EFFECTIVE IDENTITY MANAGEMENT ON MOBILE DEVICES USING MULTI-SENSOR MEASUREMENTS

A Dissertation

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Tao Feng

May 2015

EFFECTIVE IDENTITY MANAGEMENT ON MOBILE DEVICES

USING MULTI-SENSOR MEASUREMENTS

Tao Feng

APPROVED:

Weidong Shi, Chairman
Dept. of Computer Science

Albert Cheng
Dept. of Computer Science

Stephen Huang
Dept. of Computer Science

Shishir Shah
Dept. of Computer Science

Bogdan Carbunar
School of Computing and Information, FIU

Dean, College of Natural Sciences and Mathematics

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor Dr. Weidong Shi for his support, guidance and help. He always tried his best to offer me a better platform to pursue more achievements. I cherish the opportunities and experience he provided me: the SRA internship opportunity, which helped me to gain experience in working in industry; the patent composition, which helped me to understand how the intellectual property system works; a variety of research projects in different domains, which helped me to broad my version; and most importantly, the time he spent to advice, guide, and teach me how to solve research problems. All these would be the fortune of my life and I really appreciate all the things he has done for me. I also want to thank his wife, Yang Lu, for her support, help, and care for me and my family members during my last four years.

I would also like to thank the other professors, Dr. Albert Cheng, Dr. Stephen Huang, Dr. Shishir Shah, and Dr. Bogdan, Carbunar in my research committee. Thank you for your helpful advices and insightful discussions.

I had a great time when I interned at SRA, and I really appreciate the help and guidance from my mentors, Dr. Jun Yang, Dr. Zhixian Yan and Dr. Emmanuel Munguia Tapia. Without their help, I could not make the achievements in SRA. I also want to thank Dr. Haipeng Zhang and Dr. Chenren Xu. It was a great experience to work with you.

My I^2C lab mates helped me a lot for my PhD journey. I really appreciate the support and help from them. Dr. Xi Zhao and Dr. Lei Xu, thanks for your help

and advices on my research and my life. I learned a lot from you and I really enjoy working with you. Yuanfeng Wen and Ziyi Liu, you have helped me so many times, and I cherish the time and experience we had together. Varun Prakash and Pranav Koundinya, your energy and passion always inspire me. Xi Wang, Zhimin Gao, Tzu-Hua Liu, and Nicholas DeSalvo, thank you for your support and help on my research and thank you for being friends of mine. I also want to thank, Dr. Massoud Masoumi, Dr. JongHyuk Lee, Dang Khoa pham, Seunghun Kim, Kunal Parmar, Dainis Boumber, and Olga Datskova. Thank you for being great lab mates and friends. I also want to thank my friends, Xiaokang Chen, Yinyin Ma, Yunzhao Chen, Wei Fang, Qiang Li, Shaoding Du, Gao Li, Yanjun Tan, Lvlin Zeng, Xiaogang Chen, and Fan Gao, who shared my joy and tears and makes my life colorful.

I would like to express my earnest gratitude to my grandparents, my parents and my aunts. All my achievements belong to your unconditional support and help. You spent countless efforts, energy, and time on me. I cannot tell how many times you raised me up and how huge sacrifice you have made for me. I really appreciate for what you have done for me.

And last but not least, my wife, Xue Song, for her love, patience, understanding, and support. I owe so much to my wife, who gave up her own career and academic life, which I know she could make much more achievements than I had made, to support my pursue of PhD. And my one year old baby, thank you for your coming. Without you and your mom, I do not think I could achieve this Ph.D. degree.

EFFECTIVE IDENTITY MANAGEMENT ON MOBILE DEVICES USING MULTI-SENSOR MEASUREMENTS

An Abstract of a Dissertation

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

By

Tao Feng

May 2015

Abstract

Due to the dramatic increase in popularity of mobile devices in the past decade, sensitive user information is stored and accessed on these devices every day. Securing sensitive data stored and accessed from mobile devices, makes user-identity management a problem of paramount importance. The tension between security and usability renders the task of user-identity verification on mobile devices challenging. Meanwhile, an appropriate identity management approach is missing since most existing technologies for user-identity verification are either one-shot user verification or only work in restricted controlled environments.

To solve the aforementioned problems, we investigated and sought approaches from the sensor data generated by human-mobile interactions. The data are collected from the onboard sensors, including voice data from microphone, acceleration data from accelerometer, angular acceleration data from gyroscope, magnetic force data from magnetometer, and multi-touch gesture input data from touchscreen. We studied the feasibility of extracting biometric and behaviour features from the onboard sensor data and how to efficiently employ the features extracted to perform user-identity verification on the smartphone device. Based on the experimental results of the single-sensor modalities, we further investigated how to integrate them with hardware such as fingerprint and Trust Zone to practically fulfill a usable identity management system for both local application and remote services control. User studies and on-device testing sessions were held for privacy and usability evaluation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Scope

Identity is a paramount product of human cognition that put forward the development of our kind in multiple domains at different stages. When it comes to the era of internet and mobile, identity management becomes one of the key problems that need to be properly solved to ensure the privacy and security of people's information and communication. According to the market analysis in [7], there will be 640 million tablets and 1.5 billion smartphones in use globally by 2015. This increasing ubiquity of mobile devices makes device access control and data security extremely important. Furthermore, more and more sensitive information such as transaction information for bank accounts, credit cards, trade secrets, etc., are passed through mobile digital devices. While these new uses introduce more convenience and a richer experience, they also create new privacy and security issues. In response,

these devices have now become targets for hackers. We can see these trends active in the market. From 2011 to 2012, mobile malware families ballooned by 58% [64], 32% of which were used to steal information [64]. In January 2012 alone, there were 32 million data breaches, of which 40% were caused by hackers [38].

However, the need for strong authentication on mobile is countered by the still-clumsy input method of such devices and the different user expectations for interaction models, especially when compared to the standard authentication solutions. As showed in a study of over 6,000,000 passwords, 91% of all user passwords belong to a list of just 1,000 common passwords [3] (e.g., 8.5% users use either "password" or "123456" as their passwords). Moreover, complex password of other knowledge-based authentication solutions will violate usability, which make these kind of solutions discarded by the mobile users.

Contemporary mobile user authentication technology has several limitations. For example, most devices only provide login screen password security which needs to be entered several times over the course of a day and cannot detect intrusions after the device has been unlocked. As a result, there is a compelling need for a complimentary on-demand authentication to ensure owner's identity after login stage. Explicit authentication mechanisms are not appropriate for continuous authentication scenarios because they would lead to usability issues. In [34], authors found that the most desirable user authentication method should be able to implicitly and continuously perform user-identification in the background without disrupting real-life user-device interactions.

On the other hand, in response to the aforementioned shockingly large numbers, there is a need to process sensitive information in a way that is independent of a potentially infected operating system while monitoring physical events of the device to detect possible physical unauthorized use. So the goal of our work is to investigate and design implicit and continuous authentication approaches, which employing the mature smartphone sensors to provide a higher security protection and user experience to the mobile system.

To identify the identity of a people, the mobile device generally can only work the same way as a human being: take advantage of the information collected from human-mobile interaction, to acquire the knowledge of ”Something you are?”, ”Something you have?”, and ”Something you know?”. For human being, we collect information from our eyes, our ears, and we can abstract and verify the knowledge with our brains, while for the mobile devices, they collect the information from the deployed sensors and analyzed our identity by their processors based on pre-defined policies. There are many different ways to approach an identity verification on the mobile device. We do can employ specific and sophisticated tokens to construct a robust identity verification based on the factor oh ”what you have?”. We can also formulate a complex alphabetic, graphic, or other memory based identity verification approaches to ensure the factor of ”what you know” will not be compromised. However, these approaches either request extra hardware, or extra efforts from the uses, which are not implicit and violate the usability of the identification system.

So the scope of our work is to use on-board sensors to sense user's information during the normal interactions, and analyze the user's "Something you are?" information to perform implicit user-identity management. Those approaches include methods that analyzing the inputs from the touchscreen, microphone, and motion-sensors. Furthermore, considering the requirement from usability aspect, we move forward the identity management problem with a new question, "have the user been changed?", and introduced an approach and framework integrated the single-sensor modalities with fingerprint sensor and novel security and crypto hardware, such as TrustZone, to increase the feasibility and performance of the implicit mobile identity management both in applications and remote services.

1.2 Research Questions

Nowadays, a smartphone device has a set of powerful sensors that can be used to collect users' input data during the interactions as well as context information. These specific data and context information raised a possibility to identify user's identity or provide clues to detect identity change in a more implicit and continuous way. To make this possibility become reality, several research questions must be answered to complete the design and implementation of a usable identity management framework.

- **Does the sensor data collected from user-device interactions reflect user's biometric and behavioral features?**

This question is a fundamental question to all the approaches proposed for employing the on-board sensor data to perform user-identity management, and a positive answer to this question builds up the basic hypothesis to the whole user-identity management framework. User-mobile device interactions will generate data from several sensors, the touchscreen, the microphone, and the motion-sensor. For the microphone, which is basically a speaker-recognition problem. The feasibility of this solution has been proved by a variety of researchers, publications, and realistic use cases. However, the touchscreen and motion-sensor based user-identity verification remains a relatively new domain and has not been fully investigated. To ensure the effectiveness of the sensor-data-based identity management framework, we have to first prove the feasibility of the touchscreen and motion-based identity verification approaches, or at least alternatively, showing these two sensor data could provide clues and context information on detecting user-identity change, which could potentially help improving the performance of the whole system.

- **How to employ the sensor data to achieve best trade-off between privacy protection improvement and usability enhancement in each modality?**

  Different from the previous question, which more relies on machine learning and pattern recognition techniques and approaches, this question trying to reflect the feasibility of the identity management system in a different aspect, which is usability. For other user authentication and recognition system, most likely the privacy protection improvement and usability enhancement

are equally important to the system. However, in an implicit and continuous user-identity management system, the usability would be a factor that plays a more prominent role. Since its continuous attribute, identity verification would be frequently performed during users' interactions. So even a 10% false reject rate would be a unbearable burden to the smartphone users. To make the system work, investigation on how to lower the false reject rate while keeping a reasonable false accept rate must be held. We will discuss how we achieve this goal in both controlled and uncontrolled touchscreen input data in Chapter 3.

- **How to employ the sensor data to perform usable user-identity management as a whole framework?**

  This question is a follow up question of the second question. What if the single modality's usability performance still cannot satisfy the requirement of an implicit and continuous user-identity management system even after employing a set of false reject rate reduction measures? The smartphone user may require a false reject rate that even below 1%, which is not achievable due to the nature of the interaction data. In this case, we may need to change the question from "Something the user are?" to "has the user changed?". We may also change the decision making approaches from "block any unauthorized users" to "require re-authentication when sensitive information, application, or action are accessed and performed". Another possibility is to combine the results from multiple sensors, even from very accurate sensors such as fingerprint, to make the whole system works. Generally speaking, investigation

on this question would provide an opportunity of making the system really usable. In Chapter 4, and Chapter 7, we will discuss the approaches on application management and integration, which investigated the answer to this question.

1.3 Research Contributions

Herein, we make several contributions to the implicit and continuous mobile identity management field, as summarized in the following:

- We introduced the concept of on-demand user-identity verification and the concept of device-leaving-hands event on the mobile system. Unlike the previous time-out based identity management schema, the proposed mechanism and algorithm leverages the sensor input from human-mobile interaction and evaluates the mobile user's identity change. We prove these concepts using quantitative approaches and experiments.

- We exploited the trade-off between security and usability in user-identity verification on smartphone. We considered usability as a paramount factor when analyzing the performance of a security system. We designed the identity management approaches based on the theory, "The most desirable user authentication method should be able to implicitly and continuously perform user-identification in the background without disrupting real-life natural user-device interactions.". We proposed and designed a set of algorithms

and policies to balance the trade-off between security and usability.

- We investigated the authentication approaches that employ biometric and behavioral features extracted from touch gestures to identify user's identity in an uncontrolled environment. We proposed and designed algorithms to solve the specific sequence data pattern matching problem. The algorithms include touch to image transformation, sliding window and threshold schema, and multi-level template database.

- We designed and conducted experiments to investigate the feasibility of employing the touchscreen usage data to perform user-identity verification. Over 200 subjects are invited for the data collection and on-device testing, and IRB of the data collection and user experiments is filled and updated. During the whole process, 140,000 touch gestures are collection from the touchscreen. I carried out my research and implementation of the touch module in three steps. We design and implement FAST, to evaluate the feasibility of touch-based identification with both touch and finger motion data collected from a designed digital glove. We then improve the touch-based identification by a novel graphic feature that can better differentiate phone owner from the others using touch gestures. After solving the above touch-based identity authentication problems in controlled environment, we move on and developed TIPS, a context-aware implicit user-identification system considering context applications and performs identity verification in an uncontrolled environment using One Nearest Neighbor and Dynamic Time Warping. While to our best knowledge, we are the first research team trying to solve the touch-based

user authentication in an uncontrolled environment.

- To investigate the feasibility of speaker-recognition on smartphone device and understanding its related usability issues, we designed and implemented a unified speech-speaker recognizer (USR) framework that provides specific response corresponding to different user-identity based on a customized identity management policy, which leveraging the speaker based identity verification. We collected over 40,000 speech commands are collected from microphone. To test the performance on real device, we developed an open-source Android library for speaker-recognition. Besides of the accuracy, we also carried out a comparison study of USR and the Google speech recognition framework.

- For the motion-sensors on the smartphone device, we proposed two verification methods, Statistical Method and Trajectory Reconstruction Method, to extract specific features and verify user's identity. We conduct experiments a multi-session MDP motion database. We also showed evidence that verification performance can be affected by the user body movements, such as walking. However, the investigation results on the motion-sensor based user-identification showed that it is not satisfactory as touch and speech, and later was served as a context information provider in the entire framework.

- To further take advantage of each single-sensor modality, we designed and implemented the IdentityTracker and the Secure Session Service. For the IdentityTracker, we designed an identity management scheme, a framework

that not only tracks the user's identity through the fingerprint sensors, it combines these results with data from both the motion-sensors and interaction inputs, including touchscreen usage and speech inputs, while concurrently managing app-level access on smartphones in post-login stages. To evaluate the performance of said system, we conducted two sessions of data collection and tested the trained model during the user's natural use. Results have shown that our approach improves user security by not granting app-level access to unauthorized guest users while at the same time promoting usability by greatly reducing the amount of unnecessary authentications for the smartphone's owner. For the Secure Session Service, we designed a framework for secure session-based applications, which leverages TrustZone and continuous biometric authentication schemes, IdentityTracker. The framework designed contributes to security heavily while not only maintaining, but improving device usability and convenience for the user. We implement identity verification schemes identifying its security properties and the results indicate our solution is practical.

1.4 Outline of the Thesis

We organize the remainder of this thesis as follows. In Chapter 2, we provide background knowledge and our approaches and briefly survey the related work and organize them with regard to our approaches. In Chapter 3, 4, and 5, we describe the approaches of employing single on-board sensor data, including the

touchscreen input, speech command, and motion-sensor data for the mobile identity management respectively. In Chapter 6, we introduced an approach that integrated fingerprint sensor and Trust Zone with the other sensor approaches to perform context aware user-identity management both local and remote. We conclude this thesis in Chapter 7.

Our published papers and their corresponding chapters:

- Chapter 3
  - **Tao Feng**, Ziyi Liu, Kyeong-An Kwon, Weidong Shi, Bogdan Carbunar, Yifei Jiang and Nhung Nguyen. "Continuous mobile authentication using touchscreen gestures". The 12th annual IEEE Conference on Technologies for Homeland Security (HST 2012). [25]
  - Xi Zhao, **Tao Feng** and Weidong Shi. "Continuous Mobile Authentication Using A Novel Graphic Touch Gesture Feature". The IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS 2013). [73]
  - **Tao Feng**, Jun Yang, Zhixian Yan, Emmanuel Munguia Tapia and Weidong Shi. "TIPS: Context-Aware Implicit User Identification using Touch Screen in Uncontrolled Environments". The 15th International Workshop on Mobile Computing Systems and Applications (HotMobile 2014). [26]
- Chapter 4
  - **Tao Feng**, Zhimin Gao, Dainis Boumber, Tzu-Hua Liu, Nicholas DeSalvo,

Xi Zhao, Weidong Shi, “USR: Enabling Identity Awareness and Usable App Access Control During Hand-free Mobile Interactions”, Sixth International Conference on Mobile Computing, Applications and Services (Mobicase 2014). [24]

- Chapter 5
  - **Tao Feng**, Xi Zhao and Weidong Shi. “Investigating Mobile Device Picking-up Motion as a Novel Biometric Modality”. The IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS 2013). [28]
- Chapter 6
  - **Tao Feng**, Xi Zhao, Nick DeSalvo, Zhimin Gao, Xi Wang and Weidong Shi, “Security after Login: Identity Change Detection on Smartphones Using Sensor Fusion”, 2015 IEEE International Symposium on Technologies for Homeland Security (HST 2015). [27]
  - **Tao Feng**, Nicholas DeSalvo, Lei Xu, Xi Zhao, Weidong Shi, “Secure Session on Mobile: An Exploration on Combining Biometric, TrustZone, and User Behavior”, Sixth International Conference on Mobile Computing, Applications and Services (Mobicase 2014). [23]

Chapter 2

Background and Related Works

Research of continuous and implicit mobile identity management draws from multiple areas. For the on-board sensor aspect, it includes implicit and continuous identity authentication and touch-gesture-based user recognition under controlled and uncontrolled environments, speech and speaker-recognition, motion-based identity and activity recognition, and identity management. On the other hand, the hardware requires background knowledge of fingerprint sensing as well as with trust zone.

2.1 Implicit and Continuous Identity Authentication

In general, there are three kinds of user authentication approaches: “Something you have”, “Something you know”, and “Something you are”. The approach of “Something you have” relies on a smartcard, a USB thumb drive, or some other

types of objects which users must have. Smartcards and USB drives must be physically inserted into the computer in order to authenticate the user. However, a smartphone itself can be considered as a token of "Something you have", and the challenges are associated with lost control of the smartphone token itself.

The arts of the approach "Something you are" can be categorized into two groups including implicit user-identification and multi-modality pattern classification, especially, multi-modality biometrics.

For desktops, researchers in the past have explored the feasibility of applying keystroke dynamics and typing patterns for user-identification. Keystrokes can be continually sampled by intercepting output from a keyboard. Ailisto et al. [51] used accelerometers in television remote controls to identify individuals. Cuntoor et al. [18] and Gafurov et al. [30] experimented user-identification using gait analysis and recognition. Koreman and Morris et al. [41] proposed a continuous multi-modal based approach for user-identification. In [34], Jakobsson et al proposed an implicit user authentication framework studied using recorded phone call history and location for continuous user authentication. Shi et al. described a multi-sensor based user authentication scheme that includes touchscreen as one of the continuous user-identification modality [63].

Some research efforts were conducted on graphical authentication method that uses the implicit drawing features to authenticate users. Jermyn, et al. [35] proposed a technique – "Draw a secret (DAS)". Users will draw a graph on a 2D-grid, and the information about which grid is occupied and in which order will be recorded. When trying to login, users will repeat the drawing. According to Jermyn

et al., a relatively small grid is in fact secure enough. But according to Thorpe et al.'s study [66], DAS's security perhaps is not so good as once believed. In the real world, people usually use signature to prove their identities. So it is natural that Syukri et al. [65] proposed a similar method in the cyber world. In their scheme, users need to draw their signatures with mouse, and system will normalize the data and record them into a database. During authentication, the system will extract the characteristics from the newly entered signature, and compare them with the pre-stored version. Furthermore, Varenhorst, et al. [67] proposed a method of drawing doodles rather than signatures. They used several methods to analyze the data, including grid, speed, doodle variance and a combination of all the above and achieved very high accuracy based on their evaluation.

There has been a body of literature on combining multiple biometric inputs to produce aggregated user-identification results. In [33], Indovina et al. identified that biometric integration can occur on the feature level, or at the score level. In feature level integration, all of the initial features from measurements are grouped together into a single feature vector for classification. Although the most information is available at this point, feature-level integration suffers from the so-called curse of dimensionality. Additionally, the features of some measurements may not always be available. Furthermore, several implicit identity sensing approaches have been proposed in the past that leverage the sensors on mobile devices such as accelerometer [51], GPS [52], touchscreen [60, 63], and microphone [47].

2.2 Touch-Based Identity Verification

2.2.1 Touch-Gesture-Based User Recognition under Controlled Environments.

Touchscreen gestures, as a normal and widely used user-device interaction method, have been recently used as a biometric modality for user-identity recognition and verification. Feng *et al.* [25] extracted finger motion speed and acceleration of touch gestures as features. Luca *et al.* [50] directly computed the distance between gesture traces using the dynamic time warping algorithm. Sae-Bae *et al.* [61] designed 22 special touch gestures for authentication, most of which involve all five fingers simultaneously. They computed dynamic time warping distance and Frechet distance between multi-touch traces. Frank *et al.* [29] studied the correlation between 22 analytic features from touch traces and classified these features using k-nearest-neighbors and Support Vector Machines. Shahzad *et al.* [62] proposed to use touchscreen gestures as a secure unlocking mechanism at the login screen. However, all prior works either require users to perform pre-defined touch gestures, or the data being collected under controlled experimental environments which might not be representative of natural user interactions. In our work [26], we explore implicit real-time user-identification from data collected under more natural uncontrolled environments.

2.2.2 Motion-Sensor Enhanced Touch Gesture User Recognition.

Bo *et al.* [12] presented SilentSense, a framework to authenticate users silently and transparently by exploiting the dynamics mined from users' touch behavior biometrics and the micro-movement of the device caused by users' screen-touch actions. Although implemented on the Android platform as a background service, SilentSense does not explore the data variations in uncontrolled environments. Furthermore, our TIPS approach can also leverage the application context to improve performance. We did not employ motion-sensor data when authenticating user-identity since it is power consuming and we wanted to focus on pure touch-screen based user recognition in our work.

2.3 Speech and Speaker Recognition

2.3.1 Speech Application

The Android speech recognition API allows developers to integrate speech recognition directly into their applications. Since its release, numerous applications have been developed that leverage speech recognition capabilities provided by the Android platform. These applications include the voice dialer, voice search, voice note, personal assistant (similar to Apple's Siri), voice navigator, voice controlled camera, voice commands, etc. Table 4.1 lists some smartphone applications that use speech recognition. Many speech recognition applications allow a user to interact with a mobile device hands free. They often provide a speech user interface

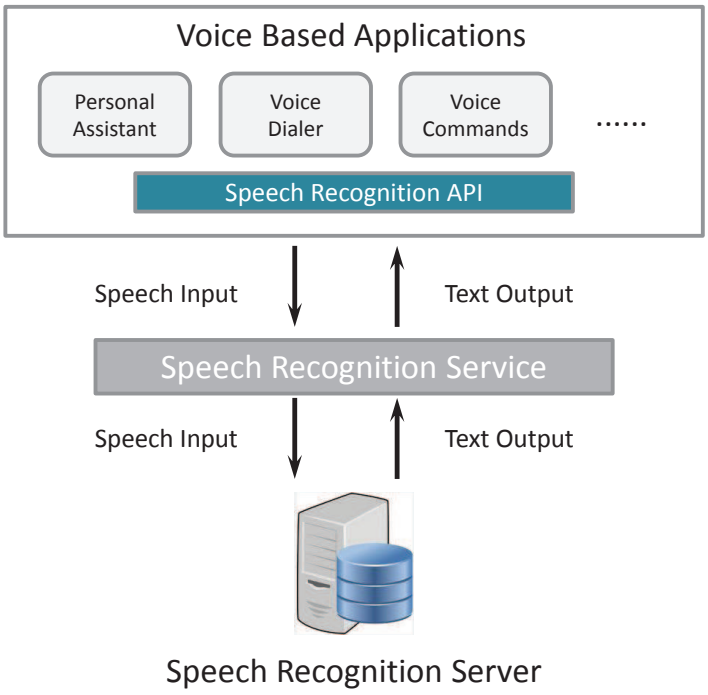


Figure 2.1: General mobile speech recognition framework

that supports features such as waking up on voice commands, automatically posting messages to Twitter and Facebook using speech-to-text, launching applications based on voice commands, opening calendar, searching based on voice inputs (*e.g.*, search contact list and automatically dial a person's number), even controlling mobile-device hardware, such as camera, using voices.

Though providing convenience to a mobile user, these speech-recognition based applications are potentially vulnerable to malicious exploits. Almost none of these applications we studied offers the capability to differentiate the speakers and enforces appropriate policies on who can interact and control a mobile device using speech. Things can get even worse when we dive deeper into the speech-based API and scrutinize its security. In order to support hands free interactions, activities triggered by speech can be launched while a mobile device is locked in a secure mode. This means voice-based actions can take precedence over the secure mode that requires a user to unlock a mobile device. By setting the "FLAG_SHOW_WHEN_LOCKED" flag, a user may bypass the lock screen and interact with the mobile device when the device is still in a secure mode. To give a concrete scenario, an imposter may post to a victim's Twitter or Facebook account using speech when the mobile device is in locked state.

2.3.2 Speech-Recognition Framework

Figure 2.1 shows a general framework for current mobile speech recognition. When user inputs a speech command to a speech application, the speech application records its voice and calls the system speech recognition API to activate a preset speech recognition service. The recorded *wave* file is then sent to the speech recognition server through the speech recognition service running in the background. After the speech recognition server transcribes the speech, it returns the text of the command to the speech recognition service and the speech applications, and then the speech application follows the text command.

From the general mobile speech-recognition framework, two weaknesses are obvious during the text command transfer from speech recognition service to system speech recognition API:

- The text command is transferred without any consideration on which application is calling the service and what is the feature of the application (*where usability can be promoted*).
- The text command is sent without any authentication process, which leaves a potential threat to the mobile system since the speech command will always follow the text command (*where privacy can be enhanced*).

2.4 Motion-Based Identity and Activity Recognition

Although physiological biometrics (*i.e.,* retinal patterns [44], fingerprint [56], facial features [74]) could provide stable and accurate verification rate, behavioral biometrics [69] have advantage over traditional biometric technologies due to their non-obtrusive.

Prior research on mobile motion is focused on two aspects, activity recognition and mobile motion-based user authentication. Some existing works have explored user activity inference methods with accelerometer sensors [13], [45], [11]. In [49], Lu *et. al.,* proposed a continuous sensing engine for activity recognition on mobile platform, which can robustly detect five common physical activities, stationary, walking, cycling, running, and in a vehicle (i.e., car, bus). Yang *et. al.,* [70] also researched on activity recognition by exploiting the accelerometer data. For mobile motion-based user authentication, [30] demonstrated an identity verification method using gait data. In [75], authors modeled gesture patterns through a continuous n-gram language model using a set of features constructed from mobile sensors, where picking-up a mobile device from table is considered as a task in the context sequence. Different from the aforementioned existing works, to our best understanding, we [28] consider the process of MDP motion as a biometric modality for user authentication.

2.5 Identity Management

Multi-user mobile and other devices have been researched as a new topic for recent years. In [38], Karlson *et al.* discussed the privacy and security issues when users of smartphone lend their phone to other physical users. Mobile permission management as a derivative research topic used to handle the privacy and security problems then attracts researchers' attention. Rofouei *et al.* [59] researched on multi-user device-display interaction identity identification by using a group of devices, including a Kinect camera, a multi-touch display and 2 accelerometer-equipped phones (one visible). [46] also present xShare, a protection solution to address privacy and security issues for the shared mobile scenario. However, previous work focusing only on how to utilize permission management to improve privacy and security, and ignores the potential usability promotion can achieved by identity awareness.

2.6 Fingerprint Sensing With Trust Zone

TrustZone has been gaining more and more attention from the smartphone security commmunity. Korean researchers have built a TrustZone-based platform for Android to prevent malware infections [10]. Also based upon TrustZone, Luo Jing and his colleagues designed a dual operating system, one satisfying users' application requirements and another acting as a secure OS providing specific security services [36]. Martin Pirker and Daniel Slamanig proposed a platform

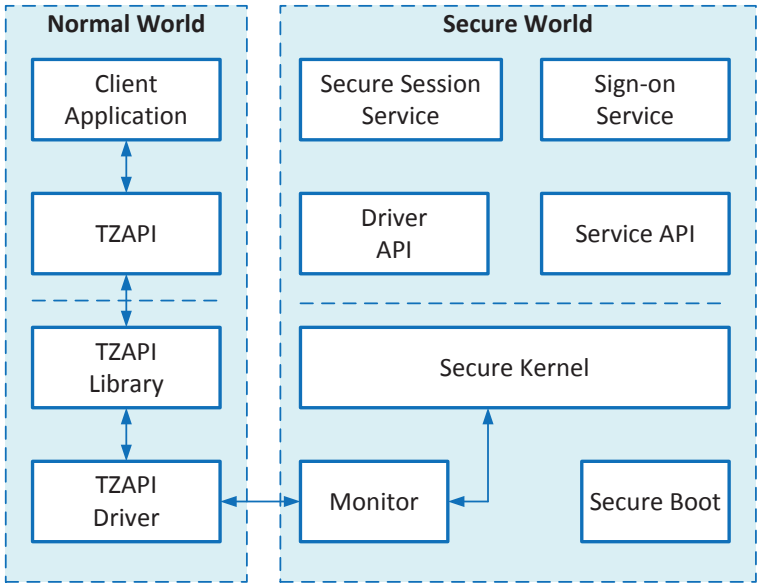


Figure 2.2: The general software layout of TrustZone. Note that there is the secure world and the normal world. Our approach places a transaction service within the secure world.

framework on TrustZone that may be used for arbitrary applications requiring a privacy-preserving online remote prepaid payment system suitable for micro as well as macro payments [55]. However, neither system provides a concrete design of user authentications on TrustZone. Researchers from ETH Zrich suggested utilizing TrustZone with a password to solve secure enrollment problem [53], but password authentication would be useless if the phone is lost and the password has been stolen.

Of the main components in fingerprint sensing with trust zone framework, as shown in Fig. 2.2, TrustZone is the most crucial, as it controls the secure processing of sensitive data. TrustZone is an extension to the SOC (system-on-a-chip) ARM design covering everything from the processor to the memory to the peripherals.

Using TrustZone design, the physical core is virtualized into two separate cores: one of which is referred to as the “secure world,” and the other one is the “normal world.” The secure world is used when processes will be interacting with or collecting sensitive data. This could range from using the keyboard, fingerprint readers, or extended to interactions with a bank app. In contrast, the normal world is used when there is no sensitive data being processed. This could include things such as playing games, taking pictures, etc. The apps that run in each respective world can be chosen and defined as per what is deemed sensitive. These two modes are completely isolated from each other to stop data leaks. Secure services that are run can range from a complete operating system to trivial services. TrustZone achieves this functionality through the novel addition of monitor mode. The monitor is used as a faucet of communication between the two worlds. Because the system is only as robust as the monitor, it must be sensitive to what is transferred between the worlds. The monitor also handles the context switches between the two modes which naturally gives it the responsibility of saving the state and switching safely between the two modes. The way that a program will enter secure mode is by throwing an exception. The exceptions that may trap to the monitor causing a world switch are FIQ, IRQ, and external aborts. This effectively allows sensitive processes to be run in this mode such that they are secure from any malware that could be present within the normal world. This is the main motivation of the TrustZone architecture.

