedings of the Royal Society, 1947, pp. 134, 349. 94

[75] Roberto Perdisci and ManChon U, *Vamo: towards a fully automated malware clustering validity analysis*, Proceedings of the 28th Annual Computer Security Applications Conference (New York, NY, USA), ACSAC '12, ACM, 2012, pp. 329–338. 48

[76] Sarah Perez, *Mobile app usage increases in 2014, as mobile web surfing declines*, Techcrunch, http://techcrunch.com/2014/04/01/mobile-app-usage-increases-in-2014-as-mobile-web-surfing-declines/ (2014). 2

[77] M. K. Petersen, C. Stahlhut, A. Stopczynski, J. E. Larsen, and L. K. Hansen, *Smartphones get emotional: mind reading images and reconstructing the neural sources*, oct 2011. 120

[78] Tien Pham, Wanli Ma, Dat Tran, Phuoc Nguyen, and Dinh Phung, *Eeg-based user authentication in multilevel security systems*, Advanced Data Mining and Applications (Hiroshi Motoda, Zhaohui Wu, Longbing Cao, Osmar Zaiane, Min Yao, and Wei Wang, eds.), Lecture Notes in Computer Science, vol. 8347, Springer Berlin Heidelberg, 2013, pp. 513–523 (English). 22

[79] Tom Pick, *21 vital mobile marketing facts and statistics for 2014*, Bussiness 2 Community, http://www.business2community.com/mobile-apps/21-vital-mobile-marketing-facts-statistics-2014-0850425 (2014). 1

[80] J.R. Pierce, *An introduction to information theory: Symbols, signals & noise*, Dover Science Book, Dover Publications, 1980. 78

[81] The Statistics Portal, *Statistics and facts about mobile app usage, http://www.statista.com/topics/1002/mobile-app-usage/*, 2014. 2

[82] Pew Research Internet project, *Mobile technology fact sheet, http://www.*

pewinternet.org/fact-sheets/mobile-technology-fact-sheet/, 2014. 2

[83] Moo-Ryong Ra, Ramesh Govindan, and Antonio Ortega, *P3: Toward privacy-preserving photo sharing*, CoRR abs/1302.5062 (2013). 89, 90

[84] I. Rassameeroj and Y. Tanahashi, *Various approaches in analyzing android applications with its permission-based security models*, Electro/Information Technology (EIT), 2011 IEEE International Conference on, may 2011, pp. 1 –6. 48

[85] V. Rastogi, Yan Chen, and Xuxian Jiang, *Catch me if you can: Evaluating android anti-malware against transformation attacks*, Information Forensics and Security, IEEE Transactions on 9 (2014), no. 1, 99–108. 3, 19

[86] A Rehman, M. Mustafa, N. Javaid, U. Qasim, and Z.A Khan, *Analytical survey of wearable sensors*, Broadband, Wireless Computing, Communication and Applications (BWCCA), 2012 Seventh International Conference on, Nov 2012, pp. 408–413. 10

[87] F. Roesner, T. Kohno, A. Moshchuk, B. Parno, H.J. Wang, and C. Cowan, *User-driven access control: Rethinking permission granting in modern operating systems*, Security and Privacy (SP), 2012 IEEE Symposium on, may 2012, pp. 224 –238. 15

[88] Raul Rojas, *Neural networks - a systematic introduction*, Springer-Verlag, Berlin, New-York, 1996. 63, 64

[89] Jon Crussell Jeremy Erickson Ryan Stevens, Clint Gibler and Hao Chen, *Investigating user privacy in android ad libraries*, IEEE Mobile Security Technologies (MoST) (San Francisco, CA), May 2012. 50

[90] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, and P.G. Bringas, *On the automatic categorisation of android applications*, Consumer Communications and Networking Conference (CCNC), 2012 IEEE, 2012, pp. 149–153. 20, 51

[91] A. Shabtai, Y. Fledel, and Y. Elovici, *Automated static code analysis for classifying android applications using machine learning*, Computational Intelligence and Security (CIS), 2010 International Conference on, 2010, pp. 329–333. 48

[92] Robert Siciliano, *Not such a tweet: Could your latest follower be a hacker?*

every day, 3.5billion malicious tweets spread spam and viruses, Mail Online, http://www.dailymail.co.uk/sciencetech/article-2034618/Twitter-3-5bn-malicious-tweets-distribute-spam-viruses-daily.html (2012). 3