The other critical component of this system, as seen in Fig. 2.3, is a biometric fingerprint sensor. As the main focus of this paper is not fingerprint sensor

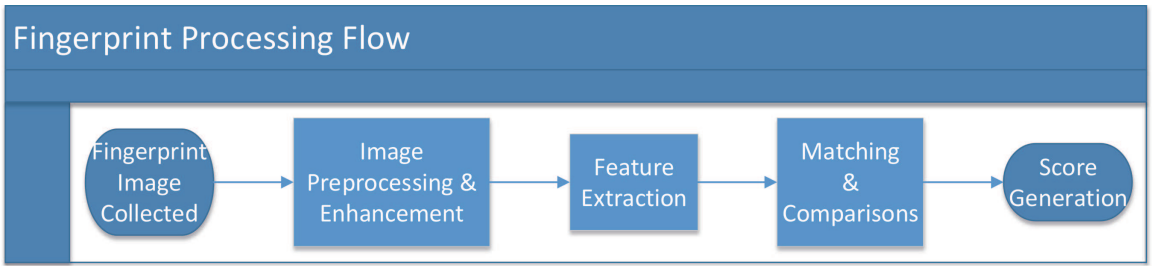


Figure 2.3: The general flow of processing a fingerprint

technology, this will not be addressed in depth. The type of fingerprint reader is irrelevant. The sensor must first generate an image of the fingerprint. This can be through either optical, capacitance, or ultrasonic means [54]. Optical sensors use visible light to capture an image. This has major weaknesses in that if the finger is dirty it can be difficult to retrieve a good image. Capacitive readers use capacitance in order to generate an image using an array of sensors to detect the fingerprint. Last but not least, ultrasonic readers use high frequency sound waves to generate an image. Regardless of the way that the image is retrieved, it is first prepossessed where the picture is enhanced and adapted so that feature extraction can generate more reliable identity traits from the user. After feature extraction, a feature vector can be obtained containing discriminative properties (such as ridge ending, bifurcation, and short ridges). The last step is to compare this vector with existing templates where a matching score is generated. If this score is above a threshold, the fingerprint is accepted as valid. If not, then it is rejected [54, 57].

Chapter 3

Touch-Gesture-Based Mobile User Authentication in Controlled and Uncontrolled Environment

Touchscreen gestures have recently gained popularity as a new ”biometric” signature for user authentication [25, 29, 61, 73]. This is because: (1) touch data is indicative of two biometric features, *i.e.,* the user hand geometry and muscle behavior. Such biometric characteristic variations have the potential to provide user discrimination; (2) touch data can be easily accessed with very low overhead on mobile devices nowadays.

We proceed in three steps to investigate on the touchscreen input based user-identity verification problem. For the first step, we designed and implemented FAST, to evaluate the feasibility of touch-based identification with both touchscreen

(a) Flick (b) Pinch (c) Spread (d) Drag (e) Rotate

Figure 3.1: Example Multi-touch Gestures

input data and finger motion data. Then we improved the touch-based identification solution with GTGF, a novel graphic feature that can better differentiate phone owner from the others. After solving the above touch-based identity authentication problems in a controlled environment, we move forward to solve the problem in an uncontrolled environment in the background. So we designed and developed TIPS, a context-aware implicit user-identification system considering application context.

3.1 Touch-Gesture-Based User Authentication Preliminary Study

Unlike PCs, touchscreen is the primary input medium on smartphones and tablets. Multi-touch inputs embed behavior characteristics that are user specific and can be used for detecting mobile users. We classify touch input into three categories: touch gestures (e.g., flick, spread, pinch, drag, and tap) see Figure 3.1; virtual typing (e.g, typing using a touchscreen based keyboard, entering a phone number using touch); and touch-based drawing (e.g., drawing shapes using fingers). For

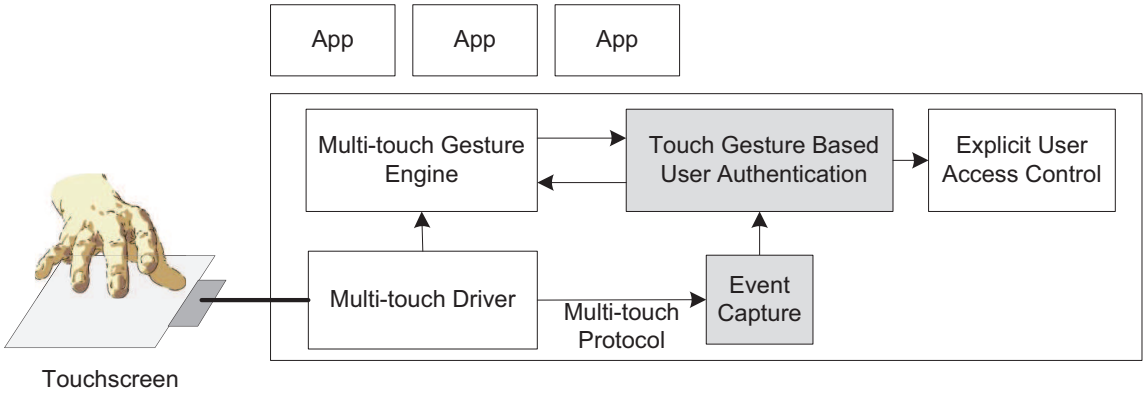


Figure 3.2: FAST Design

each category, user specific features can be extracted from traces collected from a device touchscreen.

We propose a touch-gesture-based user authentication system, FAST (Fingergestures Authentication System using Touchscreen), that focuses on post-login user authentication. Figure 3.2 shows a high-level diagram of the design. As long as the smartphone is used, FAST authenticates the user continuously. After user login, FAST continues to authenticate the mobile user in the background using intercepted touch data from normal user-smartphone interactions. To achieve the objective, FAST relies on gesture based smartphone owner detection. The detection approach is invoked on-demand whenever touch inputs are received and is transparent to the smartphone user. Only when there is sufficient evidence that the current user is not the smartphone owner, traditional user authentication is activated.

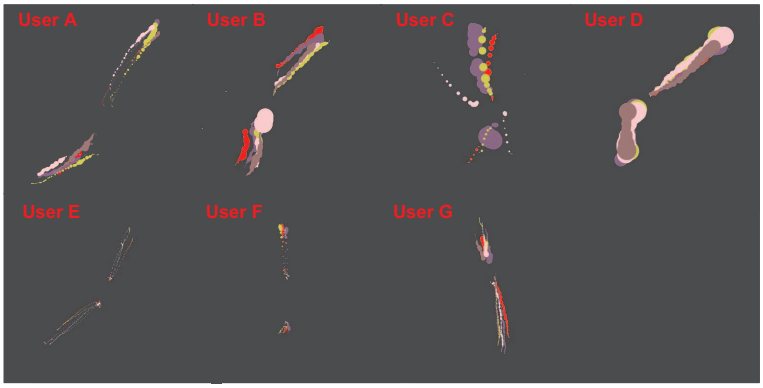
3.1.1 Touch Gestures

FAST collects selected touch gesture information including gesture type, X and Y coordinates, directions of the finger motion, finger motion speed, pressure at each sampled touch point and the distance between multi-touch points. In total, there are 53 features for each touch gesture. We consider only the six most frequent and useful gestures: down to up swipe, up to down swipe, left to right swipe, right to left swipe, zoom-in, and zoom-out. Since a smartphone user may apply different levels of touch pressure at different stages of a touch gesture FAST also divides each gesture into three segments, (i) the beginning of a touch motion, (ii) the main touch motion, which is the longest segment and (iii) the end of a touch motion.

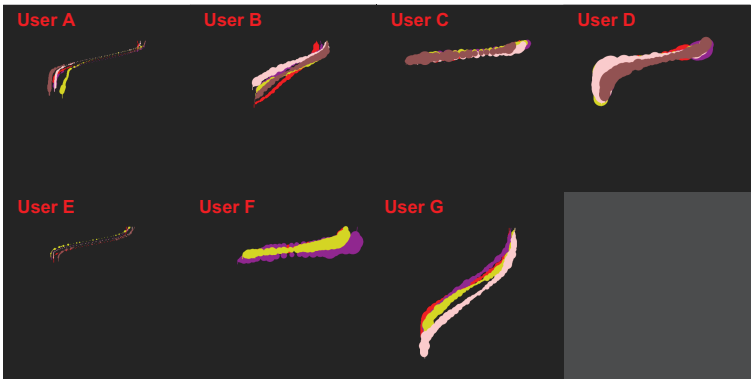
We have implemented an Android application that collects touch information from touchscreen-equipped smartphones. In a preliminary user study, we have collected the touch inputs of 7 users (three females and four males). Each user was asked to perform a set of touch related tasks, including controlling the smartphone UI using flick-touch gesture (left-to-right flick, right-to-left flick), mobile web browsing using pinch and spread-touch gestures, dragging icons and drawing simple shapes using finger touch. Each task was repeated multiple times by the same user.

Figure 3.3(a) shows sample spread-touch traces of the seven tested users. Each subfigure cell contains plotted traces of one user. In each subfigure, traces from different test trials are plotted using different colors. For each touch trace, the size

(a) Sample Spread Profiles of Seven Users



(b) Sample Pinch Profiles of Seven Users



(c) Sample Flick Profiles of Seven Users

Figure 3.3: Sample Multi-touch Traces from Users

of trace dots increases with the level of touch pressure.

Figure 3.3(b) contains pinch touch traces of the same seven tested users. Similar to the plotted spread traces, each subfigure cell shows plotted traces of one user. As indicated by Figure 3.3(a) and (b), each user has his/her own distinctive spread-touch style. No two of the seven users share the exact same spread-touch style. For the same user, there is a high degree of consistency that the same user exhibits similar spread and pinch touch style. Though collected from different trials, some of the spread-touch trace patterns of the same user match with one another almost perfectly.

Figure 3.3(c) shows sample flick touch traces of the seven tested users. Different from spread and pinch, a flick is a single finger gesture. Each subfigure cell contains plotted traces of one user. In each subfigure cell, the horizontal-axis denotes screen location translation and the vertical-axis denotes time. Each cell of figure 3.3(c) shows traces from different test trials using different colors. For each trace, the size of the trace dots increases with level of touch pressure. By observing the traces, one can find that for each trace, there was a finger acceleration stage, a steady movement stage (middle section of each flick trace), and a de-acceleration stage. FAST extracts steady touch pressure, minor/major ratio, steady finger moving speed, and acceleration/de-acceleration speed as features. The features are defined in Table 3.1.

Furthermore, FAST complements touchscreen gesture information with information collected from a digital sensor-glove. The glove provides X, Y, and Z axis angular rate information, the yaw, pitch and roll of finger movements, for a total of

Feature	Definition
Touch pressure	The touch pressure value of each touch point is acquired by mobile system API, and it is a mean value between several touch points.
Minor/major ratio	The angular information calculated by the x and y axis values of several consecutive touch points.
Finger moving speed	The speed is calculated by the distance divided by the time duration. The distance is the length from the start point to the end point, while the time duration is the difference of timestamp between the starting point and end point.
Acceleration/de-acceleration	The acceleration/de-acceleration is calculated by the speed difference divided by the time duration. The speed difference is the speed change between the start point speed and the end point speed, while the time duration is the difference of timestamp between the starting point and end point.
Mean of IMU sensor reading	The mean value of a single degree of sensor readings, and 9 features for 9 degrees.
Minimum of IMU sensor reading	The minimum value of a single degree of sensor readings, and 9 features for 9 degrees.
Maximum of IMU sensor reading	The maximum value of a single degree of sensor readings, and 9 features for 9 degrees.
Standard deviation of IMU sensor reading	The standard deviation of a single degree of sensor readings, and 9 features for 9 degrees.

Table 3.1: Features employed in FAST

36 additional features.We use these features to validate and complement features extracted from touch gesture for the user authentication process that occurs during normal smartphone interactions. Our intuition is that additional insight can be obtained by examining touchscreen traces and finger motion-sensor data together.

3.1.2 FAST: Putting It All Together

FAST collects, separates and stores the above three types of data, into two databases. One database is used for training classifiers and the other for testing the trained classifiers. Collected touch inputs are split between the two databases to avoid over-fitting.

FAST uses the several machine learning classifiers to classify a smartphone user based on her touch behavior. FAST uses the results of the classification to improve smartphone security in the following scenario. In the post-login stage, FAST extracts touch gesture and digital sensor-glove features and uses them to authenticate the user.

Care must be taken to achieve the proper balance of the FAR and FRR values. During the post-login stage, due to the constant user monitoring and frequent transparent authentication based on touch gestures and sensor-glove inputs, a low FRR is the primary objective – during normal user-smartphone interactions, usability is more important. This is because the frequency of the authentication operations ensures a rapid detection of intruders even for larger FAR values.

Touch-Sequence Length and Authentication Threshold. During the post-login phase, FAST continuously monitors the authenticity of the mobile user in a user transparent fashion. FAST achieves this by intercepting touch gestures inputs, and strives to achieve a low FRR. However, a user's touch gestures and corresponding sensor-glove inputs may vary over time. Thus, a user authentication solution that relies on just single input instances of touch gestures is unlikely to be reliable and accurate.

Instead, FAST adopts an aggregated authentication approach where results from a sequence of touch instances are combined. To control the quality of the aggregated user verification performance, FAST uses two metrics: the *Touch Sequence Length*(TSL), the length of touch input sequences and (ii) the *Authentication Threshold*(AT), for aggregating results. The AT metric is used to provide the lower bound on the touch sequence length: If the number of accepted touch inputs during one sequence is below the threshold, FAST considers that the current user is unauthorized and invokes an explicit authentication process.

3.1.3 The Equipment

To evaluate the ability of FAST to authenticate users, we have used the following equippment.

Sensor-glove. We have created a digital sensor-glove with IMU digital combo boards ITG3200/ADXL345. The glove provides 6 degrees of freedom and allows

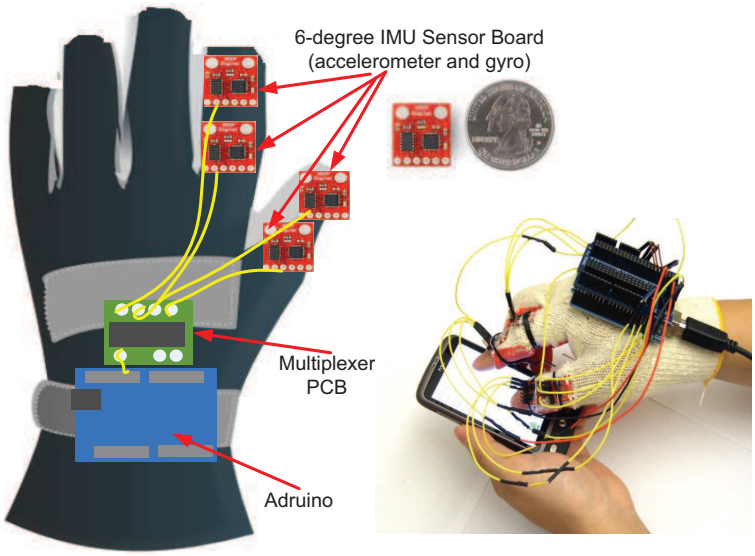


Figure 3.4: The Sensor-Glove with 6-degree IMU Boards

us to collect fine-grained biometric information of finger movements. This includes the three angle information: yaw, pitch, and roll, which is computed from the output of the three accelerometers on the digital combo board. The ITG-3200 is a single-chip, digital-output, 3-axis MEMS gyro IC. It outputs X-, Y-, and Z-Axis angular rates with a sensitivity of 14.375 LSBs per /sec and a full-scale range of 2000/sec. The ITG-3200 has three internal 16-bit analog-to-digital converters. The ADXL345 is a small 3-axis accelerometer with high resolution (13-bit) measurement at up to 16 g.

Smartphone. We used several HTC Android smartphones (Sensation model) for data collection. The model features a 4.3 inch capacitive S-LCD Gorilla glass touchscreen with qHD (540960) resolution at 256.15 PPI. We have developed an Android program for collecting touch gesture data from the HTC smartphones.

3.1.4 Glove Data Collection

We have divided the participants in a user study into two groups: the users in one group were equipped with a digital sensor-glove, the users in the other group were not. All the participating users were asked to perform smartphone functionalities using common touch gestures (*i.e.*, zoom-in, zoom-out, spread). The participant's touch gesture data were collected and stored.

40 subjects participated in the study. 11 users first joined the experiment with digital glove. However, for comparison, we have also collected their data without wearing the digital glove. Furthermore, because in the common case, people using smartphone were not wearing a digital glove, we collected the 40 subjects' touch gesture data without digital glove and stored in another database. The experiment was conducted at University of Houston. Participants were provided with a written consent form, including sections that describe the purpose of the study, its duration, the right to withdraw from participation and to refuse participation, the confidentiality of the information obtained and the use of research results. Participants were required to sign the form before participating in the experiment.

3.1.5 Results

3.1.5.1 Touch Gestures With Sensors

For multi-touch gestures involving two fingers, the set of touch-related features include, gesture types, sampled locations of the two fingers, directions of the touch

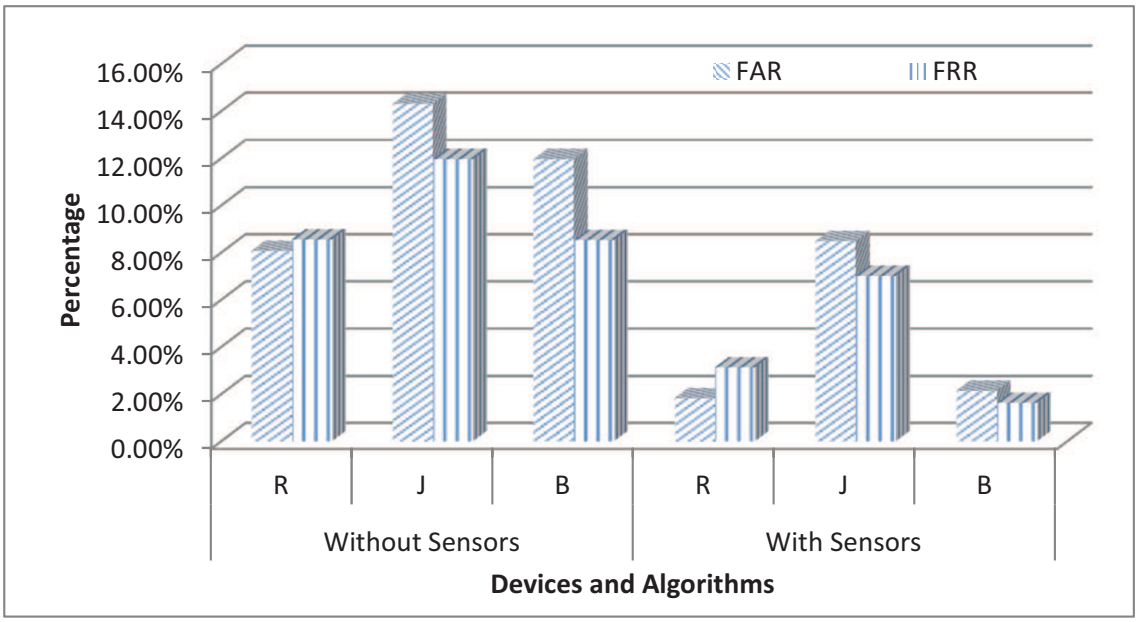


Figure 3.5: The Comparison of Data with Sensor Features and Data without Sensor Features

motion of the two fingers, time and pressure history of touch points, and the distances of the two touch points. Furthermore, with the help of the digital sensor-glove, some more user specified biometric features can be acquired and applied for user classification such as the X-, Y-, and Z-Axis angular rates of fingers when performing touch gesture inputs. We applied the three algorithms, Random Forest, J48 Decision Tree, and the Bayes Net on the collected touchscreen and sensor-glove data. The performance results achieved are shown in Figure 3.5.

As indicated by Figure 3.5, for both data with additional sensor-glove information and data without sensor-glove information, the Random Forest Classifier always outperforms the other two classification algorithms in terms of FAR value. However, the Bayes Net classifier always outperforms the other two in terms of the

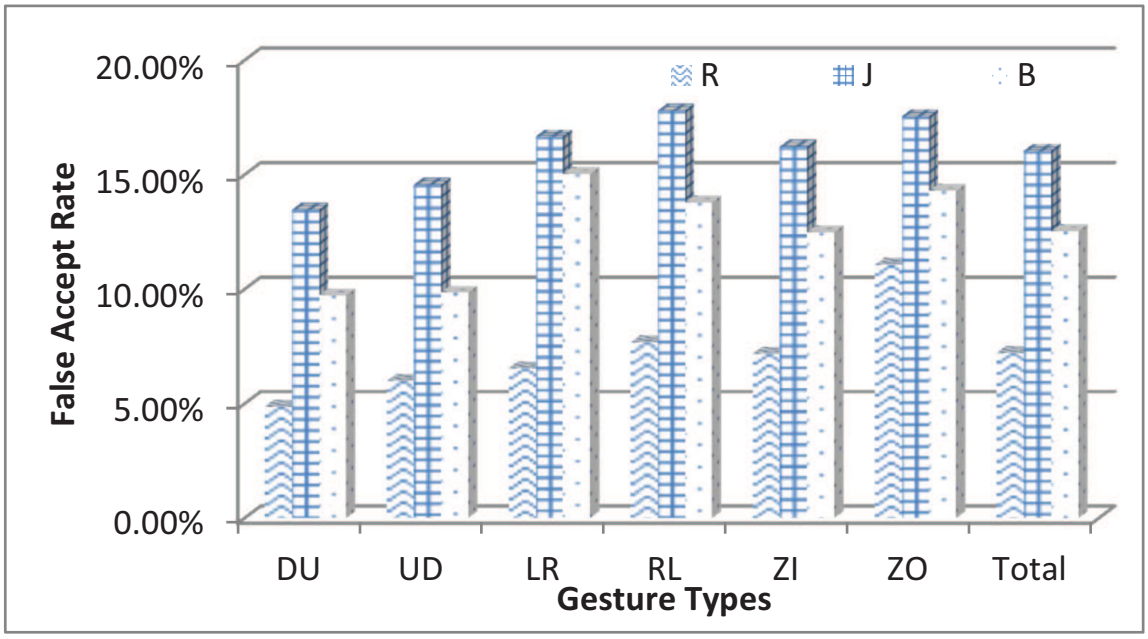
FRR metric. Since during the post-login stage, it is critical not to annoy the user and interrupt normal smartphone interaction with explicit access control activated by false rejection, we choose the Bayes Net classifier.

Furthermore, for all the three tested classifiers, the results achieved with the sensor-glove information significantly exceed the results achieved without it. FAST achieves a FAR of 11.96% and a FRR of 8.53% without external sensor information, when applying the Bayes Net classifier for single touch gestures. When additional sensor-glove information is present, FAST achieves a FAR of 2.15% and a FRR of 1.63%. This suggests that touch gestures of different people and smartphone touch gestures can be used as a source of information for user authentication. Furthermore, this also indicates that the biometric information acquired from the digital sensor-glove is helpful in authenticating the users when combined with the touchscreen inputs.

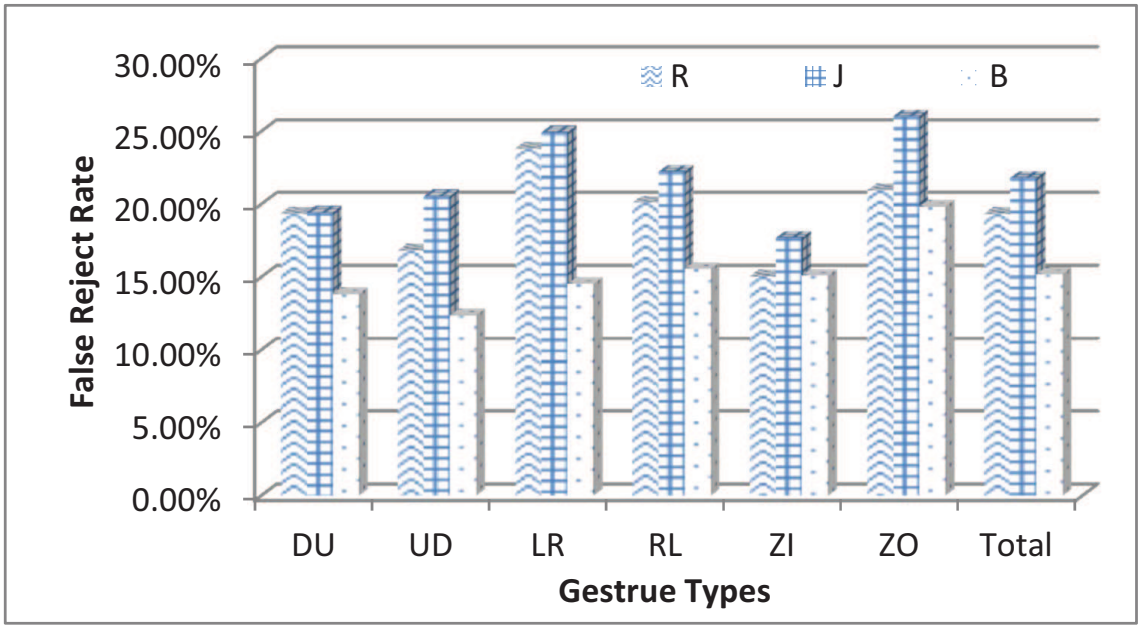
3.1.5.2 Touch Gestures Without Sensors

We further continued the user study using the touch gesture data of the 40 participants when no additional sensor-glove information is available. We performed this in order to simulate the normal user-to-smartphone interaction conditions.

We applied the same three algorithms, Random Forest, J48, and Bayes Net as the classifiers. The results are shown in Figure 3.6. R, J, and B respectively stand for Random Forest, J48 Decision Tree, and Bayes Net. The data sets are divided according to the gesture types: DU, UD, LR, Rl, ZI, ZO and Total respectively stand



(a) FAR (False Accept Rate)



(b) FRR (False Reject Rate)

Figure 3.6: The FAR and FRR of Different Algorithms and Gesture Types

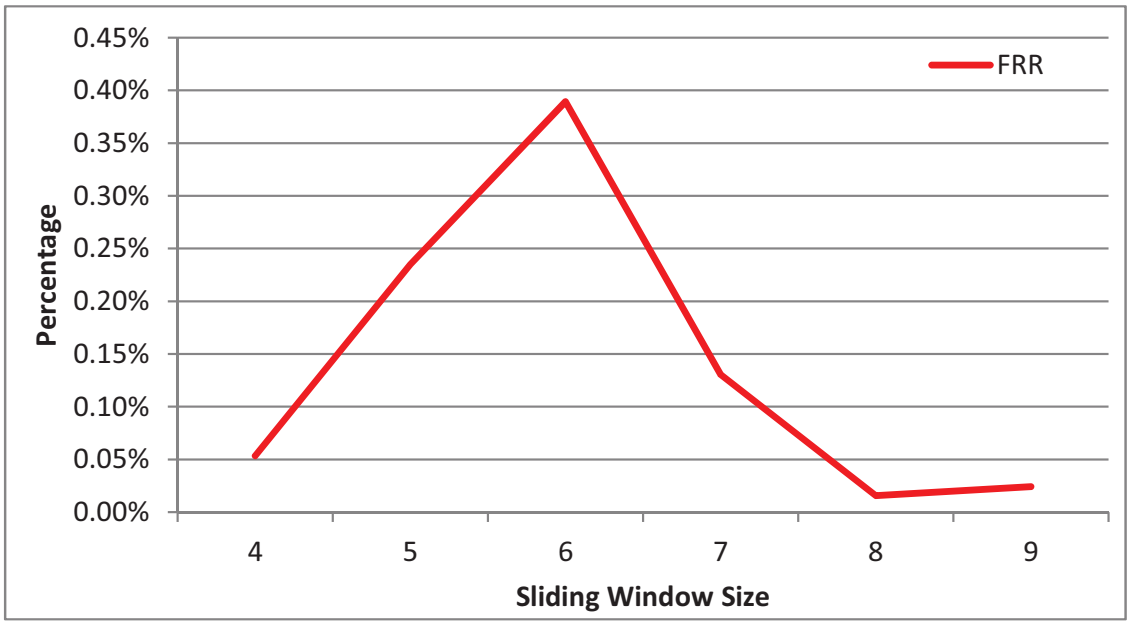
for, swipe from down to up (DU), swipe from up to down (UD), swipe from left to right (LR), swipe from right to left (RL), zoom-in (ZI), zoom-out (ZO) and the overall performance of all the combined gesture types (Total).

Figure 3.6 shows that the Random Forest classifier always performs better than the other two classifiers in terms of FAR. However, in terms of FRR, it performs worse than the Bayes Net. Although FAST can achieve, on average, a 14.02% FAR and a 18.92% FRR using the Bayes Net classifier using limited data provided by a single touch gesture, it is still not good enough for meeting the design requirement of low FRR. Consequently, we proposed a sequence-based authenticate mechanism. It is described below.

Gesture Sequence Based Authentication. Figure 3.7 shows the FAR and FRR values achieved by FAST as a function of the Touch Sequence Length (TSL) metric. The x-axis shows the TSL value of an authentication cycle and the y-axis shows the best FAR and FRR values that can be achieved under the TSL. It shows that the best FAR/FRR combination is achieved when the TSL is 7. Thus, we set TSL to 7.

Furthermore, Figure 3.8 shows the FAR and FRR values under an AT of 2 (FAR=21.54% and FRR=0.01%) and an AT of 3 (FAR=4.66% and FRR=0.13%). Thus, both values are applicable for authentication purposes. Since the FAR of AT=3 is significantly smaller than for AT=2, we choose AT=3. This means that for every 7 valid touch gestures, if 3 or more gesture inputs are recognized as inputs from the authorized user, then this input sequence is accepted as being valid – the user is authenticated. Otherwise, the input sequence is considered as an unauthorized sequence.

(a) FAR (False Accept Rate)



(b) FRR (False Reject Rate)

Figure 3.7: The FAR and FRR under different sequence length values.

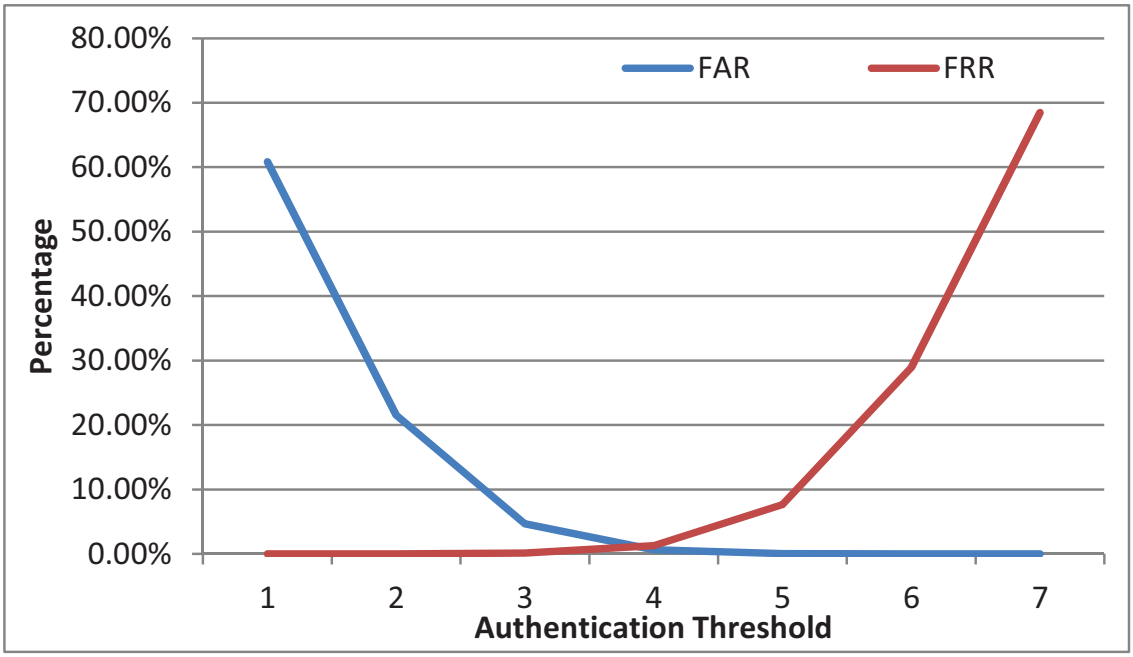


Figure 3.8: FAR and FRR of different sequence-based threshold values.

An FRR of 0.13% is equivalent to one wrong user logout every 800 touches or about 1 hour of continuous system use. A FAR of 4.66% means that after 3 attempts, an unauthorized user will be still authorized with a probability of 0.01%. Thus, FAST's gesture sequence-based authentication mechanism provides strong post-login security protection while significantly reducing user interruptions.

Furthermore, FAST uses a time threshold of sixty seconds to limit the valid time window size of an unfinished gesture input sequence. This means that if a gesture sequence is incomplete and there are no more gesture inputs for more than sixty seconds, a new sequence will be created upon receipt of the next touch gesture input with the unaccepted touch number of the previously incomplete sequence.

This time threshold is set to protect the system in the case where an attacker continues to use the device left off by an authorized user who has already completed several gesture inputs.

3.2 Touch-Gesture-Based User Authentication Using A Novel Graphic Feature

3.2.1 Graphic Touch Gesture Feature

A touch trace is a series of x-y coordinates of finger touch points with pressure values and time stamp. From each series, we can extract the time duration, the length of touch traces, the directions and speeds of finger movements, and the tactile pressures. All these extracted features imply the user's hand geometry and muscle behavior. In this section, we introduce a novel feature to convert a touch trace into an image, where all aforementioned features can be represented in an intuitive and explicit manner.

The first step is to extract the touch traces from the touchscreen outputs. All captured points with their pressure over a threshold are read. Thus, a single fingertip touch trace is encoded as a series of N point samples $S_n = (x_n, y_n, t_n, p_n), n \in 1, 2, ..., N$, where x_n, y_n is the touch coordinate, t_n is the time stamp and the pressure p_n. Multiple fingertip gestures (*i.e.,* Pinch, Spread) include multiple series of

Figure 3.9: Examples of the GTGF extraction. The first row includes features of five gesture traces from a single subject; and the second row includes features of five traces from another subject. The movement and pressure dynamics can be explicitly represented in GTGF. We can observe major variations on the image pattern between two rows but minor variations within one rows.

Table 3.2: Gesture descriptions

Annotation	Gesture	Fingertips	Touch type
DU	Slide up	Thumb or index finger	Single
UD	Slide down	Thumb or index finger	Single
LR	Flick right	Thumb or index finger	Single
RL	Flick left	Thumb or index finger	Single
ZI	Pinch	Thumb and index finger	Multiple
ZO	Spread	Thumb and index finger	Multiple

samples. Extracted traces are further filtered into one of the six predefined gestures, as described in Tab. 3.2. The filtering is based on screen regions where the traces start and end. In case of multiple fingertip gestures, the Euclidean distances between two fingers at the start and end of traces are also used.

In order to normalize and register the traces with different number of points and time intervals, we use cubic interpolation, as shown in Eq.3.1, to resample the traces in terms of their x-y coordinate, time and pressure series so that all traces have a fixed number of sampled points (*e.g.,* 50).

$$\begin{aligned}\mathfrak{F}(f_o, f_1, f_2, f_3, t) &= (-\frac{1}{2}f_0 + \frac{3}{2}f_1 - \frac{3}{2}f_2 + \frac{1}{2}f_3)t^3 \\ &+(f_0 - \frac{5}{2}f_1 + 2f_2 - \frac{1}{2}f_3)t^2 \\ &+(-\frac{1}{2}f_0 + \frac{1}{2}f_2)t + f_1 \end{aligned} \tag{3.1}$$

where t represents timestamp where to interpolate, f_n represents the known samples at predefined time, $\mathfrak{F}$ is the cubic interpolation function and outputs the interpolated value f_t (*i.e.,* x, y, p) at timestamp t. Cubic interpolation is chosen because it is the simplest method that offers true continuity between the samples. After normalization, a single touchtip trace $\hat{S}$ (*i.e.,* UD, DU, LR, RL) is obtained consisting of 50 samples or a multiple touchtip trace (*i.e.,* ZI, ZO) is obtained consisting of 100 samples, 50 samples of $\hat{S}$ and 50 samples of $\hat{S}'$. Each sample includes a pair of x-y coordinates $\hat{x}, \hat{y}$, a pressure value $\hat{p}$ and a timestamp $\hat{t}$. Note that the time intervals between samples may vary since traces may have different time duration but the same number of samples.

$$\begin{aligned}\hat{S}_n &= (\hat{x}_n, \hat{y}_n, \hat{t}_n, \hat{p}_n), n \in 1, 2, ..., 50 \\ \hat{S'}_n &= (\hat{x'}_n, \hat{y'}_n, \hat{t}_n, \hat{p'}_n), n \in 1, 2, ..., 50 \end{aligned} \tag{3.2}$$

After obtaining the $\hat{S}$, we further convert the normalized traces into GTGF $\boldsymbol{T}$. This conversion needs to be conducted in a way that the discriminative power of the original traces are preserved while the graphic features are easy to compute for a mobile based platform. Thus, we create a zero-valued image template $\boldsymbol{\mathcal{T}}$ with

resolution set to 100×150. This size is a tradeoff between the feature 'discriminability' and computational efficiency.

For each sample $\hat{S}_n$, we use a block $\boldsymbol{C_n}$ which has a width of three columns to represent its information. The block is evenly divided into upper block $\boldsymbol{C_n^p}$ and lower block $\boldsymbol{C_n^d}$. In general, the upper subblock describes the x direction related features exclusively, and the lower subblock describes the y direction related features exclusively. The height H_n^p and the intensity of the upper and lower subblock I_n^p, I_n^d are the three important properties which are used to represent the tactile pressure and the movement dynamics along x and y axes at the timestamp $\hat{t}_n$. They are computed as:

$$I_n^p = \lceil I_m * \frac{U_x - \Delta\hat{x}_n}{U_x} \rceil \tag{3.3}$$

$$\Delta\hat{x}_n = \hat{x}_{n+1} - \hat{x}_n$$

$$I_n^b = \lceil I_m * \frac{U_y - \Delta\hat{y}_n}{U_y} \rceil \tag{3.4}$$

$$\Delta\hat{y}_n = \hat{y}_{n+1} - \hat{y}_n$$

$$H_n^p = \lceil H_c * \frac{\hat{p}_n}{L_p} \rceil \tag{3.5}$$

where $\lceil \rceil$ is the ceiling function fetching the nearest greater integer. $H_c = 50$, which is half of the preset image height, $I_m = 128$ is chosen because it evenly divides the intensity space [0, 256] so that the sign and the absolute value of Δx_n and Δy_n can be represented by intensity values. The Δx_n values below 128 imply the hand moves to the right, and the values above 128 imply the hand moves to the left. The Δy_n values below 128 imply the hand moves up, and the values above 128 imply

the hand moves down. The greater $abs(I_n^p - 128)$ or $abs(I_n^b - 128)$ is, the faster the fingertip moves. Meanwhile, the greater H_n^p is, the harder the finger touches the screen. We repeat Eq. 3.3,3.4,3.5 for all N samples on a trace so that all the block C_n are computed. For multiple touchtip gestures, we create $\boldsymbol{T}$ and $\boldsymbol{T'}$ for S and S' respectively. In this way, the direction, the pressure and the dynamics of the traces are directly encoded into the GTGF, as depicted in Fig. 3.9. The images within the same row are quite similar with each other. But major differences can be observed between two rows. The differences stems from the different user identities.

The extraction of GTGF has multiply benefits. First, original traces have different spacial topology and temporal duration. This causes difficulties in registering the traces. The proposed GTGF solves this difficulty via resampling the traces and fitting them into the graphic template $\boldsymbol{\mathcal{T}}$. Second, the dynamics is considered to be an important factors in other pattern recognition problems, *i.e.,* facial expression recognition [72] and speech recognition [31]. However, they are not commonly considered in the touch-gesture-based authentication literature. The proposed GTGF is able to represent the gesture dynamics in terms of movement and pressure intuitively and explicitly. Third, due to the inhomogeneity between location and pressure data, it is difficult to combine their discriminative power in the feature level. However, extraction of GTGF takes both features into consideration and fuse their discriminative power together.

3.2.2 Score metrics and normalization

We use image processing techniques to compute the score between two images. Since all the GTGF have the same dimension (the size of $\boldsymbol{\mathcal{T}}$), the score metrics can be computed directly on the images without extra processing and registration steps.

The first score metric we use is based on the normalized cross correlation:

$$NC(a,b) = \left\langle \frac{\boldsymbol{T}_a}{\|\boldsymbol{T}_a\|}, \frac{\boldsymbol{T}_b}{\|\boldsymbol{T}_b\|} \right\rangle \tag{3.6}$$

where $\langle \cdot, \cdot \rangle$ is the inner product and $\|\cdot\|$ is the L_2 norm. The normalized cross correlation described the similarity between two GTGF $\boldsymbol{T_a}$ and $\boldsymbol{T_b}$. We get the distance in the verification via $D_n(a,b) = 1 - NC(a,b)$.

The second score metric is L_1 distance:

$$L_1(a,b) = \sum_{l \in \mathcal{T}} |\boldsymbol{T}_a - \boldsymbol{T}_b| \tag{3.7}$$

where l is the index of image pixels and $\mathcal{T}$ is the image template with the same size as $\boldsymbol{T}_a$ and $\boldsymbol{T}_b$.

The third score metrics is L_2 distance:

$$L_2(a,b) = \sqrt{\sum_{l \in \mathcal{T}} (\boldsymbol{T}_a - \boldsymbol{T}_b)^2} \tag{3.8}$$

For touch gestures with multiple traces, scores from T, T' are summed to a single score for authentication.

Each query gesture has a set of scores corresponding to query-target pair comparison. However, different queries may have different distributions of their score

set. This affects the verification performance when setting the canonical threshold for verification. Thus, we use the Z score normalization [14] to normalize their score distributions into zero mean and unit variance. Their mean μ and the standard deviation σ need to be pre-computed, D is a score value and Z is the normalized score:

$$Z = \frac{D - \mu}{\sigma} \tag{3.9}$$

3.2.3 Data Acquisition

To collect user-touch gesture data, we developed an Android program which captures touch gestures using a standard API of Android system. When fingers contacted the touchscreen, it started to record the trace by recording raw touch samples from the API. For each sample in a trace, an event flag (*e.g.,* onDown, OnScroll, onFling, Zoom), the absolute event time in *ms*, and the (x,y) coordinates and the tactile pressure per contacted finger are captured.

We recruited 30 subjects for our study, of which 28 were right-handed, 24 had touchscreen device experience. The data acquisition contains 6 sessions collected over several weeks. In the first session for each subject, we explained the purpose of the study and the usage of our data acquisition program. Then, the subjects were left for 5 to 10 minutes to practice. After he/she can use the program in a natural way, the subject provided 20 traces for each gesture in Tab. 3.2. Then the following 5 sessions occurred with at least three days in-between two consecutive

sessions. Subjects provided at least 20 traces per gesture during each session. The touch gestures are performed in the way that the subjects feel natural and comfort. Thus, subjects may vary in the manner they hold the phone (*i.e.,* left-hand holding vs right-hand holding, whole palm holding vs half palm holding), the hand pose (palm down vs palm up) or the fingers used to perform the single touch gesture (thumb vs index finger) or the finger orientation between sessions. We name the dataset collected in the first session as XTOUCHv1 and the dataset collected in the following sessions as XTOUCHv2. In our study, the XTOUCHv1 is mainly utilized for the feature evaluation and authentication system design. The XTOUCHv2 is mainly utilized in the research on inter-session authentication and continuous authentication.

Equal Error Rate was mostly used as the evaluation criterion for our verification experiments. It is the common value when the false acceptance rate (FAR) and the false rejection rate (FRR) are equal. We opt to use it simply because it accounts for the trade-off between FAR and FRR. Meanwhile, we also reported ROC curves in some experiments for clarity purpose.

3.2.4 Experimental Results

Figure 3.10 depicts the results of grid search on the three dimensional parameter space, including the relative movement upper bound U_x, U_y in the x, y directions and the upper bound for the tactile pressure L_p. They vary in the range from 0.3 to 0.7, from 20 to 60 and from 30 to 70 on the L_p, U_x, U_y axes, respectively. For

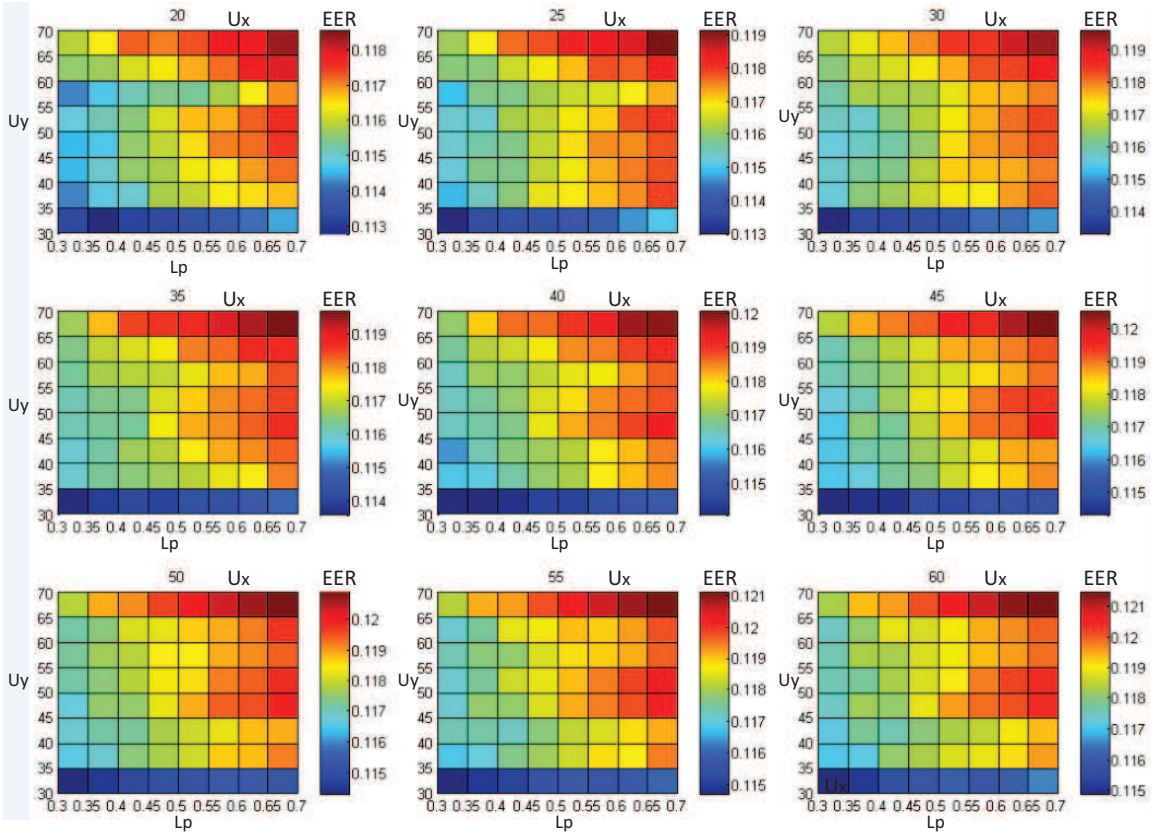


Figure 3.10: Results of grid search on the parameter space (L_p, U_x, U_y) on the evaluation dataset XTOUCHv1, where L_p denotes pressure parameter, U_x denotes speed parameter on x-axis and U_y denotes speed parameter on y-axis. Each subfigure is a slice on the U_x axis in the 3-dimensional parameter space, and the x, y axes in all subfigures represent L_p and U_y axes in the parameter space. The color represents the average Equal Error Rate for the six gestures under a set of parameter. The color of the figure presents the value of the Equal Error Rate. The lighter the color, the higher the EER. Note that the same color in different subfigure may represent different EER values according to the colormaps accompanied beside.

%	**NS**	**L1**	**L2**	Mean
DU	14.18	8.83	9.66	10.89
UD	16.46	7.07	7.98	10.50
LR	16.82	12.02	12.61	13.82
RL	16.83	11.86	12.05	13.58
ZI	19.31	14.12	15.40	16.28
ZO	20.16	13.78	16.55	16.83
Mean	17.29	11.28	12.38	

Table 3.3: Score metric comparison on user verification scheme. The Equal Error Rate is presented in each blank as a criterion for user verification. **NS** stands for the normalized cross correlation, **L1** stands for L_1 distance, and **L2** for L_2 distance.

the experiment setup, we randomly selected half portion of the traces per subject per gesture from XTOUCHv1 dataset as gallery (target) set and used the other half portion per subject per gesture from XTOUCHv1 dataset as probe (query) set. The experiments were conducted six times for all gestures and ERR reported was computed by averaging the ERRs from all the gestures. L_1 distance was used as the distance metrics. The variations on EER is from 11.28% to 12.14% and the best EER has been achieved at $0.35, 20, 30$ in parameter space. This demonstrated the proposed GTGF feature is not sensitive to the parameter variations in the aforementioned tested range. Meanwhile, the parameters at which the best ERR is achieved are adopted in the following experiments.

We evaluated the performance of GTGF with different score metrics, including the L_1 distance, L_2 distance and the normalized cross correlation. We used the same gallery and probe sets as the previous tests. Table 2 depicts the EER of authentication using each gesture respectively and compares the different distance

%	DU	UD	LR	RL	ZI	ZO	Mean
a	8.83	7.07	12.02	11.86	14.12	13.78	11.28
b	12.70	12.29	21.10	20.24	20.42	24.88	18.61
c	12.50	9.86	14.47	14.70	16.97	17.11	14.27

Table 3.4: Score metric comparison on user verification scheme. Index a, b, c corresponds to three cases in Fig.3.11.

metrics. The gestures DU and UD achieve average EER below 11% but ZI and ZO achieve average EER above 16%. For different distance metrics, the mean EERs achieved by the L_1 distance, the L_2 distance and the normalized correlation are 11.28%, 12.38% and 17.29%. In general L_1 distance performs better than the two distance metrics. This implies that DU and UD gestures could have more discrimative power than the other touch gestures and the L_1 distance is more discriminable than the other two metrics among subjects. In the following tests, we opt to adopt the L_1 distance as the distance metrics for our authentication scheme.

In order to evaluate the contribution of movement and pressure dynamics for the feature's discrimative power, we 'mute' them respectively and conducted the following tests on the XTOUCHv1 dataset. The gallery set and probe set were the same as the previous tests. In the first test, we used the normal GTGF feature, as in Fig. 3.11a. In the second test, we set the outer edge to be constant, as in Fig. 3.11b. Only the intensity values, which represent movement dynamics, were used for authentication. Then, in the third experiment, we set all image intensity value to 128, as in Fig. 3.11c, so that only the outer edge, which represents tactile pressure dynamics, were used for authentication. Results in Tab. 3 demonstrates

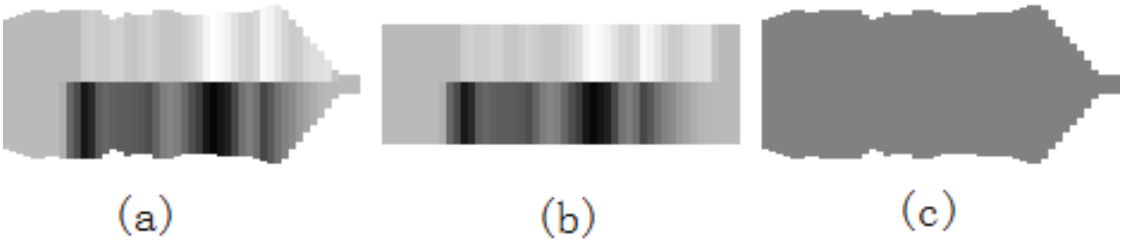


Figure 3.11: Examples on extraction of GTGF. (a): The original GTGF; (b): The pressure-muted GTGF; (c): The movement-muted GTGF.

that with only movement dynamics (row b), EER increased from 11.28 to 18.61 (over 64.9%); in case using only pressure dynamics (row c) EER increased from 11.28 to 14.27 (over 26.5%). Note that the poorer the authentication performs, the more EER increases. Thus, the pressure dynamics has more distrinictive powever compared to the movement dynamics in the touch-gesture-based authentication.

To explore the variance on the temporal factor of the authentication, we tested our proposed method across five sessions in the XTOUCHv2 dataset. For each gesture, we setup up five gallery sets and five probe sets, where gallery and probe sets were obtained via randomly and evenly dividing the touch data in a session. Then, we applied our proposed method on 25 combinations of gallery and probe pairs for each gesture and depicted the results in Fig. 6. The six gestures correspond to the six subplots respectively. First, EER of intra-session tests (bars on the diagonal) are generally lower than the other inter-session tests (bars on the other positions). This is consistent with the result in [29], the touch gestures from a user followed more similar patterns within a session than those in mixed sessions. Second, a longer time interval between the gallery and probe sessions does not necessarily result in

a higher EER. Some intersession EERs tested on the first session gallery and the fifth session probe are lower than some of other tests using the gallery and probe sets with less time intervals. Thus, temporal interval between gallery and probe is not the reason to the decreased intersession authentication performance. More probably it is because the change of the manner users perform the touch gesture, (*i.e.,* left or right hands, hand poses, finger orientations). Third, the EERs of tests on multiple touchtip gestures (*i.e.,* ZI and ZO) have less variations compared to the tests of the single touchtip gestures (*i.e.,* DU, UD, LR, RL). When users perform the multiple touchtip gestures, there are less variations on the manner how users perform them. Since both hands and both thumb and index finger are involved in these relative complex touch gestures, variations on the hand pose and used fingertips are reduced. This results support our analysis that the manner in which the users perform the touch gesture could probably be the reason for the decrease of intersession authentication performance.

To assess the fusion strategy of gestures for continuous authentication, we tested it on the whole XTOUCHv2. The gallery set per gesture is consisted of a combination of the aforementioned five gallery sets, and the probe set per gesture is a combination of the aforementioned five probe sets. After score computation and Z-normalization, we can obtain one score matrix for each gesture. Then we fused the multiple gestures using the sum rule, which is proved by Kitller *et. al.,* [39] to be superior in comparison of other rules, *i.e.,* product, min, max, median rule. Figure 3.13 depicts a comparison of the Receiver Operating Characteristic (ROC) curves

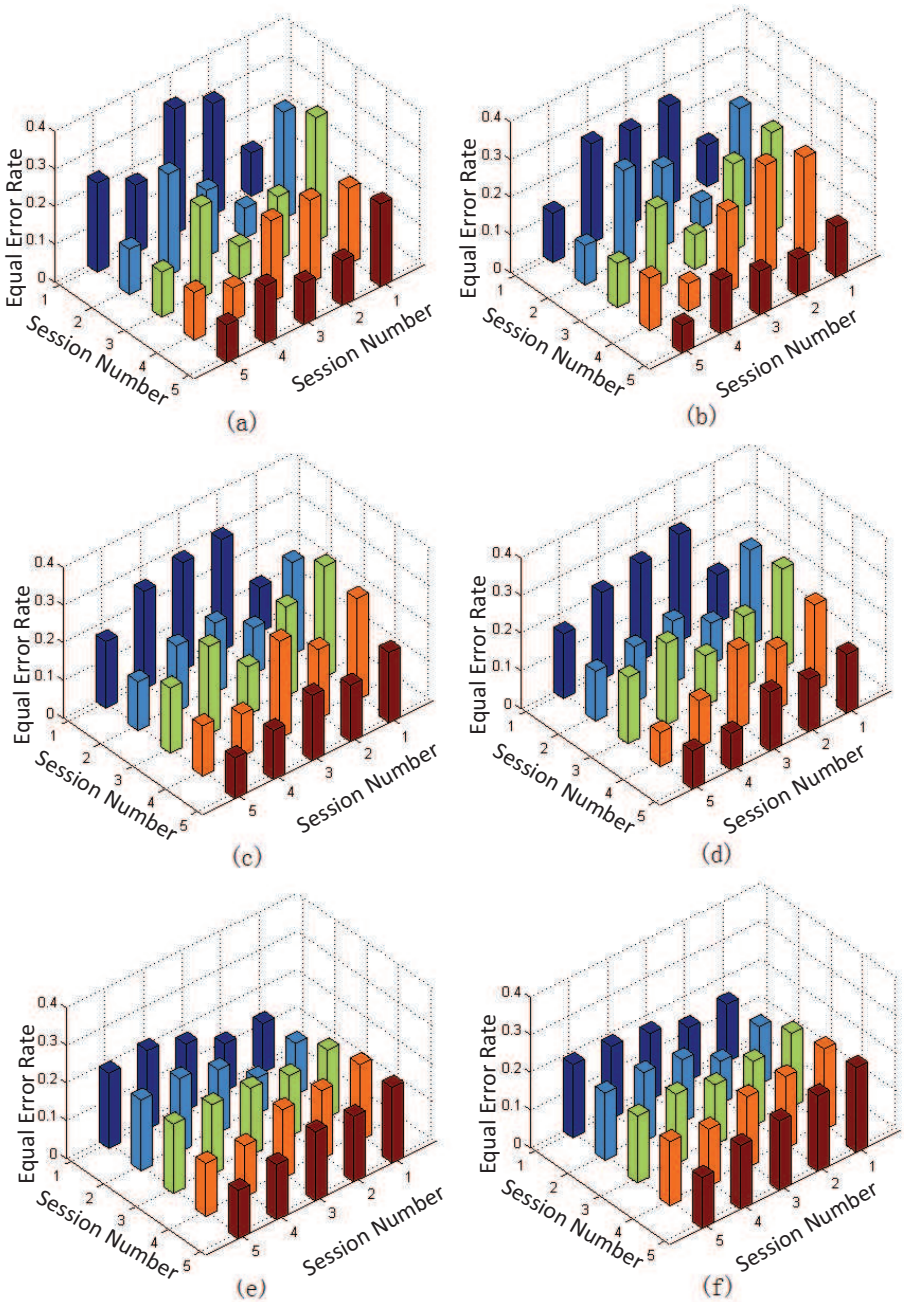


Figure 3.12: A depiction on the results of intersession authentication for different gestures: (a) DU, (b) UD, (c) LR, (d) RL, (e) ZI, (f) ZO. The height of bars is the Equal Error Rate, the X, Y axes are the session index and the bars with the same color have the same gallery set.

for different fusion schemes and methods: a) GTGF-A: fusion of the score matrices of the six gestures computed from the proposed method; b) GTGF-S: fusion of the score matrices of four single touchtip gestures computed from the proposed method; c) TA-A: fusion of the score matrices of the six gestures computed from the method in [29]; d) DM-A: fusion of the score matrices of the six gestures computed from the method in [61]; e) GTGF-M: fusion of the score matrices of the two multiple touchtip gestures computed from the proposed method. A Receiver operating characteristic (ROC) curve was created by plotting the true acceptance rate (TAR) vs. false acceptance rate (FAR), at various threshold settings. Since the true acceptance rate is equal to 100% substracting false rejection rate, the EER is the intersection between a ROC curve and the black straight line connecting [0%,100%] with [100%, 0%] in the figure. From Fig. 3.13, the EER are 2.62% (GTGF-A), 4.31% (GTGF-S), 6.07% (TA-A), 7.06% (DM-A) and 7.81% (GTGF-M). Comparing different fusion schemes, the best performance has been achieved by fusing all six gestures while the worst performance is obtained by fusing just two multiple fingertip gestures, namely ZO, ZI. On one hand, as most of score-level fusion scheme, combining more scores from different channels would probably increase the overall performance. GTGF-A scheme includes six gestures while GTGF-M includes only two. On the other hand, gestures ZO,ZI themselves have a relative low discriminative power compared to other gestures referred in Tab. 2. Furthermore, the performance of the proposed method demonstrated a clear improvement over the other methods in the scheme of combining all the six gestures. There are two reasons for the improvement. First, compared with the method in [61], we adopted

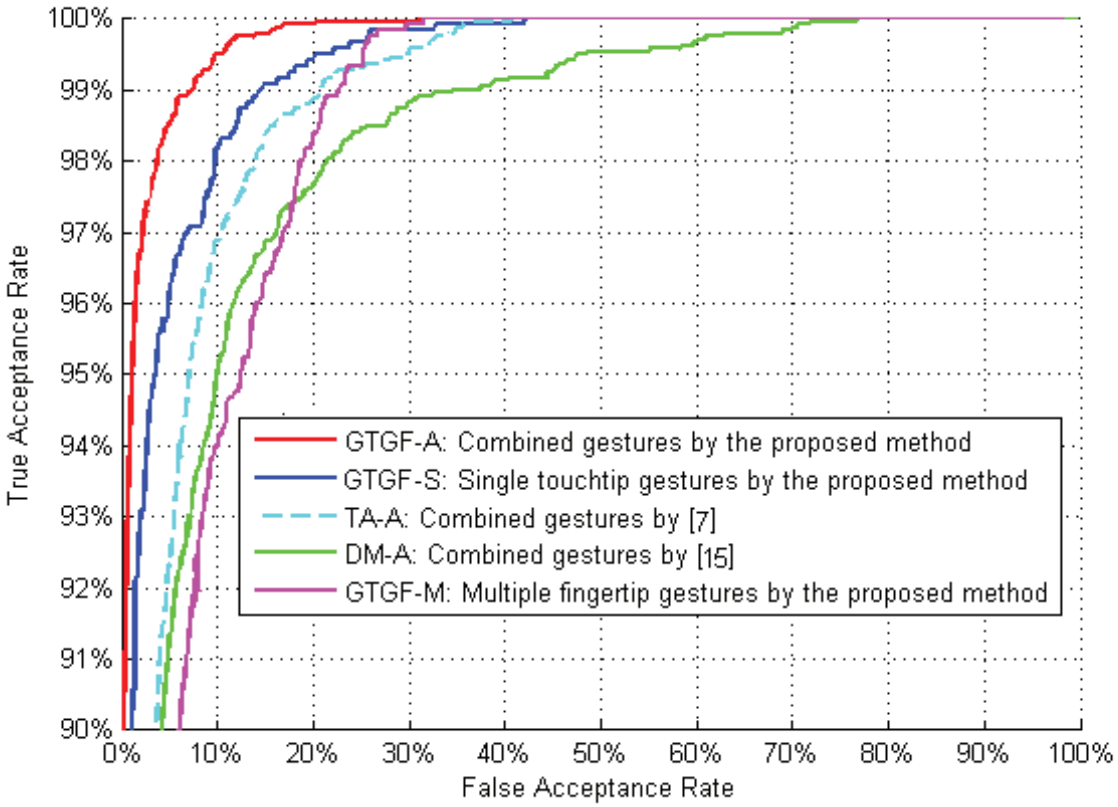


Figure 3.13: A comparison on ROC curves for different fusion schemes and methods.

the tactile pressure in our method, which contains extra clues on subject's muscle behavior. However, their method just ignored this information. Second, compared to the work in [29] which extracted 22 static analytic feature values, the proposed method includes movement dynamic and pressure dynamic of the touch gesture during feature extraction.