[93] Jonathan Skillings, *Smartphones outpace feature phones for first time ever*, CNET, http://news.cnet.com/8301-1035_3-57581566-94/smartphones-outpace-feature-phones-for-first-time-ever/ (April 26, 2013). 2

[94] T. Spyrou and J. Darzentas, *Intention modelling: Approximating computer user intentions for detection and prediction of intrusions*, IN: S.K. KATSIKAS, D. GRITZALIS (EDS.), INFORMATION SYSTEM SECURITY, SAMOS, GREEZE, Chapman & Hall, 1996, pp. 319–335. 5

[95] A. Stopczynski, J. E. Larsen, C. Stahlhut, M. K. Petersen, and L. K. Hansen, *A smartphone interface for a wireless EEG headset with real-time 3D reconstruction*, oct 2011. 119

[96] Symantec, *Symantec content encryption for symantec messaging gateway. url: http://www.symantec.com/page.jsp?id=preinstall*, 2013. 90

[97] ERIC A. TAUB, *Dragon audiomining - enable text search of audio files*, Computers & Security (Accessed On: Feb. 2013. URL: http://bits.blogs.nytimes.com/2008/04/02/voip-system-security-time-to-worry-or-maybe-not/?smid=pl-share). 82

[98] John Wagley, *New voip encryption challenges*, Security Management, The magazine (Accessed On: Feb. 2013. URL: http://www.securitymanagement.com/article/new-voip-encryption-challenges-005680). 89

[99] Rob Waugh, *Almost 5% of smartphones lost every year*, McAfee Blog Central, http://blogs.mcafee.com/consumer/almost-5-of-smartphones-lost-every-year (2011). 4

[100] Xuetao Wei, Lorenzo Gomez, Iulian Neamtiu, and Michalis Faloutsos, *Permission evolution in the android ecosystem*, Proceedings of the 28th Annual Computer Security Applications Conference (New York, NY, USA), ACSAC '12, ACM, 2012, pp. 31–40. 50

[101] Piotr Wendykier, *Jtransforms*, Software, July 2011. 127

[102] J. Weng, E. Lim, J. Jiang, and Q. He, *Twitterrank: finding topic-sensitive influential twitterers*, Proc. 3rd ACM Int. Conf. Web search and data mining (New York), 2010, pp. 261–270. 53

[103] John S. Whissell, Charles L.A. Clarke, and Azin Ashkan, *Clustering web queries*, Proceedings of the 18th ACM Conference on Information and Knowledge Management (New York, NY, USA), CIKM '09, ACM, 2009, pp. 899–908. 37

[104] J.S. Whissell and C.L.A. Clarke, *Classification-based clustering evaluation*, Data Mining (ICDM), 2013 IEEE 13th International Conference on, Dec 2013, pp. 1229–1234. 37

[105] Wikipedia, *Twitter*, *Available: http://en.wikipedia.org/wiki/Twitter* (2010). 61

[106] ———, *Electroencephalography*, 2014, [Online; accessed 24-September-2014]. 95

[107] C. Wu and B. Zhou, *Analysis of tag within online social networks*, Proc. ACM 2009 international conf., 2009, pp. 21–30. 52

[108] C. Yeh and S. Chiang, *Revisit bayesian approaches for spam detection*, Proc. 9th Int. conf. for Young Computer Scientists (Hunan), November 2009, pp. 659–664. 62

[109] Hao Zhang, W. Banick, Danfeng Yao, and N. Ramakrishnan, *User intention-based traffic dependence analysis for anomaly detection*, Security and Privacy Workshops (SPW), 2012 IEEE Symposium on, May 2012, pp. 104–112. 12

[110] Bin Zhou, Jian Pei, and WoShun Luk, *A brief survey on anonymization techniques for privacy preserving publishing of social network data*, Proc. ACM SIGKDD Explorations Newsletter, vol. 10, December 2009, pp. 12–22. 53

[111] Yajin Zhou and Xuxian Jiang, *Dissecting android malware: Characterization and evolution*, Security and Privacy (SP), 2012 IEEE Symposium on, may 2012, pp. 95 –109. 3, 24, 47, 50

[112] Yajin Zhou, Zhi Wang, Wu Zhou, and Xuxian Jiang, *Hey, you, get off of my market: Detecting malicious apps in official and alternative Android markets*, Proceedings of the 19th Annual Network & Distributed System Security Symposium, February 2012. 50