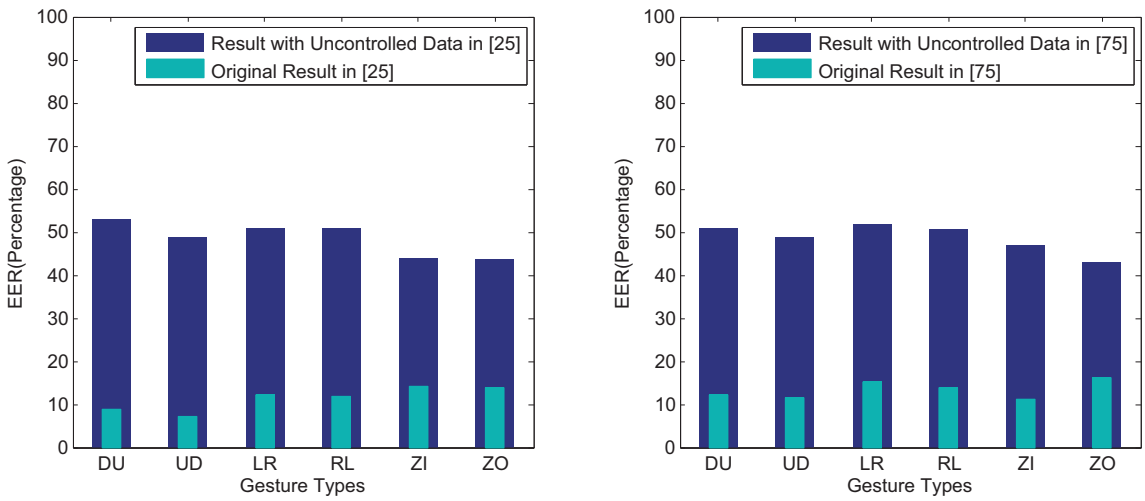


Figure 3.14: Performance of solutions in prior arts when applied on touch data collected in uncontrolled environment. DU, UD, LR, RL, ZI, and ZO indicates slide up, slide down, slide right, slide right, pinch, and spread, respectively. EER stands for equal error rate and it measures how often the user is mis-identified where the False Match Rate(FMR) equals False Non-Match Rate(FNMR).

3.3 Context-Aware Implicit User-Identification Using Touch Screen in Uncontrolled Environments

However, most of the current work solve the identity recognition problem under controlled environments. This means either users are required to perform pre-defined touch gesture patterns or the touchscreen data is collected under monitored laboratory environments. Thus, the collected touch data may not represent the **natural usage** of mobile devices and these methods may only be applicable for explicit identity recognition (e.g., at login screen). Due to these controlled conditions, these methods might be unable to capture the most valuable characteristics of touchscreen data for user-identity recognition **implicitly and continuously**.

(a) User 1 (b) User 2 (c) User 3

Figure 3.15: Three users' touch data in launcher applications

To justify our above hypothesis, we performed some experiments to evaluate the state-of-the-art methods in [25] and [73] under an uncontrolled environment using a touch dataset collected from users that are allowed to freely use the phone (e.g., checking emails and browsing the web). The performance plot in Fig. 3.14 indicates that these methods work well in touch data collected under controlled environments but fail to authenticate users under more natural conditions. The solutions in [61] and [29] employ pre-defined touch gestures, and some of them require users to use even five fingers which might be unnatural for users to perform. Therefore, these methods for explicit authentication might be difficult to apply in real-life situations for implicit and continuous identity recognition.

3.3.0.1 Touch Data in Uncontrolled Environments

Users can freely interact with mobile devices using four types of touch gestures in general: *click*, *swipe*, *zoom-in*, and *zoom-out*. In uncontrolled environments, the real-world touchscreen usage behavior is not as stationary as the one in controlled environments. The generated touch data becomes noisier and more unpredictable. There are two fundamental types of variations during natural touchscreen usage: *Usage Behavior* and *Application Context*.

3.3.0.1.1 Data Variation by Usage Behavior We implemented a background service on several Android devices to implicitly collect touchscreen usage data. Since the service is transparent to the device user, the collected touch data is natural and uncontrolled. Fig. 3.15 depicts three different users' touchscreen data in the launcher application collected over one week. The trace of the points reflect the swipe gesture, whereas the size of the points represent the size of touch area between users' fingers and the screen surface. In addition, different usage time is shown in different colors.

Fig. 3.15 shows that the touch data of different users has distinct motion patterns, and the data from the same user is not always uniform. For example, a single type of swipe gesture has significant variations in terms of *location, direction, curvature, length,* and *touch strength*. By investigating the touch behavior of users, we found that those variations might be caused by device holding patterns (*e.g.*, left-hand vs. right-hand holding the device, operating the device with one hand vs. both hands), user mobility patterns (the user is stationary, walking, or in traffic),

(a) Launcher (b) Browser (c) Map

Figure 3.16: One user's touch data in different applications

even caused by different usage behaviors over time and location (*e.g.*, weekdays at work vs. weekend at home) or longitudinal changes in user behavior over time (*e.g.*, touch data today vs. several months ago).

3.3.0.1.2 Data Variations by Application Context Users' touch gesture patterns are also strongly dependent on the current application being used and users' goal within the application. For instance, if a user tries to perform a scroll down operation while reading emails, the length of swipe gesture is mostly proportional to the length of the email. However, other applications (*e.g.*, map or browser) may not share similar scroll down operations. Fig. 3.16 depicts the touchscreen data of one user in launcher, browser, and map applications over one week. It can be seen

that user's touch patterns are significantly different and dependent on the running applications. These observations encouraged us to develop context-aware user-identification by analyzing backend information including running applications.

3.3.0.2 Research Questions and Contributions

Based on the data properties identified in uncontrolled environments, we raised three new research questions that are relevant when building touchscreen based user-identity recognition services in real-life applications.

1. **How to deal with data variations in uncontrolled environments?** As we have identified in previous sections, the touchscreen data in uncontrolled environments is noisier and non-stationary. This makes it difficult to apply static pattern recognition methods that are not adaptive to users' behavior and data variations over time. Thus, selection of a representative set of training examples of touch data is an important prerequisite before extracting useful features.

2. **How to improve the accuracy of user-identity recognition?** The information contained in a single touch gesture example may be not enough to discriminate one user from others since users may share similar touch gestures. This problem causes the difficulty in achieving high accuracy and highlights the importance of extracting highly discriminative features in combination with adaptive recognition methods.

3. **How to achieve real time recognition in practice?** Since our goal is to

develop algorithms that recognize user-identity in real time, on-device computation complexity is extremely important. System performance measurements must be considered to balance the trade-off between accuracy and computational cost on device.

We introduce a novel Touch-Based Identity Protection Service (TIPS) that addresses the above questions to perform user implicit identification in real-time in uncontrolled environments. Our main contributions are as follows:

- We study the characteristics of touch data in real-life uncontrolled environments and show that there are significant variations cannot be captured by existing methods focused on controlled environments.
- We design the TIPS - a novel, implicit, continuous, context-aware user identify recognition service using the naturalized touchscreen data. TIPS leverages a set of highly discriminant touchscreen data features as well as an adaptive sequential identification method.
- TIPS is implemented on the Android platform and evaluated extensively in practice with 123 users (23 device owners and 100 guests) and over 23 different phones (8 Galaxy S3, 3 Galaxy S4, 12 Nexus 4). TIPS can achieve over 90% accuracy in continuous user-identity recognition in real-time.

3.3.1 System Overview

Fig. 3.17 shows the high-level architecture overview for TIPS. TIPS collects the touch gestures input data and running application context information from the *Multi-touch Driver* and *Running Application Context Listener* respectively. The collected raw data is then transferred to the *Multi-touch Gesture Engine* for data preprocessing and feature extraction. In the training session, the pre-processed data is combined with the running application context information to generate a *Multi-touch Data Library* consisting of gesture templates, while in the authentication session, the touch inputs are used to first locate the templates in the *Multi-touch Data Library*, and then evaluated in the *touch-gesture-based User Authentication Module*. The *Touch-Gesture-Based User Authentication Module* compares incoming touch gestures with the templates in the *Multi-touch Data Library* and logs the result. When a user tries to access an app, the system will use the *App Library* to check whether the app allows unauthorized users. If it does not, the *Application Access Control* will prompt a password dialog to request explicit password. Otherwise, no action is performed unless there is a need to update the *Multi-Touch Data Library* due to a misclassification (adaptive component).

3.3.2 Touch-Screen Data Features

Previous works on touch data based identity recognition under controlled environment only considers biometric features. In our work, we consider two new sets of behavioral and contextual features to improve performance under uncontrolled

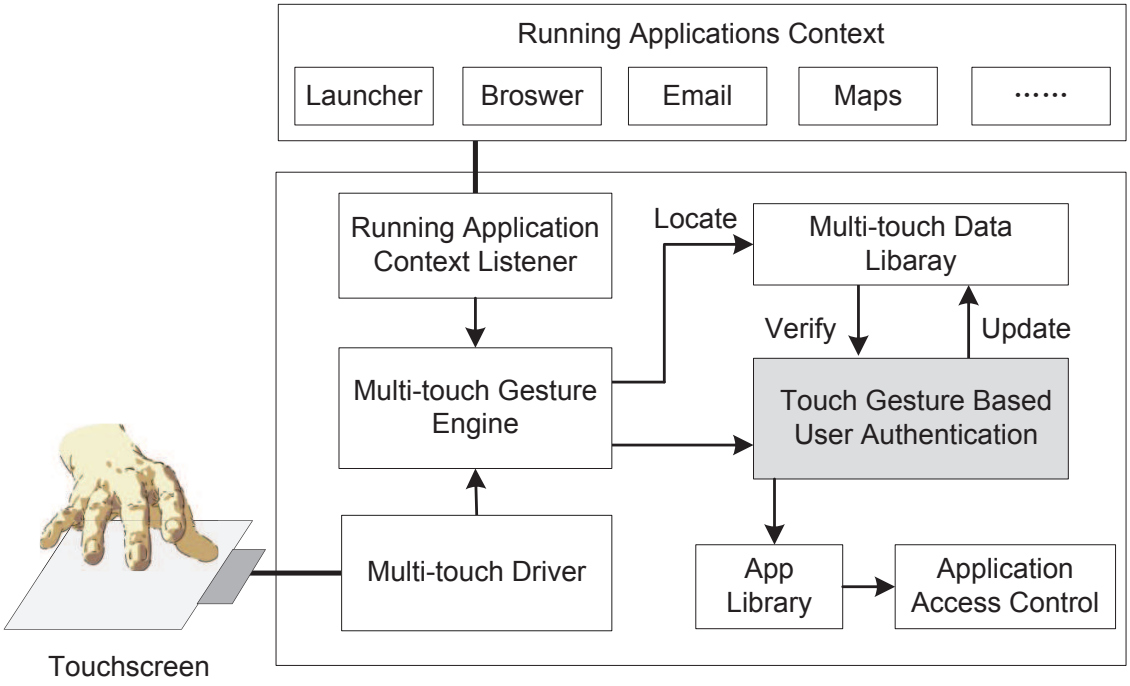


Figure 3.17: Design of TIPS

environments.

3.3.2.1 Biometric Features

Previous studies have shown that users' touchscreen data may have specific characteristics in terms of biometric features such as swipe speed and contact size.

Swipe Speed, SS_i**,** $i \in \{2, 3, ..., m\}$. reflects how fast a user performs a swipe touch gesture. Although this feature might be affected by user's current emotional state or environment, it is usually determined by the users' finger and hand muscles. Assume the speed of the first point $SS_1 = 0$, this feature can be calculated by the following equations, the θ_s is the screen size adjust metric, and x, y, and t are sensor readings of x-axis, y-axis, and timestamp:

$$
\begin{aligned}
SS_i &= \frac{\sqrt{\Delta x_i^2 + \Delta y_i^2}}{(t_i - t_{i-1}) * \theta_s}, i \in \{2, 3, ..., m\} \\
\Delta x_i^2 &= (x_i - x_{i-1}) \\
\Delta y_i^2 &= (y_i - y_{i-1}) \\
\theta_s &= \sqrt{(x_{max} - x_{min})^2 + (y_{max} - y_{min})^2}
\end{aligned} \tag{3.10}
$$

Points Curvature, PC_i**,** $i \in \{2, 3, ..., m\}$. represents the curvature between two consecutive touch points. Unlike the Swipe Curvature SC, which is most impacted by human behavior factors, this feature is most affected by user's hand and finger geometry. For each touch gesture $\mathfrak{G}$, the initial $PC_1 = 0$, and the rest are as shown in Eq. 3.11. x_i, y_i, and x_{i-1}, y_{i-1} are the x-axis and y-axis value of two consecutive touch points.

$$
PC_i = arctan(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}), i \in \{2, 3, ..., m\} \tag{3.11}
$$

Contact Size, CS_i**,** $i \in \{1, 2, ..., m\}$. is the contact surface area between users' finger and the touchscreen surface. The contact size value can be affected by how hard the user touches the screen, therefore sometimes it is also used as an approximation of touch pressure. Different mobile models employ different system readings to represent the contact size information. For instance, Samsung Galaxy S3 and S4 use TOUCH MAJOR, TOUCH MINOR, and WIDTH MAJOR, while Nexus 4 employs PRESSURE and TOUCH MAJOR. The contact size feature CS_i can be

calculated from these readings using the equation below, where s_{max} and s_{min} represent the maximum and minimum contact size value that can be captured by the touchscreen, and s_i is the contact size value of the touch gesture i.

$$CS_i = \frac{s_i - s_{min}}{s_{max} - s_{min}}, i \in \{1, 2, ..., m\} \quad (3.12)$$

3.3.2.2 Behavioral Features

From Fig. 3.15, it can be seen that specific behavioral features, such as touch location, swipe length, and swipe curvature are good indicators of users' behavioral patterns of interaction with mobile devices. We confirm this later in our experimental evaluation. These behavioral features are determined not only by users' touch behaviors, but also by the manner in which the users hold the mobile device. (*e.g.*, left-hand or right-hand holding, one hand vs. both hands.) We will explain these behavioral features respectively.

Touch Location, TL indicates the swipe location preference. For instance, when performing a vertical swipe gesture, some users like to do it on the left part of the touchscreen, while some others may prefer the right part of the touchscreen. We fractionalize the touchscreen into 16 areas, and assign values to each area of the touchscreen, the value matrix VM is shown in the following matrix.

$$VM = \begin{bmatrix} (0,0) & (1,0) & (2,0) & (3,0) \\ (0,1) & (1,1) & (2,1) & (3,1) \\ (0,2) & (1,2) & (2,2) & (3,2) \\ (0,3) & (1,3) & (2,3) & (3,3) \end{bmatrix}$$

,

By locating all the touch points P_i of a touch gesture $\mathfrak{G}$ in these areas, we can analyzing the touch location of $\mathfrak{G}$.

Swipe Length, SL represents the length of swipe gestures. This feature is application dependent. For example, during a left-to-right screen scroll operation in the launcher application, some users may swipe all the way on the touchscreen while others may only swipe a short distance. The SL can be calculated by Eq. 3.13.

$$SL = \frac{\sqrt{(x_{end} - x_{start})^2 + (y_{end} - y_{start})^2}}{\sqrt{(x_{max} - x_{min})^2 + (y_{max} - y_{min})^2}} \tag{3.13}$$

$$SC = arctan(\frac{y_{end} - y_{start}}{x_{end} - x_{start}}) \tag{3.14}$$

Swipe Curvature, SC is another useful feature which represents the slope of a users' swipe gestures. The value of SC is calculated by Eq. 3.14, where x_{start} and y_{start} respectively means the start point's x-axis and y-axis sensor reading, and x_{end} and y_{end} respectively means the end point's x-axis and y-axis sensor reading.

3.3.2.3 Context Features

As stated in the previous section, the running application context is extremely important for identity recognition. Users' touch gestures in the launcher application are significantly different from the same users' touch gestures in Email, Browser or Map applications. To address this, we maintain different gesture templates for each running application and perform adaptive classification. When a new application is installed on the smartphone device and has been used by the smartphone user, our system will automatically create a template database for this application and save templates for training.

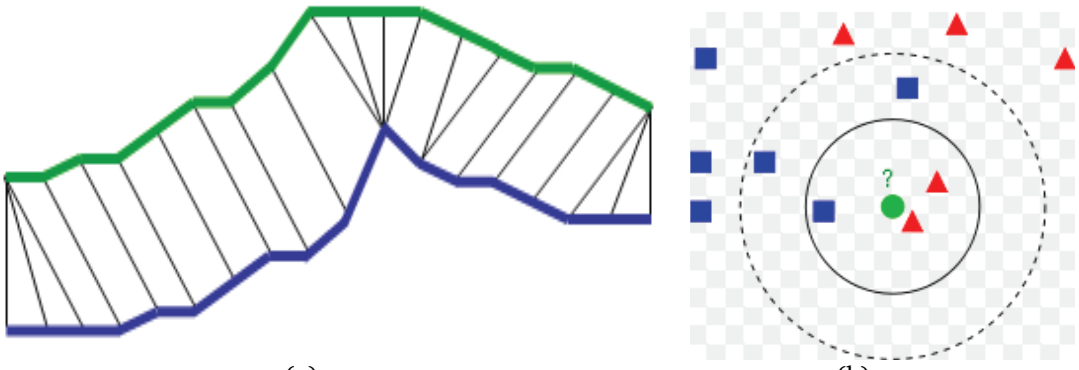
3.3.3 User-Identification Methods

In this section, we describe the details of our classification method employed to address the three research questions in Section 3.3.0.2.

3.3.3.1 1NN-DTW

To perform user-identity recognition, we combine the One Nearest Neighbor (1NN) classifier and Dynamic Time Warping (DTW). This allows us to capture the variety of user's touchscreen data by maintaining different gesture templates per application and adapt them over time and user behavior.

Dynamic Time Warping is considered an efficient way to measure the similarity between two time series. It works by computing the Euclidean distance between



(a) DTW (b) 1NN

Figure 3.18: Simple illustration to DTW and 1NN

any two input sequences of feature vectors and finds the optimal sequence alignment using dynamic programming. A simple illustration of the alignment process is shown in Fig. 3.18(a). The blue and green lines are two sequences. The black lines between them are the distance value, and by summing the shortest distance of each points on the two sequences, we can acquire the DTW distance of the two sequence.

One Nearest Neighbor Classifier is a non-parametric method for classifying objects based on the closest training example in the feature space. When applying this method to our problem, we calculate the DTW distance between an incoming touch input gesture (*e.g.*, the green circle in Fig. 3.18(b)) and all candidate gesture templates in the library. The label assigned to the incoming gesture is that of the closest gesture template in the library according to the DTW distance (*e.g.*, the red triangle in Fig. 3.18(b)).

3.3.3.2 Sequential Recognition

One way to recognize user's identity is to always use the single newest incoming gesture and compare it with the ones in the gesture template library as described above. However, this approach would not capture the temporal correlation of consecutive gesture inputs under natural uncontrolled environments. In this paper, we perform sequential user-identity recognition by first observing X number of consecutive gesture examples and accumulating their individual DTW distances (resulting from each pair of gesture comparison). We call X the *authentication length* and use it as a metric to define the number of most recent gestures used before providing an identity recognition result.

Gestures within the authentication length will be normalized and aggregated. Then the One Nearest Neighbor classifier is employed using the aggregated value to recognize the identity of the new touch input sequence.

3.3.3.3 Multi-Stage Filtering with Dynamic Template Adaptation

Theoretically, 1NN classifier can achieve very good classification performance by always comparing an incoming gesture with all the available template gestures in the library. However, the computational cost might be unacceptable. Another problem is that the template library must be updated with new training gestures regularly to compensate for user's gestural variations over time. To allow user adaptation without increasing the size of the library or recognition delay, we employ a multi-stage filtering technique combined with dynamic template adaptation

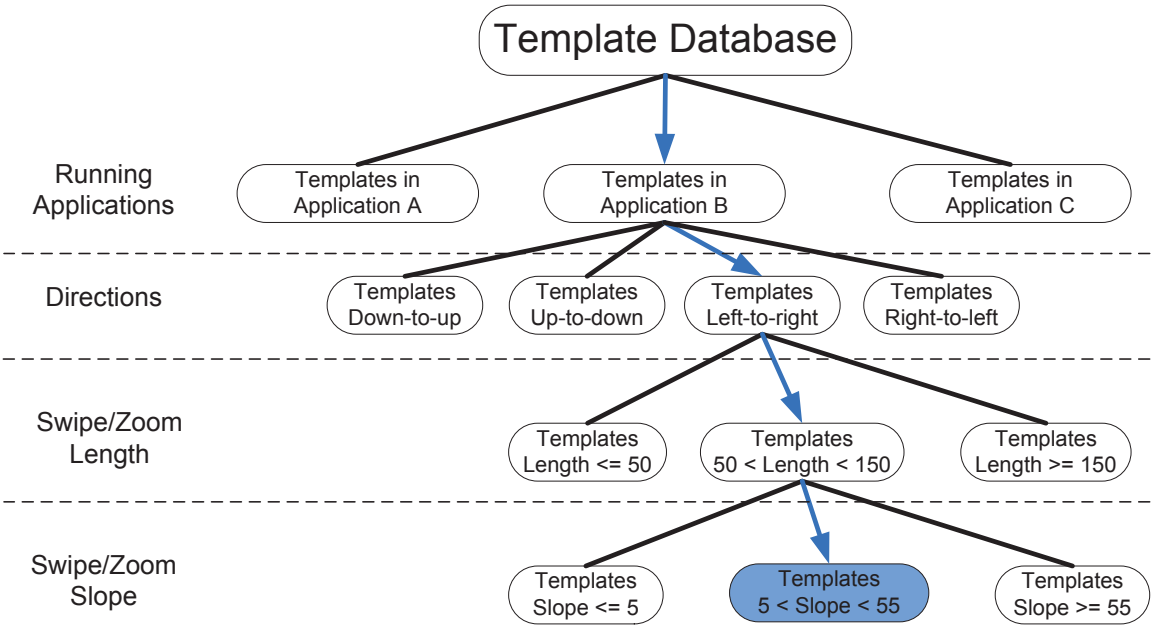


Figure 3.19: Illustration to the process of multi-stage filtering

to reduce the computational complexity while maintaining good performance.

Multi-Stage Filtering. To reduce the DTW distance calculation in each touch gesture recognition process, we impose a multi-stage filtering hierarchy over swipe/zoom gestures and click gestures. For swipe/zoom gestures, the hierarchy consists of the following four levels: (1) running applications, (2) direction, (3) swipe/zoom length, and (4) swipe/zoom curvature or slope. The changes of the order in the hierarchy would not result in different sample selection result. Fig. 3.19 shows an example of this hierarchy for a swipe input gesture in application B, with a direction of left-to-right, a swipe length of 100, and a slope of 30. For click gestures, the hierarchy consists of just two levels: (1) running application and (2) click location.

The running application layer ensures that only gestures belonging to the same

application are always compared. The direction layer further divides swipe/zoom templates into four classes: left-to-right, right-to-left, up-to-down, and down-to-up, while the click location further divides click templates into nine grid areas. The swipe/zoom length and swipe/zoom curvature layer are employed to reduce the variation caused by usage behavior following the direction layer. These two layers guarantee that only swipe/zoom gestures with similar length and curvature are compared. The above process helps reduce the number of templates a new incoming gesture needs to be compared against, thus improving recognition delay.

Dynamic Template Adaptation. To prevent our template library from growing unbounded as user's touchscreen usage behavior changes over time, we set a threshold on the number of training examples for each application to limit the size of the template database. When a new template adaptation request is detected (misclassification), TIPS will verify if the size of the library exceeds the threshold. If not, the new template will be added to the database directly. Otherwise, the oldest template will be replaced by the new template to maintain a constant computational complexity. If a touch gesture is recognized as an unauthorized input, the service will ask for password when users try to access a sensitive or customized application. A misclassification can be detected if the the user instantly inputs correct password and accesses the application.

3.3.4 Experiment and Results

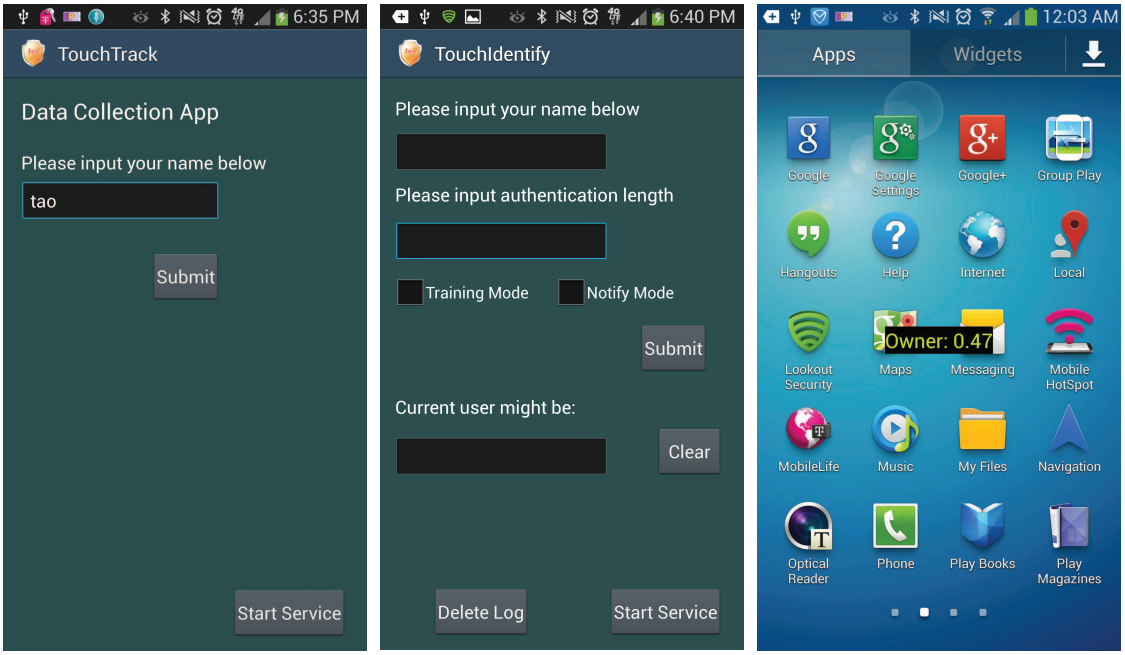
We evaluated the TIPS on Android devices for many users with both offline training and on-device real-time testing.

3.3.4.1 Experimental Setup

We implement TIPS as an Android background service that implicitly collects touchscreen data and authenticates user-identity continuously. All the touch inputs and application context are collected from system level. No information is provided by application side so there is no need to modify each single application to acquire the touch data. The TIPS app is installed by 23 smartphone users (14 males and 9 females) on their own primary phones (8 Samsung Galaxy S-III, 3 Galaxy S-IV and 12 Nexus 4). In addition, we recruit 100 guest users (not phone owner) to play with a subset of phones (13 phones) and the performance of real-time user authentication.

The experiment is consisted of two main phases:

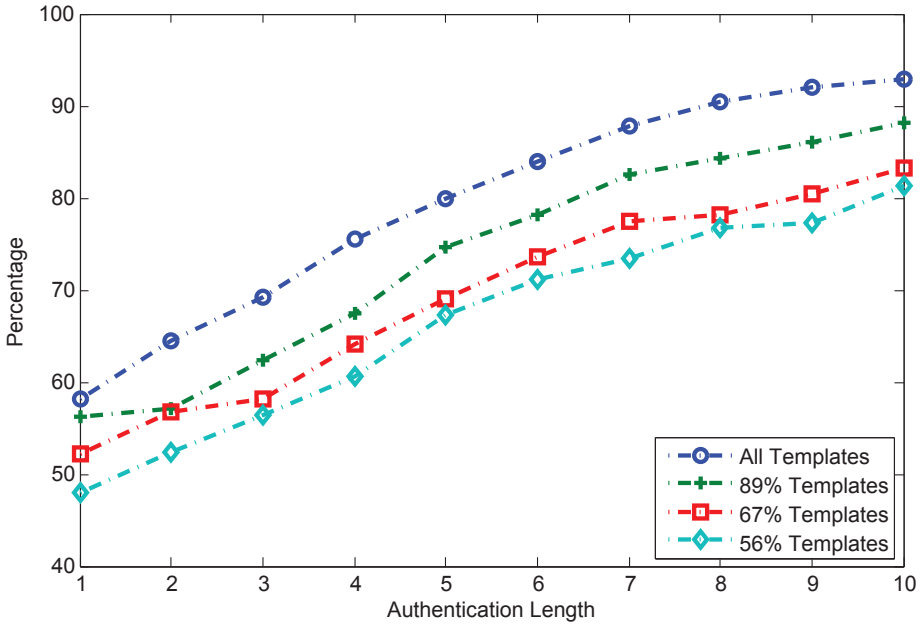
- *Off-device Simulation Phase:* We build off-device touchscreen data analysis using Matlab. The data is collected from 23 phones for 3 weeks, with triggered TIPS service, as shown in Fig. 3.20(a). Each data sequence includes timestamp, the raw touch data, and the underlying running application. 23 users' first week of touch data was used as training templates and the subsequent 2 weeks of data was employed as testing data.



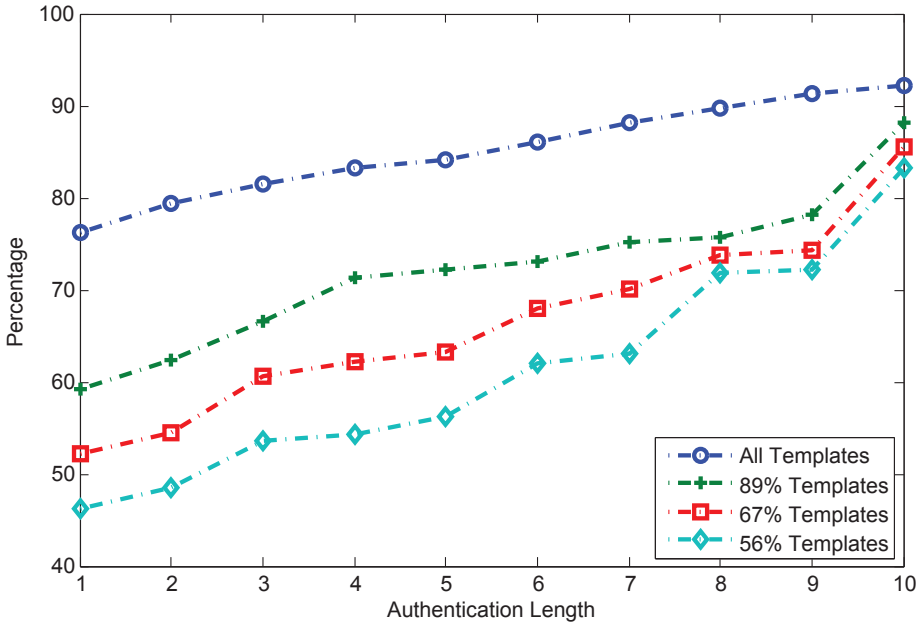
(a) Collection (b) Testing (c) Notifying

Figure 3.20: Hotservices developed for experiments

- *On-device Testing Phase:* We incorporate both online training and testing module into the TIPS service. The on-device training session takes one week and collects about 2000 touch gestures for each user. The user can customize the mode (training or notification) and the authentication length parameter, see Fig. 3.20(b). If the notification mode is selected, the authentication result will be shown in real-time in the middle of the screen (Fig. 3.20(c)) and the device will be locked if unauthorized user is detected. TIPS also logs authentication results into a file on the device for on-device performance evaluation.



(a) True positive



(b) True negative

Figure 3.21: Off-device simulation performance

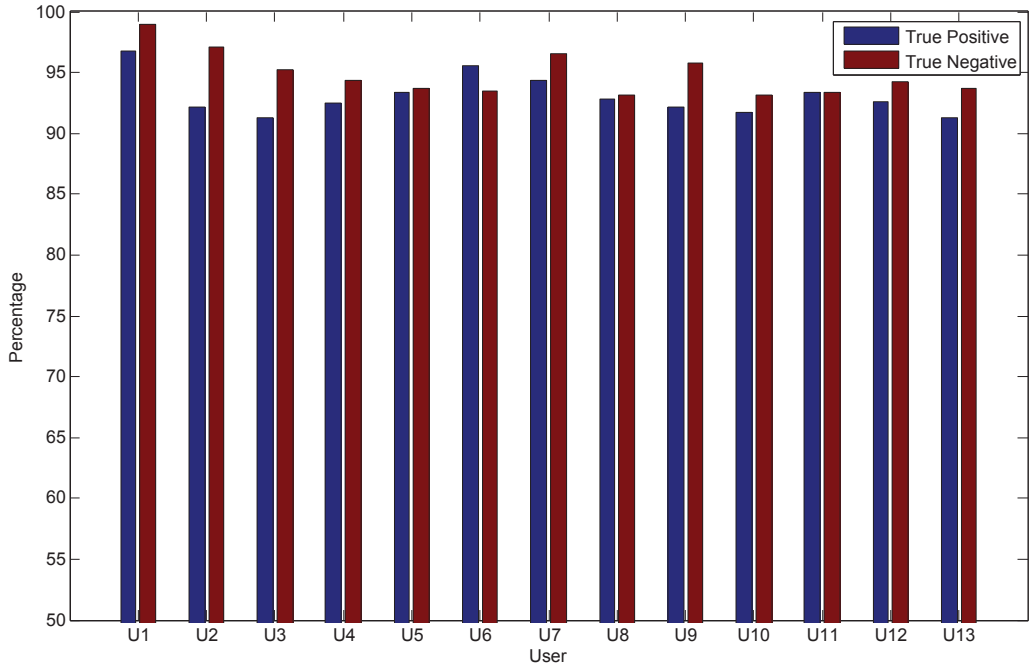


Figure 3.22: On-device testing accuracy

3.3.4.2 TIPS Off-device Simulation Results

In the simulation experiment, we evaluate the accuracy performance by setting different authentication length. The authentication length indicates how many touch inputs are employed for an identity authentication. A lager authentication length may guarantee a higher accuracy, however it may also result a longer delay. We employ two metrics, true positive and true negative, to represent the accuracy performance. True positive indicates that the portion of correctly recognized authorized inputs to all authorized inputs, whereas true negative means that the portion of correctly classified unauthorized inputs to all unauthorized inputs. We also verify our intuition that one or few patterns would not be sufficient for touch-based identity recognition in uncontrolled environments. We build a full template

library and also merge similar templates to reduce the size of template library.

Fig. 3.21 shows the simulation results. The plot (a) presents the true positive result under four different template database size settings. For instance, "*All Templates*" indicates results are being calculated using all templates collected in a week, whereas "*56% Templates*" means 44% of the templates are eliminated by merging them with similar templates. An interesting finding that support our hypotheses on pattern representation is: as template size decreases, the accuracy also decreases. This degradation is caused by the loss of information while reducing the number of templates. Significant accuracy improvements can be found as authentication length increased (for all cases with different templates database sizes). When the authentication length is 8, the true positive and true negative for "*All Templates*" already exceeds 90%. Therefore we set the authentication length to 8 in on-device testing phase. We observe similar performance trend for the metric of true negative, as shown in Fig. 3.21 (b).

3.3.4.3 TIPS On-device Testing Results

After obtaining good performance results in the simulation phase, we implemented the complete TIPS service (including both online training and notification modes) on Android phones for on-device practical test. After implicitly logging the classification results for natural touchscreen usage data over one week for 13 mobile device users, we computed the true positive rate. By requesting guest users to test these mobile devices, we computed the true negative rate. We plot the true positive and true negative rates in Fig. 3.22.

For all users, a true positive rate of 91% or more and a true negative rate of 93% or more were achieved under uncontrolled environments in real time. The result accuracy performance is evaluated based on touch sequences that combines 8 touch gestures, which indicates that during 8 natural touch inputs, TIPS can verify the identity of current user. Meanwhile, we also measured the power consumption of TIPS. From the energy usage data collected, we found that the power consumption of TIPS has an average value of 88 mW, and does not exceed 6.2% battery usage. These encouraging results show the potential of this solution to perform touch-based identity recognition in real-time and naturalistic usage.

3.3.4.4 TIPS Security and Usability Analysis

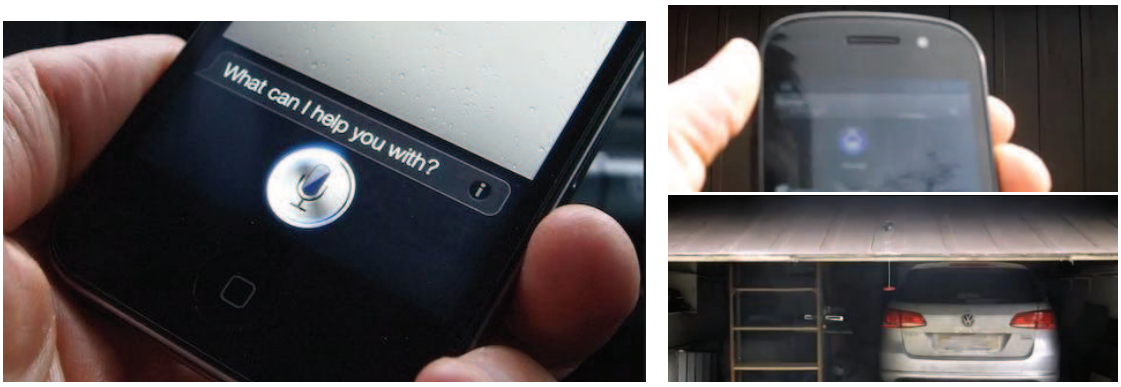
Although a 90% accuracy rate is not low for a machine learning problem, employing it in the real world may raise both security and usability concerns. Accepting 10% unauthorized input could be labelled as insecure, and rejecting 10% of the owners inputs could be labelled unusable. However, TIPS is not aimed at replacing explicit authentication mechanisms; instead, it is a complementary approach protecting the system after the users explicit login.

Chapter 4

Voice-Enabled Identity Awareness and Usable App Access Control

The commercialization of Automatic Speech Recognition (ASR) technologies has ushered in a new era of hands-free user-mobile device interactions. Unlike device adaptive interactions (e.g., touch gestures, typing), speech is a more intuitive, inter-personal communication medium [43]. This, coupled with the release of speech recognition services (e.g., Google Voice [37] and third party APIs [1, 4]), explains the surge in popularity of speech-based applications.

However, the popularity of speech recognition exposes its users to privacy vulnerabilities. Research has mainly focused on accuracy (i.e., "what the user is speaking") but not on identity management (i.e., "who is speaking"). This can significantly simplify the attackers task of accessing sensitive information *e.g.*, confidential documents, emails, contact lists) stored on victim's devices, or even allow the



(a) Bypass the login (b) Remote garage door control

Figure 4.1: Potential violations from current speech recognition framework: (a) passcode can be bypassed by Siri and Google Voice Actions, sensitive operations (*e.g.*, making phone calls, sending text messages, posting status) can be performed as if in an unlocked status; (b) sensitive applications such as garage door control do not have an identity authentication and opens to whomever holds the smartphone.

attacker to impersonate the user in highly sensitive operations (*e.g.*, posting status updates on social networks, initiating e-mail, SMS, or voice calls). Furthermore, we have shown that Siri and Google Voice Actions [5,8] can bypass the login stage authentication and follow requests of unauthorized users (see Figure 4.1). They may even enable unauthorized remote control applications. Although there are some app access control applications that may help improve user app security and avoid the aforementioned problems, mostly they are not easy to use and configure. This is due to the additional efforts of entering a password each time to access a locked app, as well as management of the locked app list. In addition to protecting user privacy, speech-based identity management can also promote mobile user experiences by allowing users to customize their app access control and function by voice commands.

Table 4.1: Example Android applications that use speech recognition.

Application	**Features**
Slyvi	wake on voice, find places and get directions, update Twitter and Facebook, car mode
Google voice search	search your phone, the web, and nearby locations by speaking, instead of typing. Call your contacts, get directions, and control your phone with voice Actions
Speaktoit	Speaktoit uses natural language technology to answer questions, find information, launch applications, and connect user with various web services. It remembers users' favorite places, services, and preferences.
Utter voice command	Utter voice command runs in the background. It does not have a user-interface and controls the device using voice commands. It supports drive mode and wake on voice commands.
Voice remote control camera	User can remote control the camera inside a SmartPhone. Responding to sounds, the camera works automatically and a user can take a photo hands free.
Drive safely	It reads text messages, SMS and emails aloud and lets you respond by voice.
AVX	It can remote control garage door respond by voice.

Previous work on speech recognition [19, 37, 42] and speaker-recognition [47, 48] has been applied on mobile devices, performing either implicit or explicit user authentication. However, to the best of our knowledge, no prior work has been jointly performed on mobile speaker and speech recognition to implement an identity awareness app access and function control framework.

We propose and implement a unified speech-speaker recognizer (USR) framework that performs permission and response management based upon a customized identity management policy. The USR framework consists of four main modules:

(i) an application interface cooperating with mobile applications, (ii) a speaker-recognition module for identity recognition, (iii) a speech recognition module transcribing speech input, and (iv) an identity-management module supervising the response to the applications according to the customized identity management policy. USR relies on several factors to perform seamless identity based application management including user-identity, application privacy level, and application function class. In comparison to previous mobile operation permission management applications, an essential feature of the proposed approach is its convenience. The identity feature (*i.e.*, speech) is implicitly captured without disrupting normal user-mobile device interactions. In addition, it offers continuous post-login protection of mobile devices during all the speech interactions, thus protecting sensitive mobile device information and functions. The contributions of USR are the following:

- Designed and implemented a unified speech-speaker recognizer (USR) framework that provides specific response corresponding to different user-identity based on a customized identity management policy.
- Developed an open-source Android library for speaker-recognition.
- Conducted a comparison study of USR and the Google speech recognition framework.

We designed USR framework to address the weakness of the current speech-recognition-based API that recognizes speech without verifying the speaker, a solution that integrates speaker sensing and identity management with speech recognition. A high-level diagram of the approach is presented in Figure 4.2. The

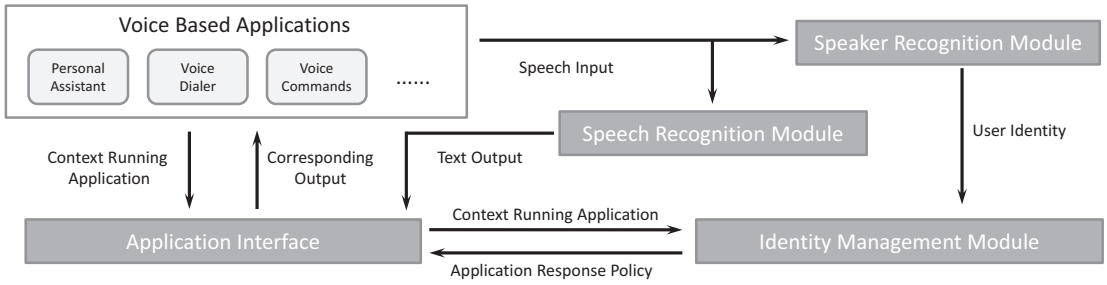


Figure 4.2: Design of USR framework

approach extends the Android speech recognition API with speaker-recognition, identity management support and access control. The new components include, an application interface that detects context running application and responds to the applications, an identity manager module that controls and enforces responding policies to speech commands based on speaker's identity, and a speaker-recognition module.

4.1 Application Interface

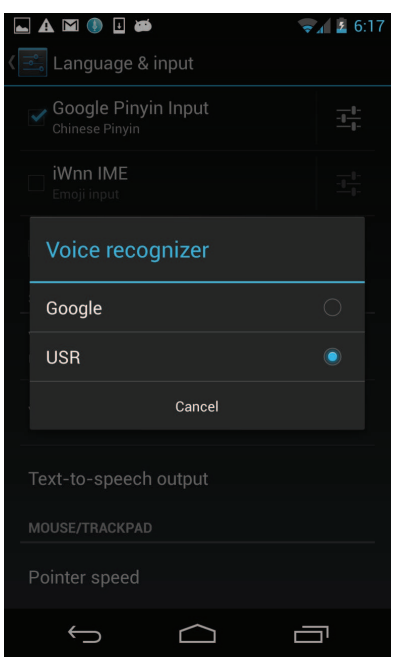
The application interface has two core functions. The first function is to detect which application is the owner of the microphone (*e.g.*, personal assistant, voice search, skype). We implement it by utilizing an Android System API, *Activity-ChangedListener*, to capture application package name (*e.g.*, "com.skype.raider" for Skype) in a background service. The application interface is then able to send the package name to identity-management module to acquire the corresponding application response policy for this application. Another important function of this module is to react to the application based on the application response policy. For

example, if the application response policy instructs the command is not allowed to send to the application, the application interface will prevent the text output from being send to the application.

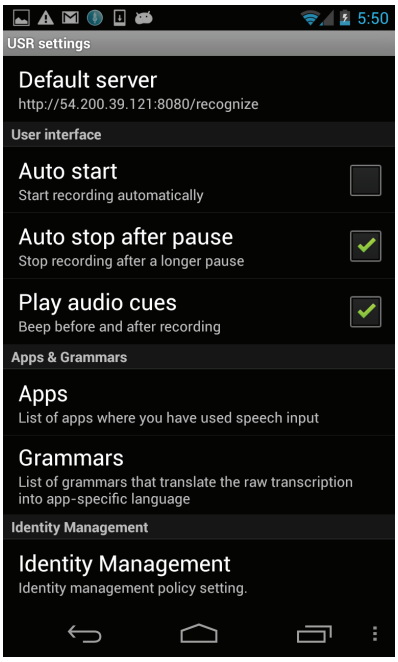
4.2 Speech-Recognition Module

The speech recognition module consists of a voice recognizer service and a speech recognition server. When USR framework is installed on the device, users can select our service in Android *Voice Recognizer* (See in Figure 4.3(a)). Users may configure the settings of the voice recognizer service, speaker recognizer, identity management policy, and other settings (*sampling rate*) by entering *voice search*. The details are shown in Figure 4.3(b) and Figure 4.3(c). By default the service connects to our speech recognition server, but the users can also connect to speech recognition server they desire by entering into the server list (Figure 4.3(d)).

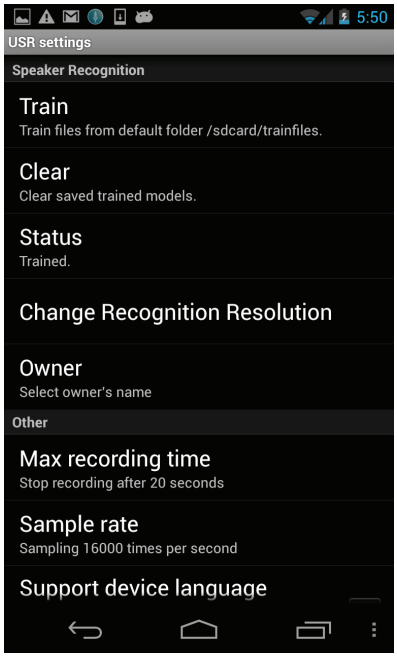
On the server side, considering the recognition accuracy, we choose Google Speech Recognition Server as the speech recognition server and implement our server as a proxy server to connect it. The reason we do not connect Google Speech Recognition Server directly on the mobile side is that Google Speech Recognition Server requires the voice data to be *FLAC* format, which is not a default encoding method supported by Android system.



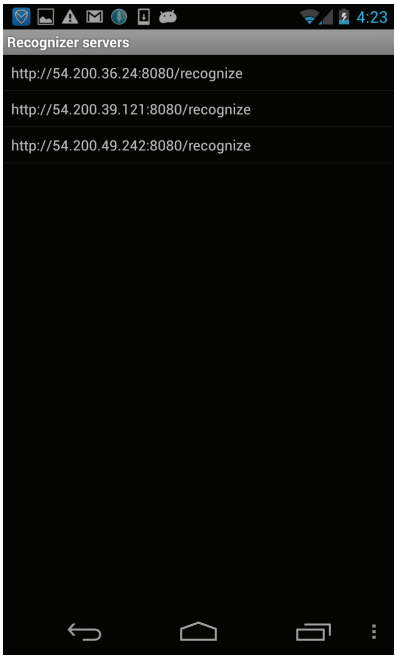
(a) Voice recognizer



(b) USR framework setting



(c) USR framework setting continue



(d) Server List

Figure 4.3: USR framework setting on the Android device

4.3 Speaker-Recognition Module

4.3.1 Speaker Recognizer Design

Methods involving a set of Mel Frequency Cepstral Coefficients (MFCCs) have been dominant in the field of speaker-recognition in the past decades. Human perception of the frequency content of sounds follows a subjectively defined nonlinear scale called the ”mel” scale [16] defined as,

$$f_{mel} = 1125ln(1 + \frac{f}{700}) \tag{4.1}$$

where f is the frequency in Hz. The calculation of MFCCSs can be summarized in Figure 4.4.

When it comes to speaker-recognition, vectors consisting of MFCCs and some features derived from them are used to build Gaussian Mixture Models (GMM) [58]. More recently, they have also been used in classification schemes that involve Support Vector Machines (SVM) [16] [71]. These methods have been very effective for user verification, often having prediction accuracy of up to 95 percent, with state-of-the-art speaker-recognition systems generally having Equal Error Rate (EER) close to 0.

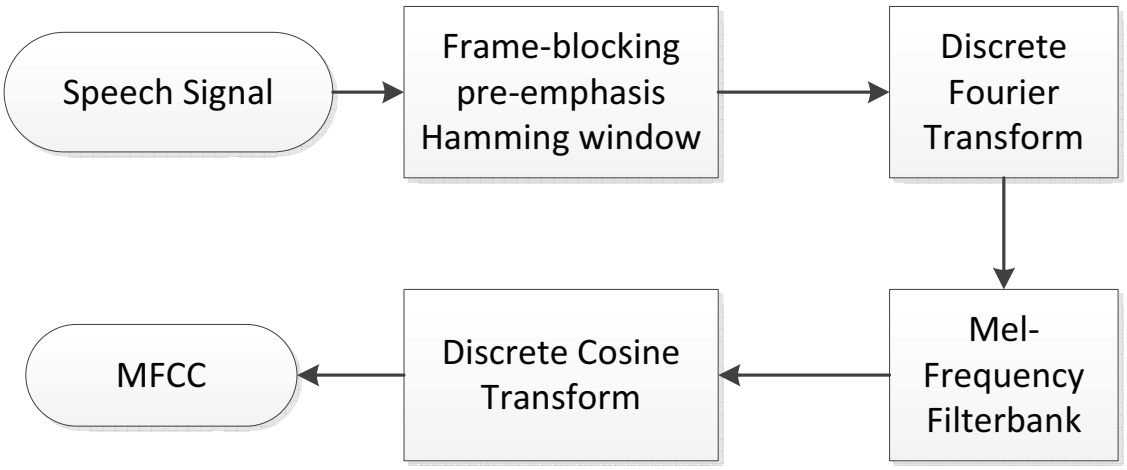


Figure 4.4: MFCC calculation process

Generally speaking, the following technique is standard: MFCC features are extracted over a chosen frame length with a frame shift of about 1/2 its size to provide for an overlap, then either their means and standard deviations or derivatives and second derivatives are computed [21]. One particular property of the described approach is that given the number of instances *n* and the number of features *m* for almost any reasonable data set, the following property holds: $n >> m$. When SVM classifier is applied to such problems, RBF or polynomial kernel is typically chosen as they transform the feature vectors into higher dimensional space, increasing the probability to find a suitable hyperplane to separate the classes. Unfortunately due to performance and battery use considerations, using anything but a linear kernel on a mobile device is simply not practical.

As a part of this study, we developed an open-source Android library for speaker-recognition. Figure 4.5 depicts a high-level diagram of the system. It consists of a Java interface and three internal modules, written in C and C++: i) Feature Extractor ii) Feature Pre-Processor iii) Classifier. For the purpose of feature extraction, openSMILE: The Munich Versatile and Fast Open-Source Audio Feature Extractor [20] has been ported to Android platform. openSMILE is capable of producing output in various formats, which usually makes it directly compatible with most machine-learning libraries, but in some cases, further processing of needed. To address this issue, component ii) has been written. Finally, based on the findings we will discuss further in this section, libSVM [15] has been re-compiled to work on the Android platform, as well. It is worth noting that our software is highly modular and extendable, allowing for extraction of various features or usage of different classifiers or even using several classifiers at once, simply by editing the text configuration file.

In our method, features are computed in 3 steps.

i) A set of Low-level descriptors (LLD) is extracted. The LLDs in question are:
Intensity, Loudness, 12 MFCC (Mel Frequency Cepstral Coefficients), Pitch (F0),
Probability of voicing, F0 envelope, 8 LSF (Line Spectral Frequencies),
Zero-Crossing Rate.

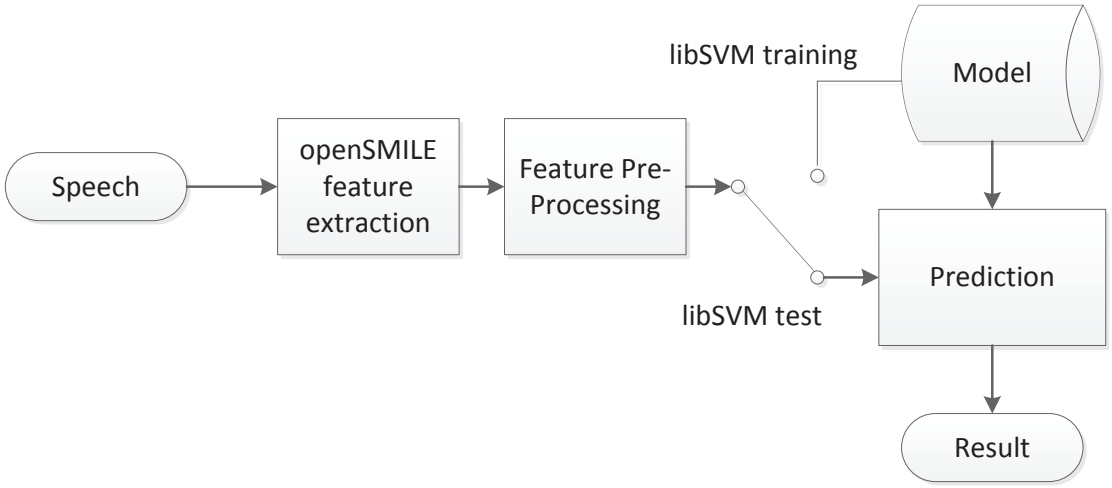
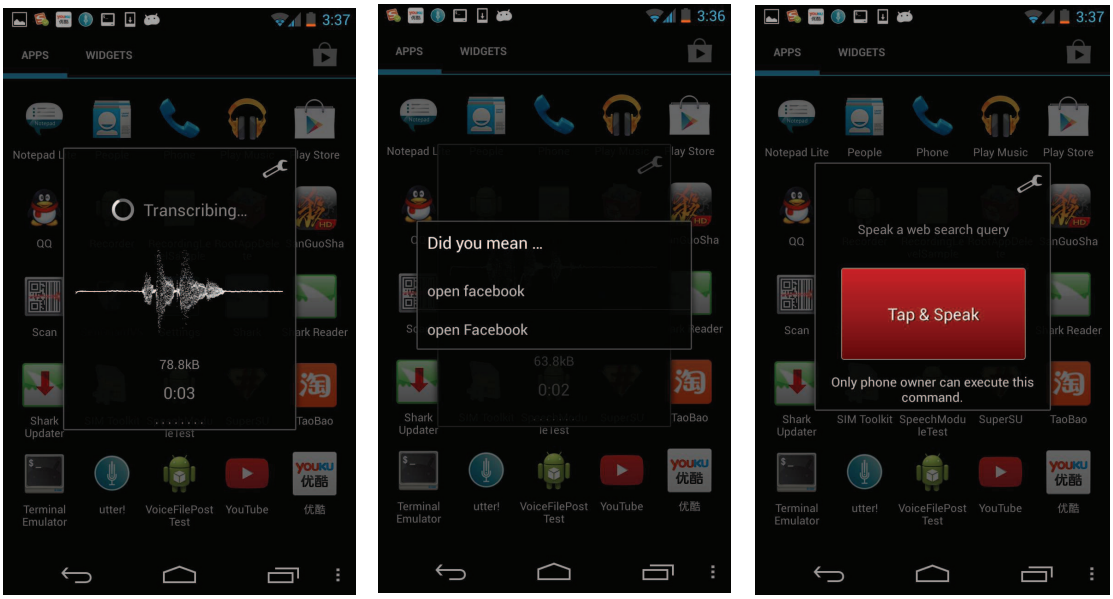


Figure 4.5: Design of speaker-recognition module

ii) Delta regression coefficients are computed from these LLD's.

iii) The following functions are applied to both the original LLDs and their delta coefficients: Max./Min. values and their respective relative positions within input, range, arithmetic mean, 2 linear regression coefficients, linear and quadratic error, standard deviation, skewness, kurtosis, quartile 13, 3 inter-quartile ranges.

The first two steps are quite similar to the usual method, in fact, MFCCs are computed in precisely the same manner. The remaining descriptors extracted are common for emotion and speaker trait recognition, but some, such as F0 and LSF have been proved useful in speaker-recognition. Step 3 results in 986 acoustic features, but reduces the number of instances to one per PCM file. Thus, we are



(a) Processing status (b) Identity match (c) Identity mismatch

Figure 4.6: USR speaker-recognition Demo

dealing with a matrix where the inverse of the stated property of classical MFCC features holds, namely, $n << m$.

To optimize our system for the smartphone environment, we propose a new method of speaker-recognition that is based on statistical descriptors of fundamental speech features. This scheme is normally used for emotion recognition [20] but not verification, so although the features themselves are not really new, their application is. Since the scheme employs, in its first stage, a modified version of the emobase feature set, proposed by Eyben et al., we named it Speaker Identification Base, or SIDBase. The proposed method, when applied to a verification problem, is almost twice faster and more power efficient than the traditional approach, while

maintaining accuracy, true positive rate (TPR) and false accept rate (FAR) similar to that of the state-of-the-art systems.

We take advantage of the increased number of dimensions and employ a linear SVM which benefits from a large number of features without suffering from the overfitting problem of most classifiers. Another reason to choose a Support Vector Machine classifier is that it is a two-class classifier, and speaker-recognition is essentially a binary problem, for which it is well-suited.

4.3.2 Speaker Recognizer in USR framework

The configuration of speaker-recognition can be found in Figure 4.3(c). Owner can perform a set of operations, such as train model, clear trained model, change owner's identity on the mobile device, etc.

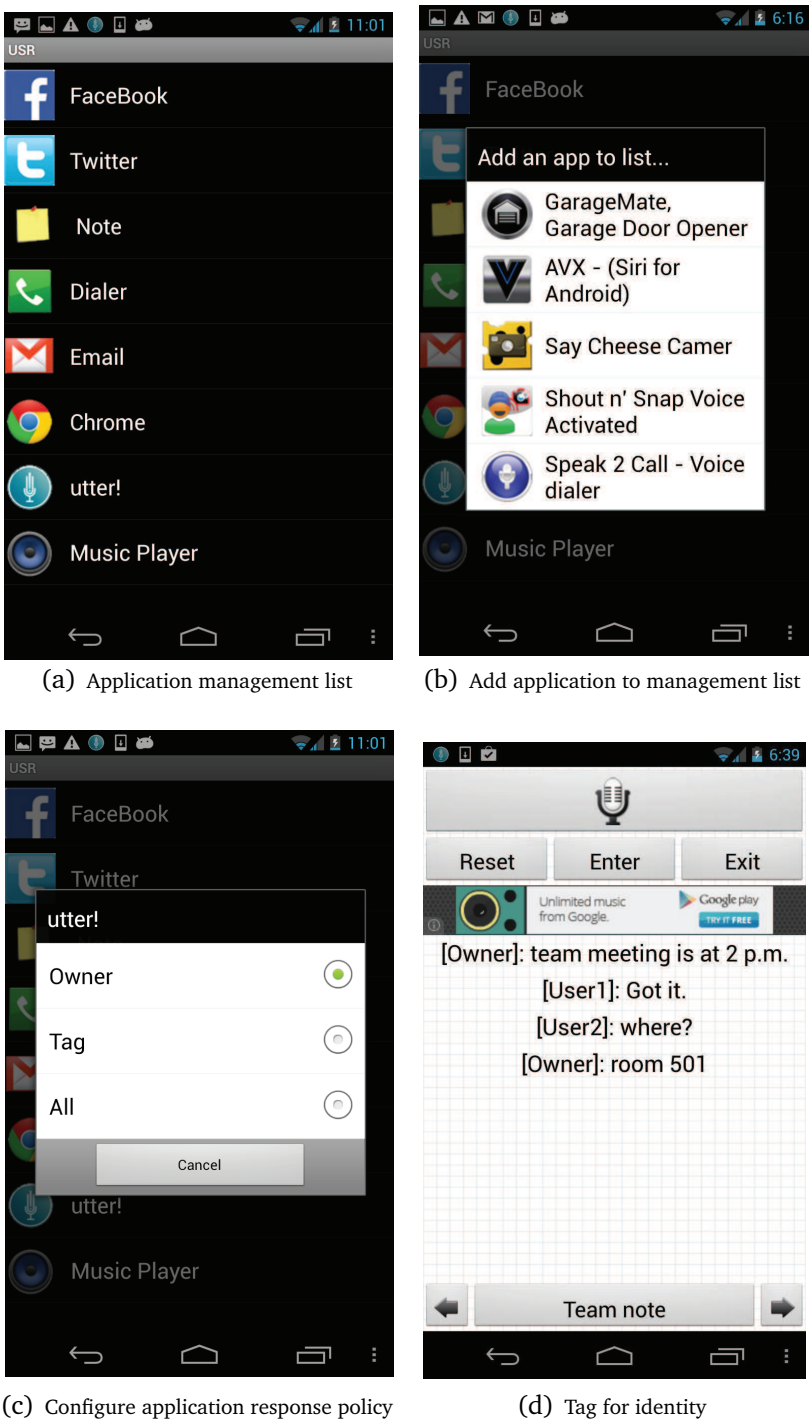
We employ a speaker-recognition demo application to demonstrate how it cooperates with speech recognition module. When a user speaks "Open Facebook" to the demo application, the application will record the voice file and show wave of input voice (Figure 4.6(a)). The voice file is then duplicated and processed in parallel by speaker-recognition in local and speech recognition in remote. If the user is the owner of the mobile, the application will show the transcribed result as in Figure 4.6(b). Otherwise, the application will return no result and send

the reject notice (*e.g.*, "Only phone owner can execute this command", "Are you Kelvin(*owner name*)?") to the user both in speech and text. (Figure 4.6(c)).

4.4 Identity-Management Module

The identity-management module can act according to the speaker-recognition results and the preset customized identity management policies. The owner of the mobile can configure the identity management policies in the USR framework setting (Figure 4.3(b)). All the applications have preset identity management policies will show as a list in the setting (Figure 4.7(a)). The owner can add new application to the management list (Figure 4.7(b)). For an application in the list, the owner may modify the identity management policy to choose whose speech inputs the application should respond to (Figure 4.7(c)). Currently, in our design, we have three types of policies:

- **Owner**. In this setting, a speech-recognition-based mobile app only responds to speech commands from a verified speaker;
- **All**. Under this setting, any user can access the app using speech without authentication; and
- **Tag**. Tagging is a novel feature of our USR framework. Instead of policing speech-based access to applications, it returns the recognized speech text with an identity tag in front of it as a prefix.



(a) Application management list

(b) Add application to management list

(c) Configure application response policy

(d) Tag for identity

Figure 4.7: USR Identity Management

For example, a user may label a hands-free voice dialer with Owner. When detecting that an unverified user is accessing the application through speech commands, the speaker identity manager will point out the current user is not the owner of the mobile device and refuse to follow the speech commands. The system will not affect speakers with verified identities as they can continue to interact with the device hands free.

In another scenario, the owner labels a notepad with Tag, the application may automatically record meeting minutes with identity if all the people in the meeting have trained model on the device (Figure 4.7(d)). More importantly, this Tag policy is intended to promote user experience by providing a **identity awareness interface**. This interface is not limited to benefit one or two single applications, but a general identity management interface for all the applications. A lot of similar scenarios (*e.g.*, posting group status on social network, automatically account switching, etc) may also benefit from our interface. It can be inferred that this identity awareness interface has a potential to greatly improve user experience.

4.5 Experiment and User Study

We conducted an empirical study in order to explore the benefits and limitations of USR framework. As a baseline for comparison, we used Google Speech Recognizer, a widely used and powerful speech recognizer which dominates the Android market, and AppLock, a representation of current app access control approachs. The

focus of the contrast experiment is not the speech recognition accuracy (because we all use Google Speech Recognition Server), but rather the privacy protection and user experience promoted by identity awareness as well as the extra instituted cost.

4.5.1 Model Training

Before we conduct the user study, we first need to train the owner's model on the smartphone device. The owner users are provided a Nexus 4 smartphone with one week for their everyday usage. They use the devices naturally and the USR training module will implicitly record their voice commands and send to the USR server. For each owner user, about 400 voice commands (average 20 times for 1 command, for 20 commands) is fairly enough for training a user's model.

4.5.2 Participant

Sixteen participants (6 female, 10 male), between the ages of 21 and 45, were recruited as mobile owner user. Another sixty subjects (24 female, 36 male), between the ages of 19 and 43 years, are enrolled as the guest users. We pre-trained the model of all mobile owner users with the voice data collected from them as positive samples. For the negative samples, we employed several voice data sets from different sources, i) a LDC speech dataset [2] that was purchased, consisting of 630 speakers of eight major dialects of American English, each reading ten phonetically rich sentences; ii) an open source voice dataset ELSDSR [22], which

contains voice messages from 22 speakers (10 female, 12 male), with ages from 24 to 63 included; and iii) a voice dataset collected from our previous data collection (12 volunteers with 6 female and 6 male). By considering speaker-recognition performance variations caused by both dialects and accents, we selected subjects from different countries and regions, including different dialects of American native speakers, Europeans, Chinese, Indians, etc.

All of the participants reported having used smart mobile devices, such as a smartphone or tablet. 75% of the mobile owner users and 68.3% of the guest users reported being familiar with voice operations on mobiles devices.

4.5.3 Design

We installed the USR framework on four Google Nexus 4 smartphones with all 16 mobile owner users' model pre-trained on each device. The mobile owner users take turns to use the mobile devices to perform operations following our experiment procedures.

The experiment consisted of two different tasks for different user groups and purposes. The first task is a privacy evaluation task. Both the mobile owner user and the guest user are required to enroll in this task. The second task is usability evaluation task, which is only conducted on the mobile owner users.

4.5.3.1 T1-Privacy Evaluation Task

This task is utilized to evaluate the privacy protection enhancements of the USR framework in comparison to the Google Speech Recognizer, which provides no identity authentication. We first gave the participants a quick tutorial on voice operations and voice applications. Then we demonstrated the security vulnerabilities in Google Speech Recognizer by showing them how to bypass the login authentication and use some sensitive operations (*e.g.*, post Facebook status, call, or send SMS to someone). Furthermore, we showed them some sensitive applications that can remotely control garage doors, automobiles, etc., all without identity authentication.

After said introduction, we let the mobile owner users set the identity management policy in the USR framework, and asked each guest user to speak at least ten different voice commands to sensitive applications or perform sensitive operations in both quiet and noisy environments. We collected the performance results of the experiment and the execution times of the USR framework with controlled execution time using Google Speech Recognizer. An exit questionnaire and interview were also required for both the mobile owner user and guest user.

4.5.3.2 T2-Usability Evaluation Task

To evaluate the usability promotion of the USR framework in comparison to the Google Speech Recognizer plus AppLock, we first asked the mobile owner users to

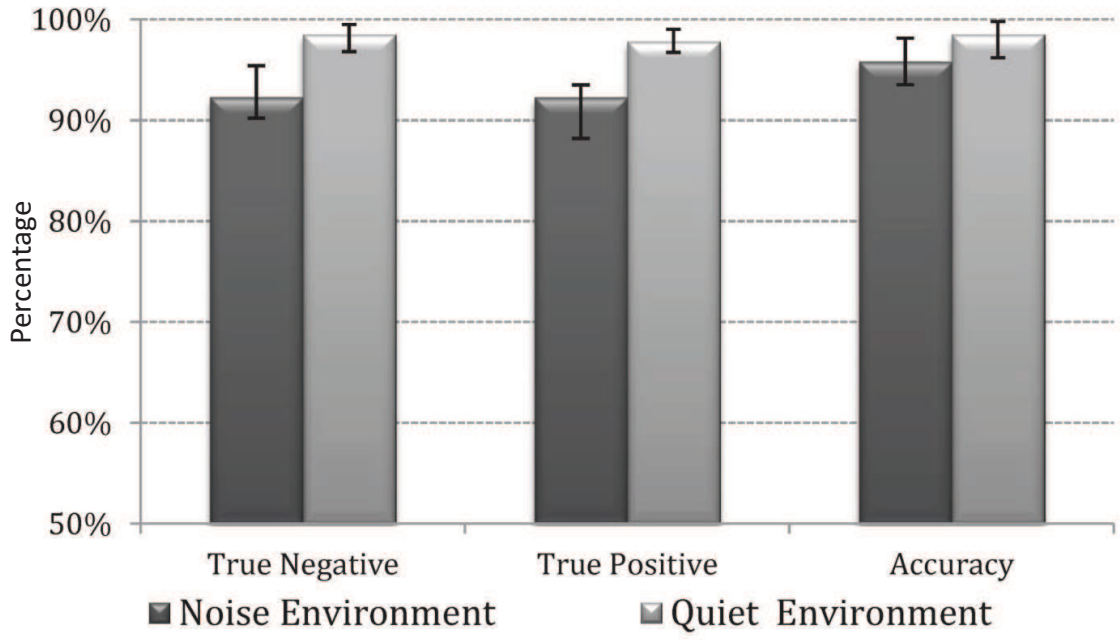


Figure 4.8: Speaker recognizer performance result

input voice commands in both quiet and noisy environments to their device to access the apps they desire to use. We then asked the users to repeat the same action by using the Speech Recognizer plus AppLock. After the comparison experiment, we simulated a multi-user scenario that requires four mobile owner users to using a notepad with Tag mode in the USR framework. Both the self-test scenario and multi-user scenario were recorded. As in the previous task, the execution times are also collected for cost evaluations. After the experiments, we introduced the potential use case of multi-user scenarios to the mobile owner users, especially highlighting its extensibility. The mobile owner users also complete an exit questionnaire and interview at the end of this task.

4.5.4 Performance Evaluation

Figure 4.8 shows the speaker-recognition accuracy performance result of privacy evaluation task and usability evaluation task. Even if we have implemented a noisy filter in USR framework, our speaker recognizer performs as we intuitively expected: in the quiet environment, all true positive, true negative, and accuracy rate are better than noisy environment. However, on the other hand, an accuracy of 95.83% also shows its ability in countering noise. Such a high accuracy rate is also a guarantee to the stability of USR framework.

Table 4.2 depicts the average extra time cost of USR framework compare to Google Speech Recognizer. Since the quiet environment is in our laboratory where high speed wifi is provided, and the noisy environment is on the street where only 3G plan is available, the extra delay in noisy environment may come from the network connection status and transmission rate. Another explicit trend we can found is that, as the length of commands increase, the delay also increases. This may result in two reasons: i) larger wave file further expose the low efficiency of our server comparing to Google Speech Recognition server; and ii) speaker-recognition for larger wave file will cost more time than small files. Since speaker-recognition and speech recognition work parallel in USR framework, the speech result maybe blocked by the speaker-recognition process. Although the for some long sentences, USR framework may cost close to 50% extra time to process it, it is still just a several seconds longer waiting time. Compared to the security it brings, this cost is acceptable.

Length of Command	Delay in Noisy	Delay in Quiet
1 to 3 words	25.18%	18.29%
4 to 6 words	35.92%	33.52%
6 and above	45.62%	39.12%

Table 4.2: Execution time delay comparing to Google Speech Recognizer

4.5.5 Questionnaires

After each task, we asked the participants to answer a post-study questionnaire about the privacy protection enhancement and usability promotion of USR framework for T1 and T2, using a 5-point Likert scale (1: Strongly Disagree and 5: Strongly Agree). Since currently there is no similar framework that protects user privacy or performs identity management during speech interactions, these Likert-scale results are not meant to be directly compared with other previous approaches.

For the first task, Overall, the participants thought USR improved privacy protection during human-mobile speech interactions (47 guest users and 15 mobile owner users Strongly Agreed, and 13 guest users and 1 mobile owner user Agreed) and the extra time delay is acceptable (48 guest users and 14 mobile owner users Strongly Agreed, 9 guest users and 2 mobile owner users Agreed, and 3 guest users Neutral).

Most of the users also thought a low rate false reject rate is acceptable considering enhanced privacy, also the Google Speech Recognizer plus Applock is more annoying since it requires explicit password input all the time (11 mobile owner users Strongly Agreed, 3 mobile owner users Agreed, 1 mobile owner user Neutral, and 1 mobile owner user Disagreed). In particular, the participants all found the Tag mode in identity management for multi-user scenarios very interesting (14 Strongly Agreed and 2 Agreed), and have potential to extend some useful applications (12 Strongly Agreed and 4 Agreed).

4.5.6 Interviews and Observations

The initial observations from the questionnaires indicated that smartphone users admit the privacy enhancement and usability promotion by the USR framework. To better understanding the detailed experience and feedback of the users, at the end of the study, participants were asked to state their attitude towards USR framework and why. Three of the mobile owner users stated that they had read the news about how to use Siri or Google Voice Action to bypass mobile login procedures, two of them are worried about the privacy vulnerabilities and have even disabled this function. After experiencing our USR framework, they feel their privacy can be properly preserved by it.

"I am impressed by the accuracy of USR framework. During the experiment of Task 1, seldom [guest] users can bypass the USR [framework] and control my device. Some [guest] users even try to mimic my accent and voice, however they still failed."-Mobile

owner user 2

"Some apps are too sensitive, like a garage control application I saw the other day [some one] can open the garage without any authentication. USR [framework] could at least provide an extra protection to those [sensitive] apps."-Mobile owner user 4

Although guest users are intend to simulate malicious intruders in Task 1, most of them share similar opinion that USR framework offers sufficient privacy protection after failing to command mobile owners' device for several times. Particularly, one guest user thought the USR framework is really helpful and can be ported to wearable devices, such as Google Glass, or Samsung Watch.

"I think this technology is useful because it implicit identify user during normal speech interactions. Since wearable devices have become more and more popular, it [USR framework] may be extend to broader usage scenarios."-Guest user 22

However, different from the encouraging feedbacks on the privacy protection, one mobile owner user considered the USR framework sacrifice too much usability in his point of view.

"Although it [USR framework] protects my privacy on the mobile device, it makes the system hard to operate for me. In a noisy environment, I was faulty recognized as a guest user within ten commands a time, which means I have to repeat one command in ten commands. I think this put extra burden on me."-Mobile owner user 8, the mobile owner user with worst performance rate and the only user to have a true positive under noisy environment.

Although current speaker recognizer performs good and designing a speaker recognizer is part of our study, it is not the most critical part. The core contribution in this work is to first propose a unified speech-speaker recognizer framework, whereas the speaker recognizer performance can be enhanced by further researching on our own speaker recognizer or simply employ and integrate other available advanced approaches. And comparing the other app access control approaches, it does improved usability since it will only require explicit login when there is a false alarm in USR - for AppLock, authentication takes place every time users try to open an app.

Nevertheless, all mobile owner users are interested in the Tag mode in identity management for multi-user scenarios, and some are extremely attract by this feature and provide us some constructive suggestions.

"Well, this idea is amazing. I can imagine several use scenarios now. It [Tag mode] could be used for implicit identity switching, like implicit switch facebook account when the command post facebook status comes from different user on a shared device. Furthermore, unlike current single user status, it may help social network extending some group activity or similar stuffs."-Mobile owner user 2, a mobile owner user with computer science background

In conclusion, participants appreciated the enhanced protection by USR framework and admit it solves practical problem in real world. For most of them, the sacrificed usability caused by USR (*e.g.*, false reject rate or response delay) is either not notable or acceptable. Specifically, all the users endorse the new Tag mode

since its potential to providing fancy services.

Chapter 5

Motion-Sensor-Based User-Identity and Context Sensing

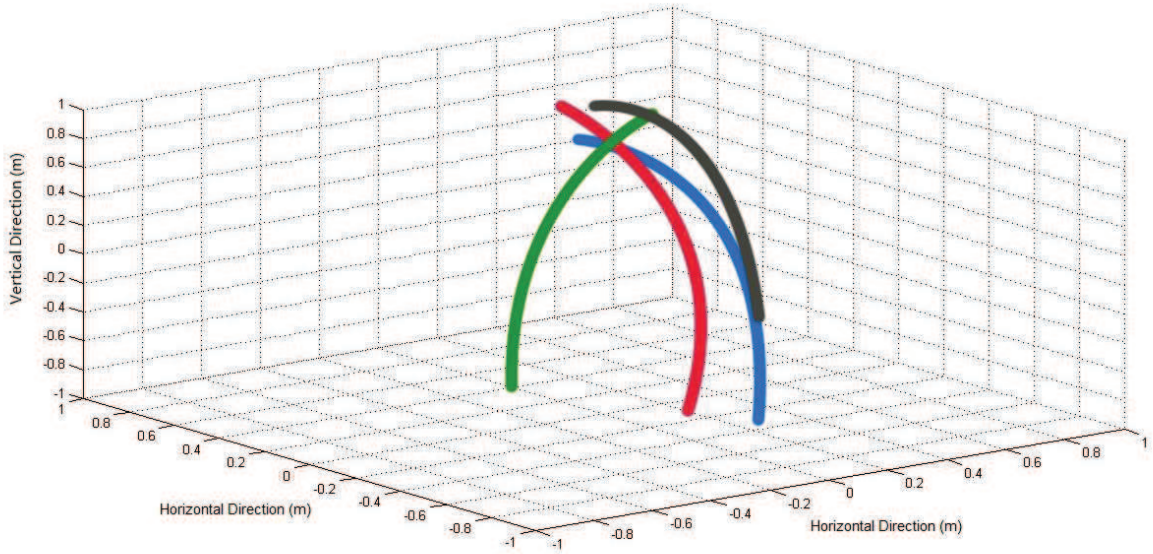
The useful gestures are performed when users interact with the phone via holding the mobile devices in hand, moving the arm or/and changing hand pose. Though users can create numerous gestures, the number of frequent gestures is small due to the limited usage of the mobile devices. These common gestures generated during the human-device interaction are shared by most of the users, which provides us an opportunity to compare their motion patterns and conduct non-intrusive biometric authentication. For example, one major usage of the smartphone is answering the call, for this purpose users perform a gesture of picking up their phone, which is referred as Mobile Device Picking-up (MDP) motion. Meanwhile, although Bluetooth headsets may avoid the MDP motion when answering a phone call, estimated by [6], in 2015, there is still only a 45 million market for the Bluetooth headsets

for all purpose of usage. Comparing it to the market of smartphone, it is easy to understand most people (at least 95.8%) performs MDP motion when answers phone call.

We investigate the MDP motion-based implicit mobile authentication via two methods, a Statistical Method (SM), and a Trajectory Reconstruction Method (TRM). For the SM, it first smoothes the sensor data by a low pass filter, then extract temporal and numerical features and apply verification algorithms on the features after normalization and segmentation. While for the TRM, it adopts Kalman filter and rotation matrix to reconstruct the MDP motion trajectories by utilizing the data from multiple sensors, and verifies the users' identity by score metrics calculated by Discrete Frèchet Distance. We tested the proposed methods on two sets of MDP motions, one stationary MDP motion(the user performs MDP motion sitting or standing stationary), and one moving MDP motion(the user performs MDP motion when walking).

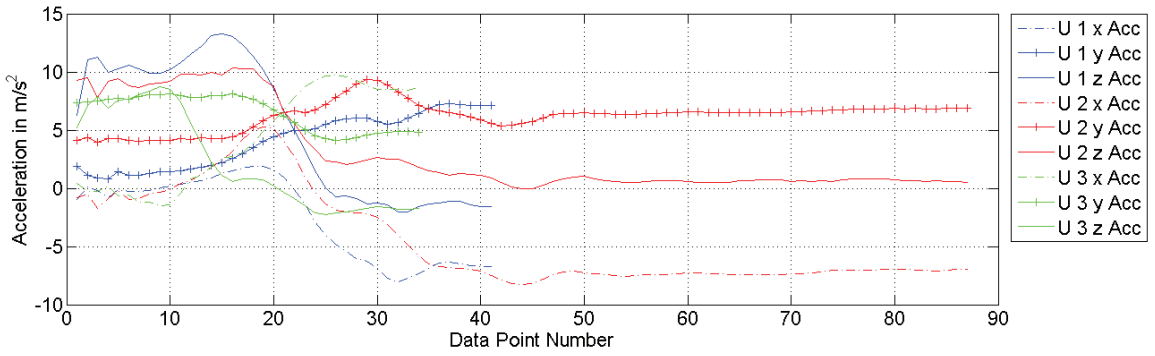
For the motion-based identity verification, our main contributions are: (1) proposed two verification methods, Statistical Method and Trajectory Reconstruction Method, to extract specific features and verify user's identity; (2) collected a multi-session MDP motion database; and (3) provided evidence that verification performance can be affected by the user body movements, such as walking.

(a) (b) (c) (d)

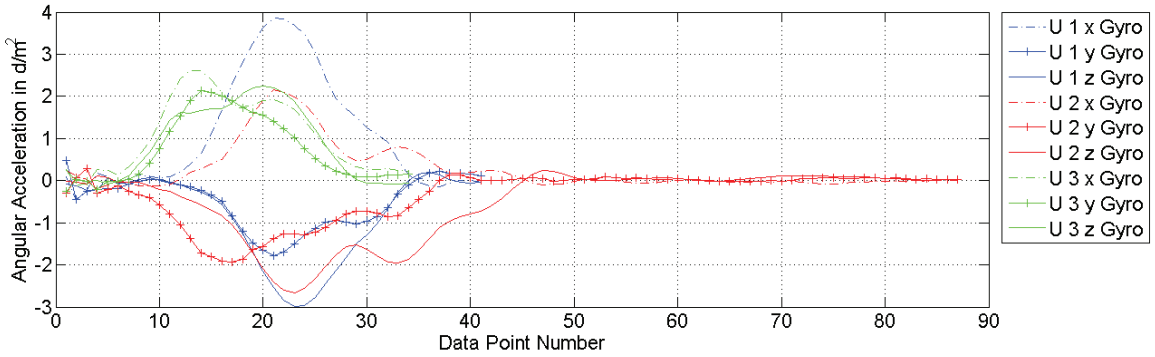


(e)

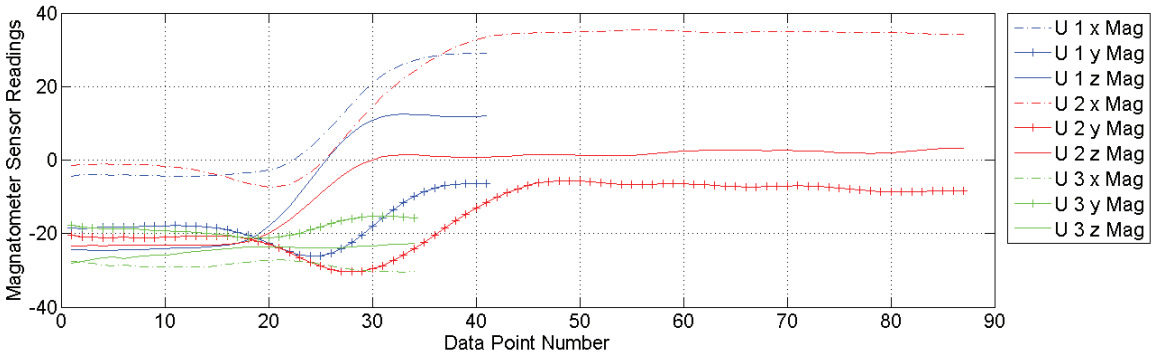
Figure 5.1: Illustration of the feasibility of distinguishing user's identity employing MDP motion, (a), (b), and (c) are MDP traces from 3 right-handed user, (d) is a trace from a left-handed user, (e) is a comparison among the 4 users' traces.



(a) Accelerometer data



(b) Gyroscope data



(c) Magetometer data

Figure 5.2: Illustration of the feasibility of distinguishing 3 user's identity using statistic features. Three smoothed 3 users' data are represented respectively by red, blue and green curves. U 1, U 2, and U 3 represent three different users, and Acc, Gyyo and Mag respectively denotes Accelerometer, Gyroscope, and Magnetometer

5.1 Evidence of Unique Biometric Qualities

When users pick up a mobile-device, they typically grab the phone from a surface or a pocket, raise the arm and hold the phone near the ear. This process is user specific since the curvature of the movement trajectory curve in 3D space is affected by the arm length and upper body morphology, the rotation of the phone is affected by the muscles near the wrist and the speed is affected by the arm muscle behavior. In such a way, the device motion during a specific task corresponds to the physiological and behavioral biometric gleaned from users' upper body morphology and muscle movement of the arm. Fig. 5.1 shows the difference between MDP motions for four people.

In Fig. 5.2, we demonstrate the smoothed accelerometer, gyroscope, and magnetometer data from three different users. The sensor data varies not only in duration, but also in the numerical range along the curves. Consequently, after normalization and segmentation, we can extract features including duration time, mean, min value, max value and standard deviation. And based on these features, we can perform user-identity management.

Furthermore, in Fig. 5.3, we show reconstructed trajectories from two users, and explicit differences can be found from two users' trajectories. The two trajectories share the same starting points because when we transfer the trajectories from local coordinate system to world coordinate system, we initialize the first position as coordinate origin.

To record a mobile picking up motion movement, we employ motion-sensors

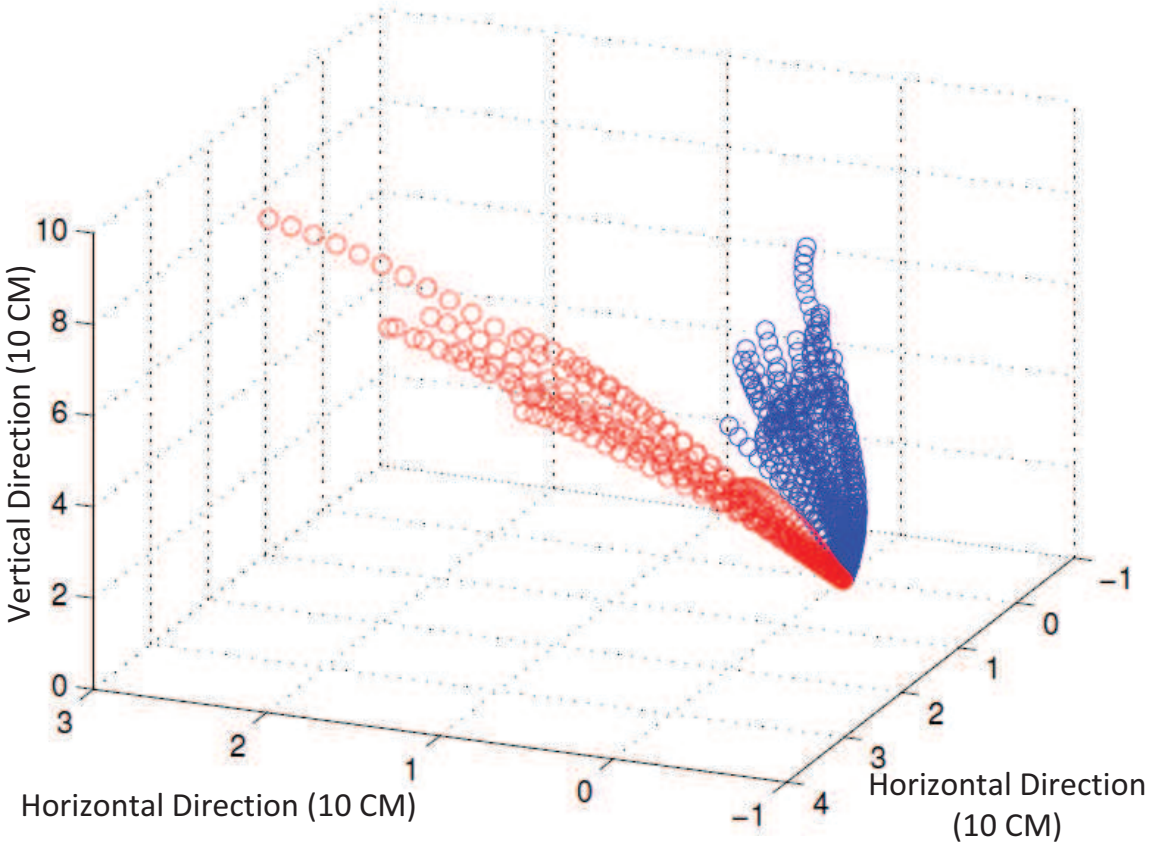
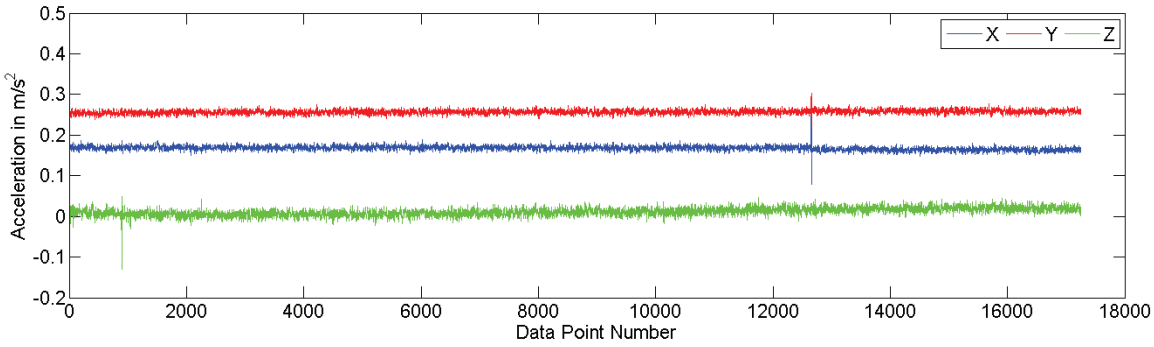
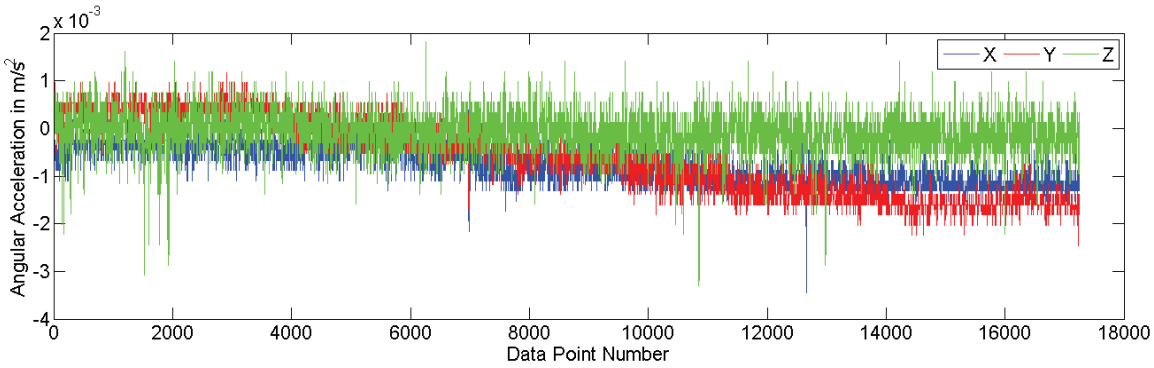


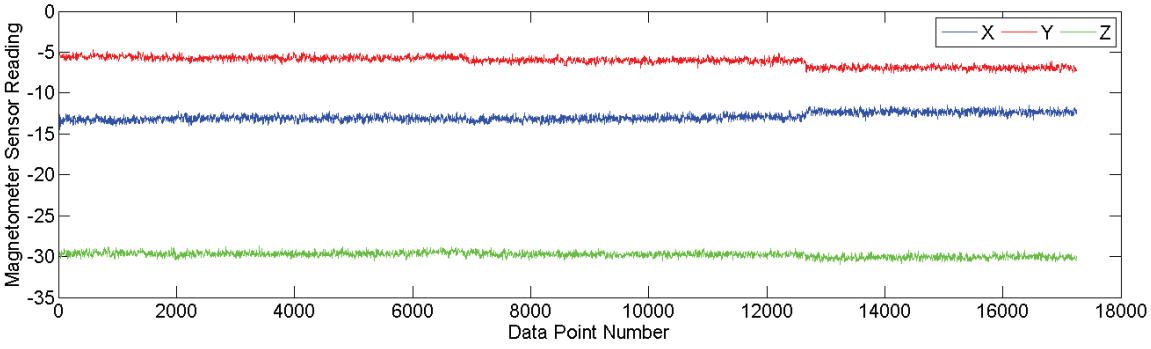
Figure 5.3: Illustration of the feasibility of distinguishing 2 user's identity using trajectory reconstruction method. The red and blue trajectories respectively represents two users 10 MDP motion. We aligned them at the same point in a 3D space and plot them.



(a) Accelerometer raw data



(b) Gyroscope raw data



(c) Magetometer raw data

Figure 5.4: Collected 9-degree raw data from 3 types of mobile sensors when a mobile device is stationary on a table. The acceleration data of z-axis has already minus a 9.8(the gravity impact)

such as 3-degree accelerometer, 3-degree gyroscope and 3-degree magnetometer that are integrated with mobile devices. By employing the 9-degree sensor data, from each mobile picking up motion, we can capture the 3-axes acceleration, 3-axes ambient geomagnetic field, the 3-axes angular velocity for the device coordinates and corresponding time stamp information. In this section, we show the explicit differences of mobile picking up motion among different users caused by arm geometry, mobile use habit and muscle behaviour. Further, based on the data collected, we introduce two methods, a Statistical Method and a Trajectory Reconstruction Method, to exploit the specific motion as a novel biometric modality.

5.2 Statistical Method

Different sensor data can reflect different motion features. Accelerometer data can be used to depict the motion speed and direction change, gyroscope data and magnetometer data show the rotation movement of wrist and the lift movement of elbow and arm. Because the characteristics of each person's motion varies, by extracting and employing the statistic features from the sensor data, we can perform user-identity management.

5.2.1 Statistic Features

Fig. 5.4 shows some raw data collected from the 9-degree sensors. Even if the mobile is fully stationary on the table, obvious jittering noises still exist in the sensor

data. We can reduce the effect of jittering noise by scaling down (x, y, z) readings by using a moving-average filter of span L and factor M. In our experiment, the M is set as 0.2, while L is set as 5. For each sensor reading element v_i in sensor reading vectors, $x, y, andz$, and n is the length of vector x, y, and z, its smoothed value v_i' will be calculated using the following equation:

$$v_i' = \sum_{j=i-2}^{i+2} round[M * v_j], i \in \{3, ..., n-2.\} \tag{5.1}$$

After smoothing the raw data, we segment each motion trajectory into n segments. For each segment, we extract its duration time, mean value, variance, and standard derivation as features for evaluation, and $28*n$ features in total $28*n$ features(1 time duration feature, and respectively 9 features for the mean, standard deviation, and the value domain size of the 9-degree sensors data in a segment). And the features are defined in Table. 5.1.

5.2.2 Statistic Based Verification Algorithms

We evaluate the performance of our system using Support Vector Machine (SVM) [17] algorithm. SVM classifiers are characterized by a decision surface that can be written as a hyper-plane in a high dimensional space $\mathfrak{H}$. The mapping between the input space and $\mathfrak{H}$ is handled implicitly by means of a kernel function $\mathfrak{K}$, which is a (generally nonlinear) symmetric scalar function of two input vectors. The decision function can then be written as,

Feature	Definition
Time duration	The time duration of a MDP motion. Its value equals to the value of the timestamp at the end of the MDP motion minus the timestamp at the beginning of the MDP motion.
Mean	The mean value of a sequence of sensor reading in one degree. For each degree of sensor reading, one mean feature is extracted, so 9 mean features in total.
Standard deviation	The standard deviation of a sequence of sensor reading in one degree. For each degree of sensor reading, one standard deviation feature is extracted, so 9 standard deviation features in total.
Value domain size	The value domain size of a sequence of sensor reading in one degree. For each degree of sensor reading, one value domain size feature is extracted, so 9 value domain size features in total.

Table 5.1: Features definition

$$\mathfrak{f}(\mathbf{v}) = \sum_{j} \alpha_j y_j \mathfrak{K}(\mathbf{s_j}, \mathbf{v}) + b \lesseqgtr 0 \tag{5.2}$$

where $\mathbf{v}$ is the input to be classified. The support vectors $\mathbf{s_j}$ constitute a subset of the training data which is determined through an optimization process. The optimization also defines the weight α_i; the y_j are constant and have a value $+1$ for the support vectors of the 'accept' class, -1 for those of the 'reject' class. Finally, b is fixed so that the hyper-plane in $\mathfrak{H}$ cuts exactly halfway between the closest training examples of the two antagonistic classes. This choice is optimal provided that the training set is equally descriptive of both populations. In the case that one class is poorly represented in the training set, we might want to compensate for that by shifting the hyper-plane further away from its support vectors.

5.3 Trajectory-Reconstruction Method

As explained by the Section 2.4, the motion trajectory varies from person to person for congenital physical and acquired habits, which could provides effective biometric features for user verification. To verify a user's identity by trajectory based method, two following steps are required: (i) reconstruct the motion trajectory; and (ii) compute the distance of different trajectory using different distance functions and perform verification.

5.3.1 Trajectory Reconstruction

Due to Einstein's principle of equivalence [40], the collected acceleration data includes both the impact of motion acceleration and gravity acceleration. Thus, we have to first use a low pass filter to isolate the contribution of the force of gravity in 3 axes as in Eq.5.3. The α is calculated as $t/(t+dT)$, where t is the low-pass filter's time-constant, and the dT is the sampling rate. The force of gravity is substracted from the acceleration as shown in Eq.5.4 to calculate the linear acceleration.

$$gravity_i = \alpha * gravity_i + (1-\alpha) * acc_i \tag{5.3}$$

$$linacc_i = acc_i - gravity_i \tag{5.4}$$

where acc_i stands for acceleration raw data on i axis, and $linacc_i$ stands for linear acceleration on i axis.

Because all the 9-degree sensor data we collected are in the device's local coordinates, it should be transferred into a canonical world coordinate system for comparison. The rotation matrix RM is employed and formulated as,

$$RM = \begin{bmatrix} H_x & H_y & H_z \\ M_x & M_y & M_z \\ A_x & A_y & A_z \end{bmatrix}$$

H_i, M_i, and A_i are calculated in Eq.5.5, where A_i stands for the gravity acceleration of i axis, E_i stands for the geomagnetic field of i axis, and i+1 and i+2 stand for the next and next next axis(for example, i+1 of y is z, i+2 of y is x).

$$\begin{aligned} H_i &= E_{i+1} * A_{i+2} - E_{i+2} * A_{i+1} \\ invH &= 1.0/\sqrt{H_x^2 + H_y^2 + H_z^2} \\ H_i &= H_i * invH \\ invA &= 1.0/\sqrt{A_x^2 + A_y^2 + A_z^2} \\ A_i &= A_i * invA \\ M_i &= A_{i+1} * H_{i+2} - A_{i+2} * H_{i+1} \end{aligned} \tag{5.5}$$

As long as we have the rotation matrix, we can calculate the acceleration along the world axis(WA) by multiply the acceleration along mobile axis(MA) with rotation matrix RM:

$$WA = RM * MA \tag{5.6}$$

However, as mentioned before, the noise of the hardware sensors is not trivial and could severely affect the correctness of the trajectory reconstruction. In order to reconstruct the motion trajectory correctly, the noise of the raw data should be eliminated. Here we adopt Kalman Filter [68] to solve the problem. We formulate our system model and correction equation as Eq.5.7 and Eq.5.8.

$$x_{t+1} = A * x_t + W_t \tag{5.7}$$

where $x_i = [x_p, x_v, x_a, y_p, y_v, y_a, z_p, z_v, z_a]^T$, $p(W) \sim N(0, Q)$ and

$$A = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix}, m = \begin{bmatrix} 1 & T & 0.5T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}$$

$$\hat{x}_{t+1} \quad = A * x_{t+1} + K_t * (M_{k+1} - H * x_{t+1}) \tag{5.8}$$

where

$$H = \begin{bmatrix} n & 0 & 0 \\ 0 & n & 0 \\ 0 & 0 & n \end{bmatrix}, n = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(x_p, y_p, z_p) is the location of the mobile device in world coordinate system, (x_v, y_v, z_v), (x_a, y_a, z_a) are the velocity and acceleration of the mobile movement. A and H are

the process model and measurement model for Kalman filter. T is the time interval between two continuous samplings, i is the index of samplings. W is the process noise, white Gaussian noise with diagonal variance Q. $\hat{x}_{t+1}$ is the corrected system state, M_{k+1} is the measurement of the mobile location. K_t is the Kalman gain calculated.

5.3.2 Trajectory-Distance Function and Decision Rule

After the trajectories are reconstructed, we can employ the Discrete Frèchet Distance [9] to calculate the distance between a pair of polygonal trajectories representing mobile picking up motion sequences. Let $S_1 =< p_1, p_2, ..., p_m >$ and $S_2 =< q_1, q_2, ..., q_n >$ be two polygonal trajectories, $M = (p_i, q_i)$ be an order-preserving complete correspondence between S_1 and S_2, and $d(p, q)$ be a matching cost between p and q, the following distances are defined as,

$$\delta_D(S_1, S_2) = \min_{M}(\max_{(p_i,q_i)\in M} d(p, q)) \tag{5.9}$$

For a training set $T_t = S_1, S_2, ..., S_n$, which comprises n trajectories of one user, we rate each trajectory with a training score $SCORE_S$ as following,

$$SCORE_S(S_i) = \sum_{j\neq i, t_j\in M} \delta_D(S_i, S_j) \tag{5.10}$$

We select the trajectory S_i with the lowest $SCORE_S$ as the model of this specific

%	5 segments	10 segments	15 segments	20 segments	Trajectory
MDPdt1	6.38	5.87	6.13	3.67	5.56
MDPdt2	8.67	5.06	5.32	3.93	8.72
MDPdt3	18.67	17.88	18.73	17.93	29.31
MDPdt1& MDPdt2	10.67	10.86	7.13	6.13	7.09
MDPdt1& MDPdt3	20.31	19.67	19.13	22.31	24.69
MDPdt1& MDPdt2& MDPdt3	20.67	17.32	19.32	18.96	24.34
Mean	14.23	12.78	12.62	12.08	16.62

Table 5.2: EER for different MDP motion dataset under different methods and segmentation settings. 5 segments to 20 segments respectively stands for the result of Statistical Method, while Trajectory stands for the result of Trajectory-Reconstruction Method.

user. When a new trajectory S_t is input, we calculate the similarity score $SCORE_T$ by comparing the input trajectory S_t with the model trajectory S_i,

$$SCORE_T(S_t) = \delta_D(S_i, S_t) \tag{5.11}$$

5.4 Experiments and Results

5.4.1 Data Acquisition

To collect user motion data, we developed an Android program which simultaneously sampled accelerometer, gyroscope and magnetometer at 25Hz using a standard API of Android system. When a data collection procedure begins, a subject is asked to pressing the ”Record” button when she/he is ready to perform the MDP motion. The application will keep recording the raw multi-sensor data from the API

until the button is released when the MDP motion is terminate. For each MDP motion sample, 3-axes acceleration, 3-axes ambient geomagnetic field, 3-axes angular velocity, and corresponding timestamps are captured.

Thirty one participants were recruited for the study, of which 24 were inclined to perform MDP motion with right hand. Their ages ranged from 20 to 45 years old. Out of these, 20 were male. The data acquisition process had 3 sessions and lasted over several weeks. The entire dataset has 930 samples (31 participants * 3 sessions * 10 data samples per session). Sessions differed with each other in terms of date or body movements. In the first session, for each participant, we explained the purpose of the study and how to use the data acquisition application. Then the participant was asked to practice the experiment procedure for 5 to 10 times to ensure that data was correctly collected and the MDP motion was natural. After the participant finished the practice, she/he performed 10 times MDP motion during stationary mode. For robustness evaluation, after one week, the participants were asked to perform the same MDP motion for 10 times as the second session. The third session was similar to the previous two sessions. One difference was that in the third session, participants were asked to perform MDP motion when she/he was walking. There was no restriction on which hand the participants hold the mobile device or how the device was held by the participants, as natural as possible to the participants. The dataset collected in session 1 to session 3 was named respectively as MDPdt1 to MDPdt3. In our study, the MDPdt1 was mainly utilized for feature evaluation and authentication system design. MDPdt2 and MDPdt3 were mainly utilized for investigating robustness of the implemented motion

based authentication across time. Equal Error Rate (EER) was used for evaluation purpose, which can achieve balance between False Acceptance Rate(FAR) and False Rejection Rate(FRR).

5.4.2 Experimental Result

Table 5.4 depicts the EER of authentication when different methods and segmentation settings are applied on different MDP motion dataset. For intra-session result using SM, verification performance of MDPdt1 and MDPdt2 demonstrated competitive results with EER respectively of 3.67% and 3.93%, while the performance of MDPdt3, with a EER of 16.88%. The result of MDPdt1 and MDPdt2 are acceptable, however the result of MDPdt3 is not encouraging. Such situation could be caused by the extra body movement in MDPdt3. To test the real world performance of our methods, in MDPdt3, we did not limit the walking pattern of the subjects, which means uniform motion, variable motion, and changing directions are all acceptable. Due to Einstein's principle of equivalence, the accelerometer cannot separate the MDP motion acceleration with the body movement. The accelerometer data is not only useless for verification but also would cause negative impact on the authentication for its noisy and incorrect features. This may also explains why the performance of MDPdt3, unlike the result of MDPdt1 and MDPdt2, would not increase as the number of segmentation grows: detailed segmentation features do not provide extra useful features but noise.

Because the TRM is based on the Discrete Frèchet Distance of reconstructed

traces, the noise of acceleration caused even severe impact on this method (the EER of MDPdt3 increased to almost 30%). Meanwhile, the performance of TRM is equally good but a slightly declined comparing to SM in the intra-session result of MDPdt1 and MDPdt2. The declination may be caused by the Integrated Error and Calculation Error when reconstructing the trajectory. Integrated Error mainly comes from two sources, (1) integration time, and the (2) hardware drift. For the integration time, the longer it is, the higher the error is. In our case, the integration time is fixed and it is determined by the sampling rate ($1s/25Hz = 40ms$). For the hardware drift, even if we employed Kalman filter to remove the drift and noise, there might still remains minor noises. And the minor noises of acceleration and velocity are amplified by the integration process. Calculation Error comes from the complex matrix and trigonometric calculations explained by the Section 5.3.1. All the aforementioned errors might lead to trajectory distortion and hence reduce the accuracy the proposed method.

From the inter-session result employing SM, we can see a slightly decline on EER in the combination of mixed MDPdt1& MDPdt2 comparing to the intra-session result of separate MDPdt1 and MDPdt2. Behavioral inertia may contribute to this phenomenon. In an intra-session data collection (*e.g.,* MDPdt1 or MDPdt2), the subjects is inclined to repeat her/his first MDP motion, which would lead a verification performance enhancement. While for the inter-session of MDPdt1& MDPdt2, because a week has passed, the usage habit and upper body morphology will play a more important role than the behavioral inertia. The positive verification performance of 6.13% EER demonstrates the proposed method is time constant. The

verification performance on the mixed dataset of MDPdt1& MDPdt3 and MDPdt1& MDPdt2& MDPdt3 still have the same acceleration noise problem with intra-session result. Since the accuracy enhancement effect of behavioral inertia does not exist in the inter-session experiment, the performance is even worse than the intra-session result with MDPdt3.

Similar results can also be found in the TRM in column 6, Table 5.4. The EER of mixed dataset of MDPdt1& MDPdt3 and MDPdt1& MDPdt2& MDPdt3 are 24.69% and 24.34%. As explained, the reconstructed trajectory of stationary status and walking status varies for the acceleration impact, verification employing Trajectory-Reconstruction Method is not feasible since the EER is already as high as 24%.

To verify that the performance change related to MDPdt3 was caused by acceleration, we conducted experiment without accelerometer data and the result is shown in Table 5.3. The results in Table 5.3 are based on 6-degree sensor data, including 3-axes ambient geomagnetic field, the 3-axes angular velocity. The results suggest that although accelerometer data was removed, the performance actually improved rather than declined. Compared with stationary session results, the performance based on data of session 3 is not as good as the first two sessions but still acceptable, which compliments TPM's merit.

%	MDPdt1& MDPdt3	MDPdt1& MDPdt2& MDPdt3
5 segments	13.24	12.35
10 segments	13.67	13.67
15 segments	11.35	9.36
20 segments	10.27	9.62
Mean	12.13	11.25

Table 5.3: EER for different MDP motion datasets without accelerometer data under and different segmentation settings.

Chapter 6

An Integration of Context, Sensors and TrustZone

We take advantage of previous research results and designed an integrated approach and framework, named as IdentityTracker for local and Secure Session Service for remote service. In this Chapter, we first show its system overview, and its local usage. Then we demonstrate its ability of solving identity management problems during remote services, such as online payment.

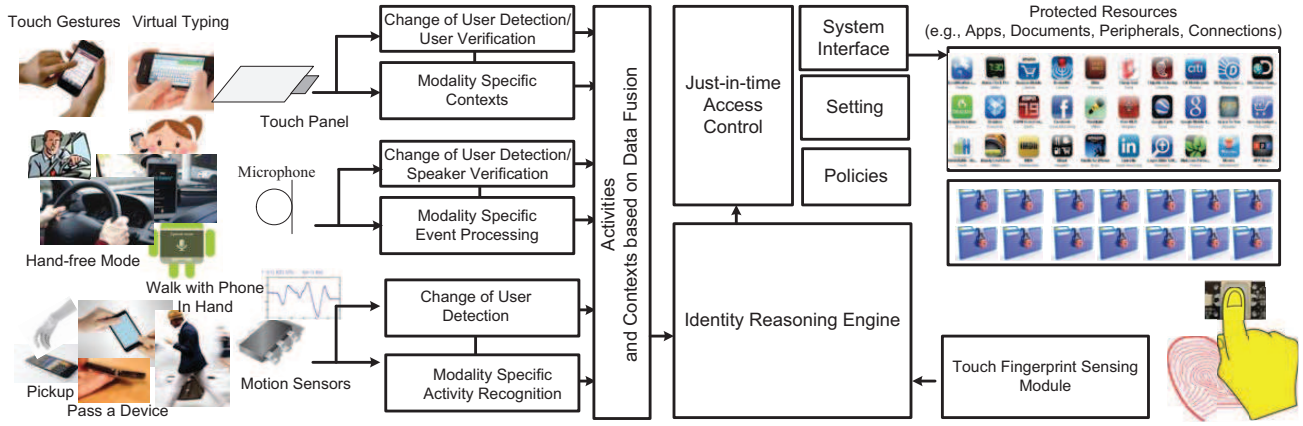


Figure 6.1: Design of IdentityTracker

6.1 System Overview

Fig. 6.1 dipicts the high-level architecture overview of IdentityTracker. There are three main components in the IdentityTracker framework: A Touch Fingerprint Sensing Module, which employs the fingerprint sensor deployed on new-generation smartphones, *i.e.,* iPhone 6 and Galaxy S5, to identify the current users' identity (by fingerprint verification); An app-level Just-in-time Access Control that manages the access permissions based on configurations of the policy and the current user-identity; And most importantly, a Fine-grained Activity Recognition Module that employs touch, voice and motion-sensors on the smartphone devices to detect device-leaving-hand events and monitor user-identity changes, and an Identity-Confirmation Module that confirms user's identity using touch and speech-based user verification approaches. In normal-usage scenarios, when a user unlocks the device with his/her fingerprint, the Touch Fingerprint Sensing Module detects and logs the user's identity. Once the identity is recognized, the Fine-grained

Activity Recognition Module continues tracking the touchscreen usage data, voice input data and the motion-sensor data to monitor and detect user-identity changes. The Identity-Confirmation Module will keep tracking and confirming user's identity using extracted input features. At anytime the current user of the smartphone device may want to access a mobile application. The app-level Just-in-time Access Control will check the current identity of the smartphone user as well as the policy of the mobile application. If an application is not locked, no matter what identity (of the current user), he/she can access it. Otherwise, the Just-in-time Access Control will react based on the identity of the current user: block the application if the current user is a guest or grant access if it is the smartphone owner.

6.1.1 Fine-grained Activity Recognition Module

The Touch Fingerprint Sensing Module and the Just-in-time Access Control are mature technologies on smartphone devices. The highlight and key point of Identity-Tracker is the Fine-grained Activity Recognition Module and the Identity-Confirmation Module. To detect the device-leaving-hand events, we first define a set of subtle gestures and their corresponding context user statuses as listed in Fig. 6.2. Since we are solving identity-changing problems in the post-login stage, we only considered the device while in an unlocked state. Essentially, there are four statuses when the smartphone device is in unlock state, which are respectively: On Table, Use by Left Hand (of a user), Use by Right Hand (of a user), and Use by Both Hands (of a user). There are four subtle gestures between these four status that

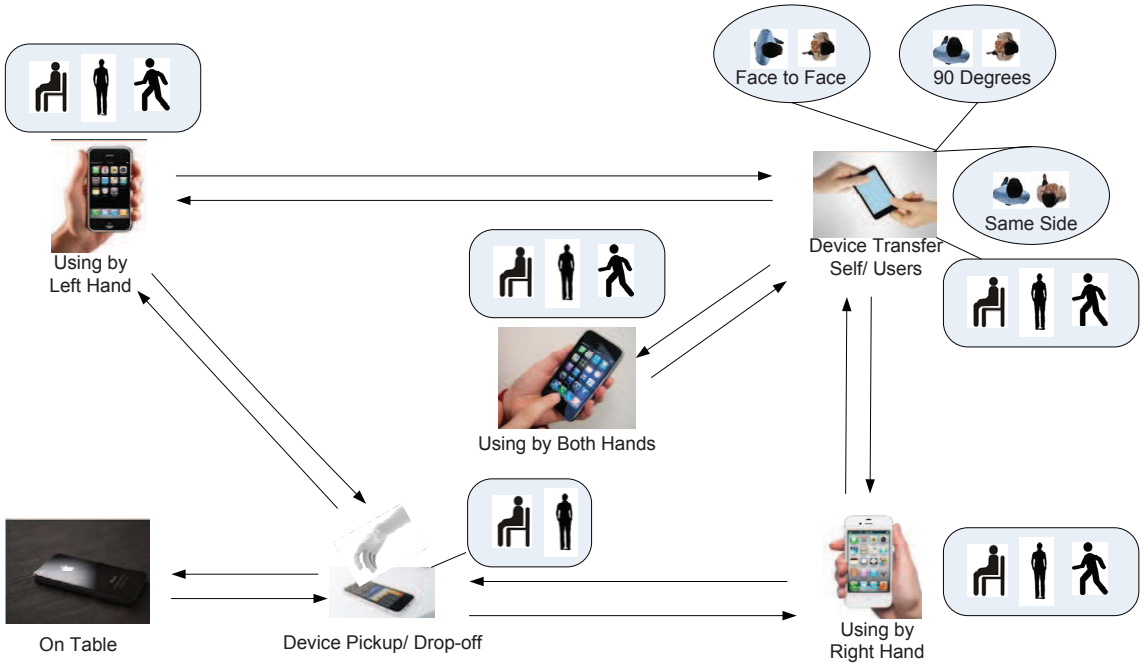


Figure 6.2: Subtle gestures for leaving-hand-events detection

trigger device-leaving-hand events, which respectively are Device Pickup from table, Device Drop-off to table, Device Transfer between the same user's hands, and Device Transfer between different users' hands. Concurrently, we need to take the users' status into consideration since the motion-sensor reading may be affected by different user statuses. During normal usage, there are three main user statuses: sitting, standing, and walking. While during Device-Transfer events between different users, users may have different relative positions(*i.e.,* Face to Face, 90 Degrees or in the Same Side).

To analyze the aformentioned concepts, touch and motion data are processed separately and then combined to predict the status or subtle gestures of the device. IdentityTracker extracts touch trace information, including touch point location,

angle and length, contact size, and speed information to analyze which hand the user is using the device with. Similar to most activity recognition works, IdentityTracker also employs motion-sensors, such as the accelerometer and gyroscope, to detect photo motion activities. The collected motion-sensor data is pre-processed in frequency domain and value domain with a sliding window size of 16 sensor readings. By employing SVM on the extracted features, Holding the Device, On Table, or Device Transfer may be detected with ease in most cases. However, there are some complicated scenarios, such as walking and Device Transfer between different a user's own hands. To solve the subtle gesture recognition in these complicated scenarios, we can leverage those more accurate predictions (*i.e.,* On Table, Using with Right Hand, or context user status, such as walking) combined with the transition map(Fig. 6.2) to analyze those hard to detect subtle gestures. In the mean time, IdentityTracker can also utilize touch data to filter out some misclassified Device Transfer events (Since there is a touch event on the touchscreen, it is not possible that the device is transferring, or the user cannot transfer the device from right hand to right hand).

6.1.2 Identity-Confirmation Module

Fig. 6.3 shows the high-level architecture overview for Identity-Confirmation Module. The Identity-Confirmation Module is mainly composed of two components: a Touch-Verification Function and a Speech-Verification Function. Touch-Verification Function and Speech-Verification Function respectively analyze all user's inputs on the smartphone device, including the touch inputs and speech inputs, and confirms

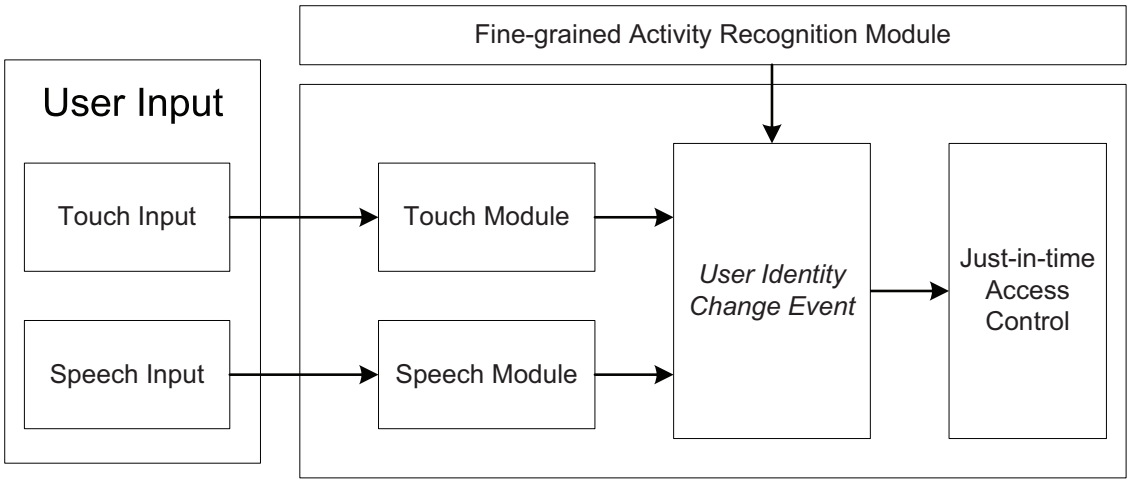


Figure 6.3: Design of IdentityValidator

the result from Fine-grained Activity Recognition Module. If the change of user-identity change event detected by the Fine-grained Activity Recognition Module is confirmed, the Identity-Confirmation Module will communicate with the Just-in-time Access Control. The Just-in-time Access Control then react to the current user based on the current user's identity.

6.1.2.1 Touch-Verification Function

Touch-Verification Function (Fig. 6.4) collects the touch gestures input data and running application context information from the *Multi-touch Driver* and *Running Application Context Listener* respectively. The collected raw data are then transferred to the *Multi-touch Gesture Engine* for data pre-processing and feature extraction. Then the pre-processed data are combined with the running application context information to generate a *Multi-touch Data Library* consisting of gesture

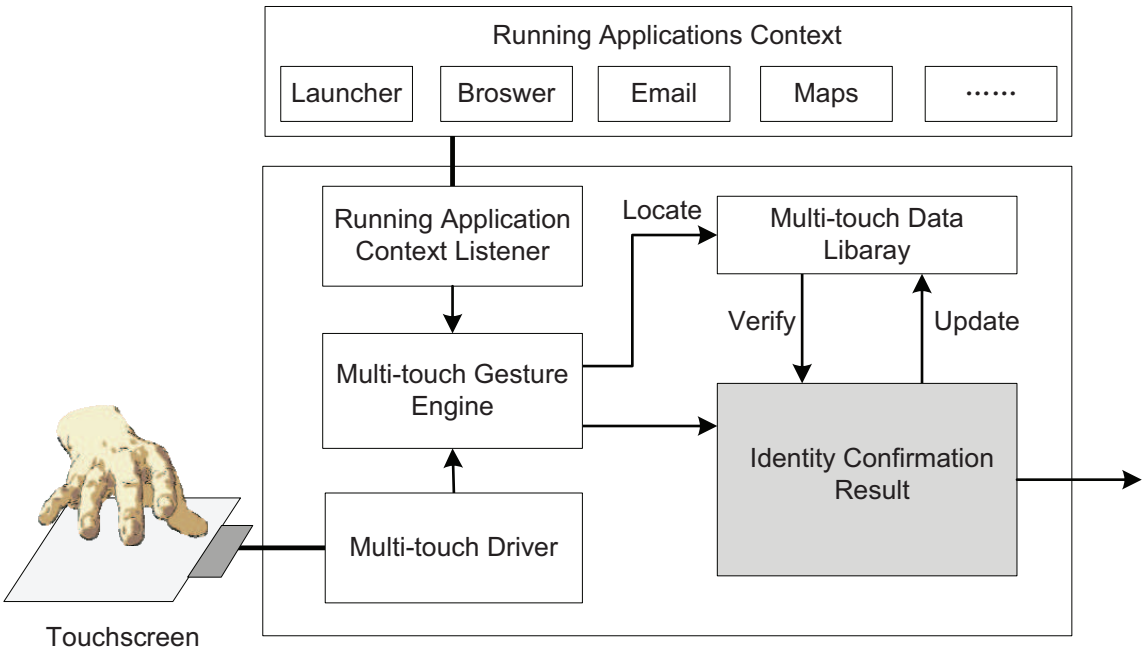


Figure 6.4: Touch Module

templates. The *Touch-Gesture-Based User Authentication Module* compares incoming touch gestures with the templates in the *Multi-touch Data Library* to confirm the current user's identity and send the result to the Just-in-time Access Control.

6.1.2.2 Speech-Verification Function

The way user interacts with smartphones using speech can be categorized into two classes, long conversations for telephone calls or recording, and short commands for speech-based commands and messages. Besides the normal long conversation based speaker-recognition, a highlight of the Speech-Verification Function in IdentityTracker is that it engages in integrating speaker sensing and identity

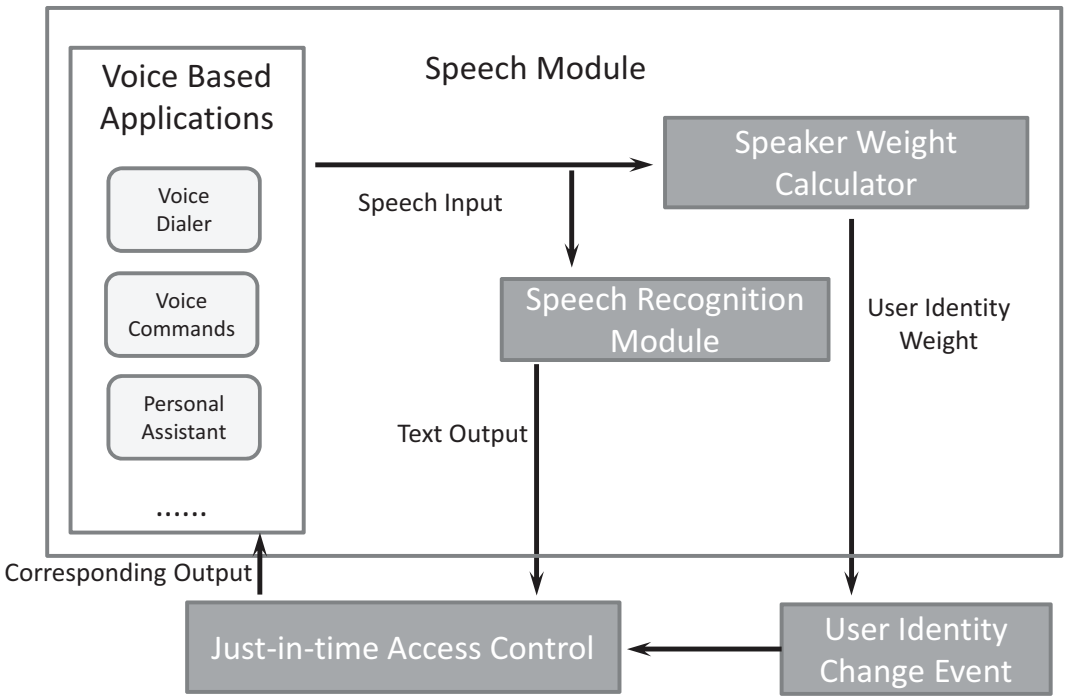


Figure 6.5: Design of Speech Module

management with speech recognition. A diagram of the approach is presented in Fig. 6.5. The approach extends the Android speech recognition API with speaker-recognition and identity management support. The new components include, an application interface that detects context running application and responds to the applications , an identity manager module that controls and enforces policies on whose speech an application should respond to and how to respond, and a speaker-recognition module.

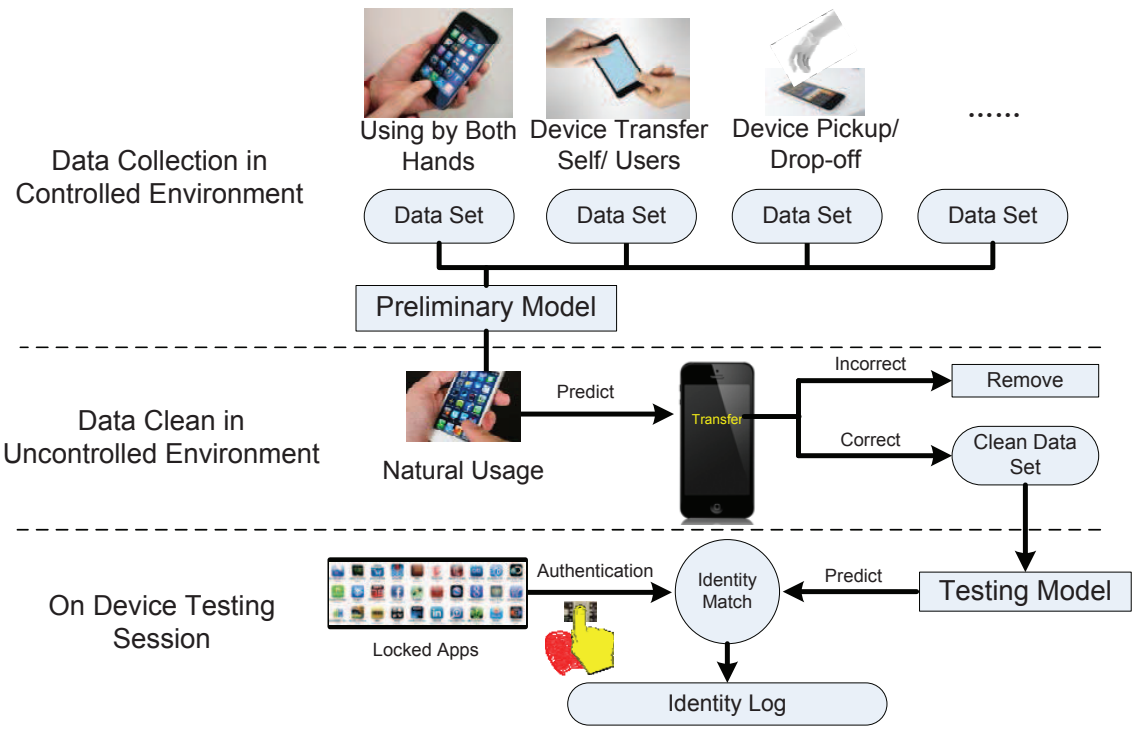
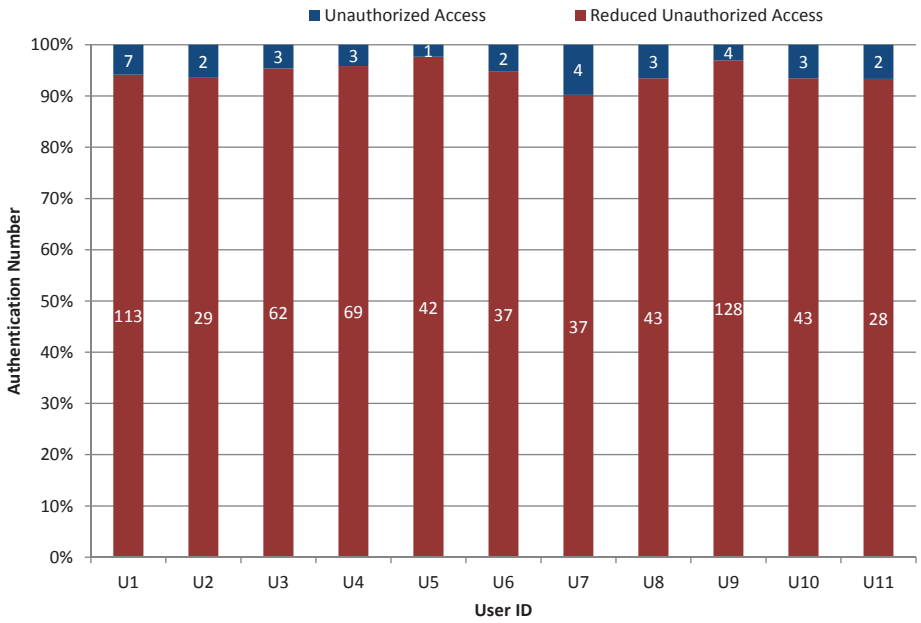


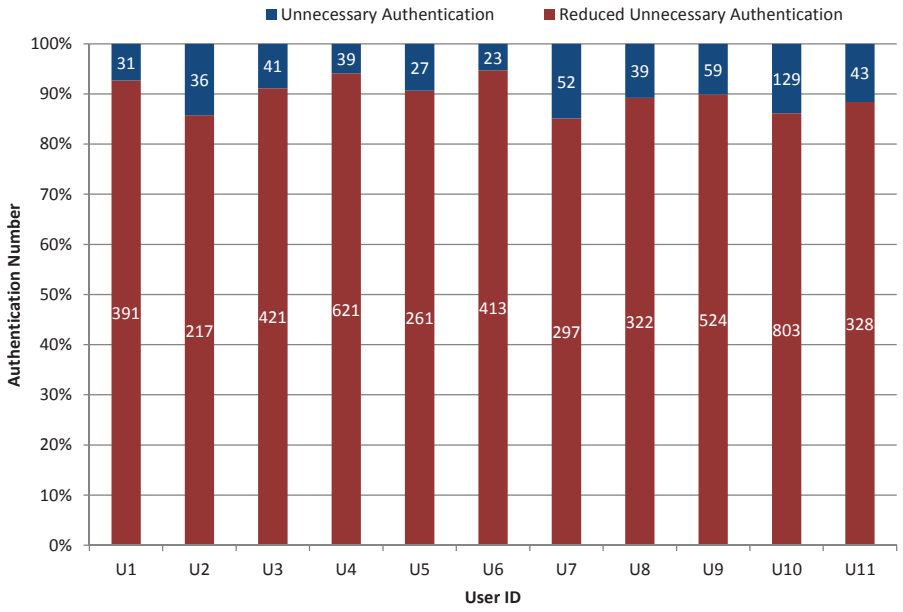
Figure 6.6: Process of the experiments

6.2 Experiment

We implement IdentityTracker as a background service that implicitly collects motion and touchscreen data, and logs the user's identity when an attempt is made to access to a locked application. The IdentityTracker app is installed on 33 smartphone users phones. The experiments consist of three sessions and the process is shown in Fig. 6.6.



(a) Security Evaluation



(b) Usability Evaluation

Figure 6.7: Performance evaluation of IdentityTracker

6.2.1 Data Collection and Data Clean

In the data collection session, IdentityTracker collects a set of phone usage data from users. Users follows the instructions provided by IdentityTracker to perform a set of gestures and operations. Those gestures and operations are predefined, including phone operation on left hand, phone transfer from left hand to right hand, phone operation on right hand,phone operation on both hands, and phone transfer to another user. Although the gestures are predefined by IdentityTracker, users have freedom to perform the gestures in their own ways. The collected data are used to train a preliminary model. The model will be used for the next steps.

After the preliminary model is trained, we use it to classify user gestures and display the results to users. Users are required to provide feedback to the system, e.g., the correctness of the classification results. Users' feedback will be used to improve the primary model and generate a new model for the testing session. Although IdentityTracker attempts to ask users to perform their natural usage during the last data collection session, they may still be affected by the tasks we asked they to perform. If we aim to perform device-leaving-hand events, detection in uncontrolled environment, it needs to first receive a more accurate set of training data. However, since all the subtle gestures listed such as Device Transfers or Device Pickups/Drop-offs happen in a very short time frame and any extra label actions would interfere with the normal gestures. In response, we decided to first train a model based on the data collected in the previous session, and employ it to predict the current status or subtle gesture of the smartphone device and display it on the smartphone device. If the prediction is correct, the data will be recorded

and labeled, otherwise the user can click on the display panel and the data will be labeled as misclassified and will not be used for final model training.

6.2.2 On-Device Testing Session

As long as IdentityTracker acquires the clean data in an uncontrolled environment, it will train a new model based upon this new data set. After we install the new model on the device, IdentityTracker will still authenticate user's identity whenever a locked app is being accessed using fingerprint authentication and logs ground truth user-identity. In the mean time, the testing model will also output a prediction result of current user's identity. The IdentityTracker will match the two user-identity results and log them for further evaluation.

6.3 Local Performance Evaluation

We evaluate the performance of our system in both security and usability aspects: i) How many times has unauthorized app access been reduced in comparison to a mobile system without app-level access control and how many unauthorized app accesses has been granted; ii) How many instances of unnecessary authentication have been reduced for the phone owners in contrast to a strict app-level access control mechanism and how many instances of unnecessary authentication have been requested.

Fig. 6.7(a) shows the identity match log results of the on-device testing session. The red bar represents the number of unauthorized accesses blocked by IdentityTracker, while the blue bar marks the number of unauthorized access IdentityTracker failed to detect. It is clear that in comparison to the mobile system without app-level access control, IdentityTracker greatly reduces unauthorized accesses (above 90% of unauthorized access request were denied) and only very few times was a guest user allowed access to a locked application.

Fig. 6.7(b) shows the usability enhancement results based upon the logged results. In comparison to the mobile system with strict app-level access control that requires authentication every time a user atempts to access to a locked application, IdentityTracker alleviates the user's burden of constant authentication when he/she atempts to access a locked application (themselves). Above 85% of unnecessary authentications may be reduced by IdentityTracker. Although IdentityTracker may introduce a few unnecessary authentication events by falsely detecting a device-leaving-hand event, it still promotes the usability.

6.4 Remote Services

In this section we provide detailed description of the proposed approach for trust zone based mobile payment activity.

Approach overview: The online mobile payment scenario may be abstracted to a tripartite interaction protocol. The three parties included are the *User*, *Smart*

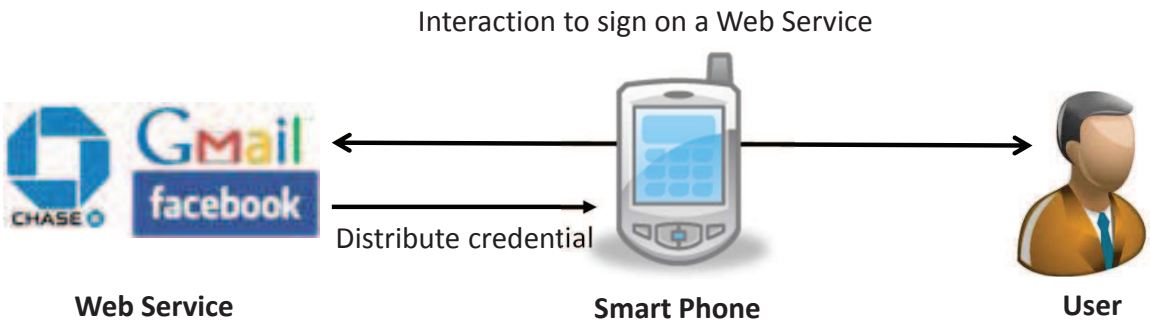


Figure 6.8: A high-level overview of secure online mobile sign on service. Notice that the phone acts as the hub between the user and the server, gathering information from each entity and distributing information back to the web server.

Phone, and *Web Service*. The *User* wants to sign on the *Web Service* using a *Smart Phone*. In response, the *Web Service* issues a credential (e.g., public/private key pair) to the *User*, which is saved in the *Smart Phone* and protected with TrustZone. The *User* submits their biometric information (e.g., fingerprint) to log in to the *Smart Phone* system and the secure session. The mobile device utilizes motion-sensors to detect user-identity changes and determines if the secure session should be terminated or continued. When a new transaction takes place, the *Smart Phone* checks the secure session status and uses saved credentials to authenticate himself to the *Web Service* and complete the transaction.

Detailed description: The proposed system is constrained only by the architecture of the processor used in said device as well as the availability of a fingerprint biometric sensor. The processor used must be an ARMv6KZ or later application profile architecture. The reason for this is that ARM architectures prior to these do not have TrustZone implemented. Since TrustZone is hardware implemented there are no other options for older devices which does not provide hardware support

for it. Once this requirement has been satisfied, the system may be very sensitive or lack some sensitivity as the process relies heavily upon the accuracy of the fingerprint sensor and the technology therein. It is assumed that when using this method, the quality of sensors will match the application i.e., an sensitive application will require and utilize high quality sensors. Please note that this process is session based, meaning, if the user requests three transactions, they will have to verify their fingerprint only one time to begin the session given the user using the device is constant.

As seen in Fig. 6.9, the software process is as follows: First the user will log on to some sensitive apps such as a bank app to pay bills. This process remains the same as is now. However, once the user requests a session, the device will throw an FIQ exception to enter the secure mode. This exception will be handled by the monitor, which verifies that the app has permission to run services in secure mode. Upon validation, the monitor will complete the context switch into secure mode. Now that the service is running inside the secure mode it will not be affected by any malware infection that may be present inside the phone because once the context switch has been made, the process is running independently and completely isolated from everything within the normal mode. The application will now request the user to submit a fingerprint for verification. Because we are not leaving secure mode to collect or process the fingerprint, it is still safe. This, if stolen by hackers, could be detrimental because fingerprints are used in many settings to verify a user. This fingerprint will be processed as is explained in the background section. Once the fingerprint has been processed if the score generated is below a

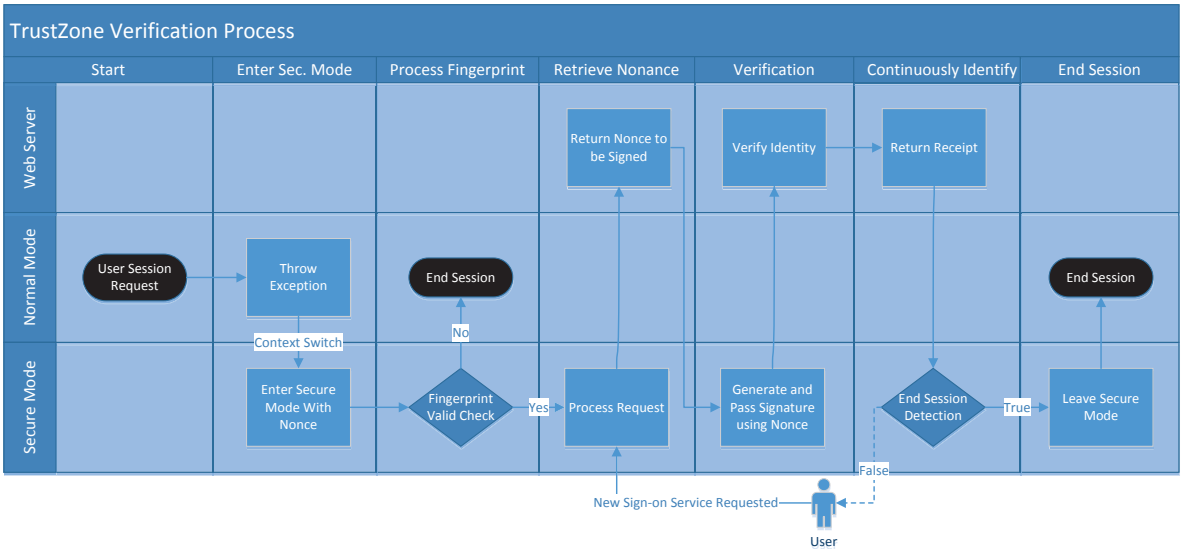


Figure 6.9: A depiction of the authentication process of a sign-on request. Black ovals represent beginning or ending points, diamonds represent decisions, and squares are general processes. The process begins as a request for a session from the user made in normal mode. The system first enters the secure mode via an FIQ exception. The program will verify the user by fingerprint. If incorrect, the process will leave secure mode. If correct, the user is able to sign on the web service. Once a sign on service is requested a nonce is retrieved from the web server. Once received, the process will sign the nonce using the certificate stored in the secure mode and send this back to the web server to complete the sign-on process. While still in session the continuous identity verification will continue until another sign-on service is requested and restart the process at the ”Retrieve Nonce” step. If while waiting the device detects a user-switch event, the session will end automatically or if the user requests the session to end.

set threshold the verification will fail, service will end and the device will be context switched back into normal mode. However, if the fingerprint matching score is above the set threshold, the service will continue inside secure mode. The user is now free to request transactions to be made. The requests will be processed and for each request a nonce will be returned from the bank server. The nonce may contain transaction information such as time, the value of the transaction, and the recipient. Never leaving secure mode, the nonce will be signed using the certificate that is also stored in the secure mode and esnd the signature to the bank server to complete the transaction. Granted the user is the same and the session is continued, the fingerprint will not be required for subsequent transactions. However if the session is ended by either the user or by the process due to detection of a user-switching event, a new session must be started (requiring a fingerprint).

As seen in Fig. 6.10, the hardware process is as follows: Once the nonce has been received and the FIQ exception is thrown, it will be trapped by the monitor which, if a valid request, will carry out the context switch and pass the nonce via a register write. The APB (AXI to Advanced Peripheral Bus Bridge) will then request the I/O Controller enable the fingerprint sensor. Once complete, the processor will generate the score. Assuming the score is above the threshold the processor will generate the signature. The signature will be passed via a direct register write, and sent to bank. Then the bank can verify and complete the transaction.

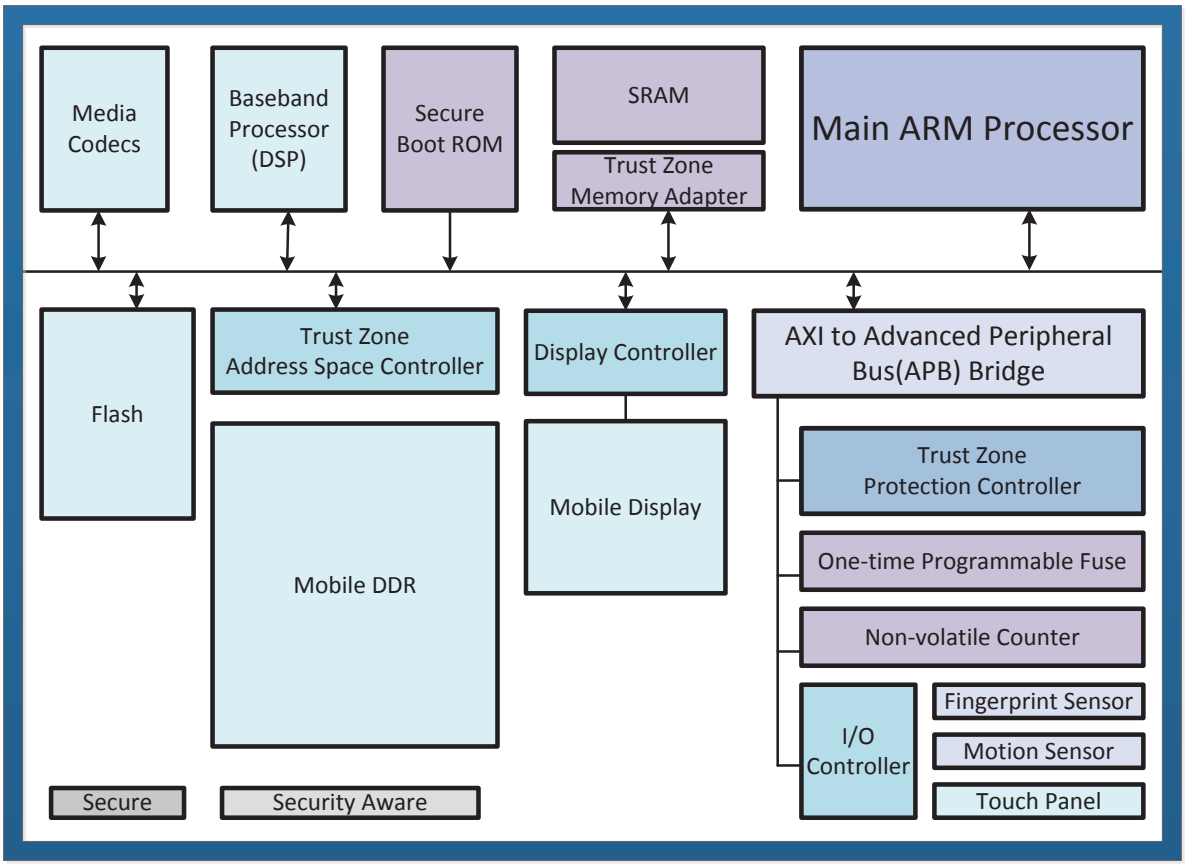


Figure 6.10: The general hardware layout of TrustZone for our design. Note that within the IO Controller we have a fingerprint sensor and touch panel which will be protected by TrustZone.

6.4.1 Unauthorized-Access Accountability Protections

In the case that an unauthorized user attempts to enter a secure session (we are able to detect this when the fingerprint verification fails) we must take steps to prevent further access and deter unauthorized users. Because of the methods currently employed in fingerprint verification forbid fingerprint recording due to privacy concern, we are not able to record the fingerprint directly. However to combat the problem, in this process, when the fingerprint verification fails (which is obviously detectable) the gps coordinates are recorded with a time stamp and a picture taken using the front camera to capture the user. This information is emailed to specified users. In this way, immediate action may be taken to mitigate further and subsequent unauthorized access attempts as well as deal with current issues. This can greatly reduce the fraud that occurs in relation to the device as well as recover lost assests in the case they are able to access the session.

6.4.2 Discussion

Of the advantages of this approach, the most substantial one is the fact that this process uses a three-stage verification that may not be tampered with by any infection located in the normal mode of the mobile device. This is achieved by context switching into the secure mode. Once the nonce has been received from the web server the context switch is completed. Regardless whether infections see the nonce, it does no good without the certificate. So it is ok to possibly expose this

to infections. Since after this time all infections have been isolated to the normal mode context (not in the secure mode). Thus, we may safely trust the secure context now being used. The first stage of verification, which is the fingerprint recognition, is also sensitive so it must be completed in TrustZone as well. Instead of sending this data directly to the bank server and risking fingerprint data stolen by hackers, we, as is the norm, use signature generation to verify the user to the server, which completes the second stage of verification scheme. This certificate is pertinent information and inasmuch must be stored in TrustZone as well. Then, granted the user does not end the session, continuous implicit biometric verification is utilized as well to further protect subsequent transactions within the session (which remains in secure mode as well). This continuous verification will monitor the users actions. In any case that the phone has possibly left the direct possession of the user the session is closed automatically. Whether the release of the device was intended or not. Throughout this whole process, no information is leaked to normal mode.

Besides the inherent protection offered by TrustZone, implicit continuous biometric verification, and fingerprint scanning technologies, there are a few other strong advantages within this approach. Currently, if a password is used, even with TrustZone, the hacker would be able to compromise the mobile device. While they would not be able to retrieve the certificate because of TrustZone, the certificate could be used to verify themselves effectively rendering the system ineffective. However, our approach does not employ these means. Since we are using fingerprint technology, the hacker would either have to have the person with them

whose account they are trying to compromise present (who would naturally not allow this), or have collected a good fingerprint sample and replicate this fingerprint in a manner that is able to trick the fingerprint sensor. Although only few hackers would have this ability. Also it is important to note that since our method uses continuous implicit biometric verification rather than time as a session ending variable, our process can correctly handle phone theft while intra-session cases that would normally result in unauthorized use. Our method would end the session automatically once detecting the phone transfer and in effect render the phone incapable to carry out any subsequent web service sign-on request with a fingerprint.

When an unauthorized user attempted to access/start a secure session in the phone takes in place, the current session, if one is ongoing would be ended and a new fingerprint verification would be required to start another. The Unauthorized Access Accountability Protections would record all data related to the denied access and how many attempts have been made. This, after being emailed to the correct user, would allow the device to be either recovered, or wiped as is per the norm. However since we record not only the face but also the location and time they may be used to apprehend the user and recover the device. This scheme also allows for the use in situations where the device is unknowingly taken and put back. Normally in this case the owner would not be aware of these attempts of that the phone had been stolen or tampered with. With our method this event would be recorded as well.

Scenarios	Definition
Baseline	Baseline scenario with every new function disabled.
Protect-Voice	In this scenario, only voice-based authentication is enabled.
Protect-Touch	In this scenario, only touch-based authentication is enabled.
Protect-Motion	In this scenario, only motion-based device status monitoring is enabled.
Protect-All	In this scenario, all the sensor based approaches are enabled.

Table 6.1: Scenarios in system overhead analysis.

Currently, mobile devices only have the ability to accomplish fingerprint reading. As technology advances, the hardware in phones would increase in sophistication as well. Inasmuch, if finger vein verification was integrated into hardware instead of fingerprint, the process would be much more robust. That said, fingerprints, as previously stated, may be dirty and make verification difficult. Finger vein scanning technology is not affected by the surface of the skin as veins are below the surface [32]. The process is also much more secure, in that as veins are hidden inside the body, there is little risk of forgery or theft [32]. While not infallible, this approach can offer increased security for settings that require it.

6.5 System Overhead Analysis

In this section, we will discuss the computational and energy consumption overheads of the implemented system. The analysis was performed on Nexus 4. For energy consumption analysis, we define different usage scenarios in Table 6.1.

In addition, we defined three usage modes to represent how busy a smartphone

Mode	Definition
Low	In this mode, most of the time the smartphone device is not interacting with the user, and the smartphone is not active longer than 6 minutes per hour on average. The users may use the smartphone to make short phone calls, send messages or read emails.
Medium	In this mode, the smartphone device is active for at least 5 minutes but less than 24 minutes per hour on average. The user can access and use most of the installed apps except games.
High	In this mode, the smartphone is active for longer than 24 minutes per hour on average. The users can access and use all the installed apps.

Table 6.2: Usage modes in overhead analysis.

is. The details are shown in Table 6.2. We conducted experiments based on both different scenarios and the usage modes as defined. The three modes in the baseline scenario served as a baseline for overhead analysis. Overheads incurred by the sensor based authentication were analyzed and compared against the baseline scenario.

The overhead of the implemented system may be attributed to various sensor based user authentication components and the overall security framework. When computing overhead was measured, we used CPU and memory usage. For measuring energy consumption overhead, we used measurements such as impact to battery life, battery usage percentage, and power consumption value as metrics. For each implemented process and module, we collected CPU and memory usage, as well as disk read and write. The details are provided in the following subsections.

Process	Resource	Consumption Level
Recording	None	None
Segmentation	File read, CPU, Memory	Low
Transferring to server	Network	Low
Receiving result	Network	Low
Perform Actions	File Write	Low

Table 6.3: Consumption in voice module process

6.5.1 Speech Analysis

6.5.1.1 Speech Processing Process

The speech-based module is invoked only when the system receives speech command inputs from the user. The speech-based module consumes very little power when the user doesn't interact with the device using speech or when the device is not in hand-free mode.

When a user decides to interact with the device using speech commands, voice data will be recorded. However, the voice recording process does not incur much extra cost since it is supported by the speech recognition application, such as Google voice, or Utter. After voice data is saved as a temporary file on the device, it will be processed and segmented into small chunks. These chunks will be uploaded to the server through Http. After the server finishes speaker verification, the outcome will be sent back to the smartphone. The speech-based module will take proper actions based on the result. The resource usage of each step of the speech processing module is shown in Table 6.3.

6.5.1.2 Computational Overhead

Since resources used by the speech-based module are subject to the frequency of speech-based interactions, we tested the system under different speech interaction scenarios. Three settings were used, which were 5, 10, and 20 speech commands per hour. In high usage mode, our evaluation was based on all of the three settings. For low and medium usage modes, we only considered the setting of 5 speech commands per hour. CPU usage information was collected during the tests. For all the three different usage modes, even in the high usage mode under the setting of 20 speech commands per hour, the CPU usage was still below 0.1%. So we concluded that the speech-based module incur minimal extra CPU usage overhead.

6.5.1.3 Energy Consumption Overhead

For estimating energy consumption overhead, we applied the same settings as in the computational overhead analysis. Fig. 6.11 shows battery life results under different settings. As suggested by the results, the speech-based module doesn't significantly reduce battery life. Battery percentage data for each usage mode is shown in Fig. 6.12, Fig. 6.13, and Fig. 6.14 respectively. For all the cases including high and medium usage modes, energy consumption from speech-based module was below 1% of the total energy consumption. According to the experiment results, one can conclude that the speech-based module does not introduce significant computational or power consumption overhead.

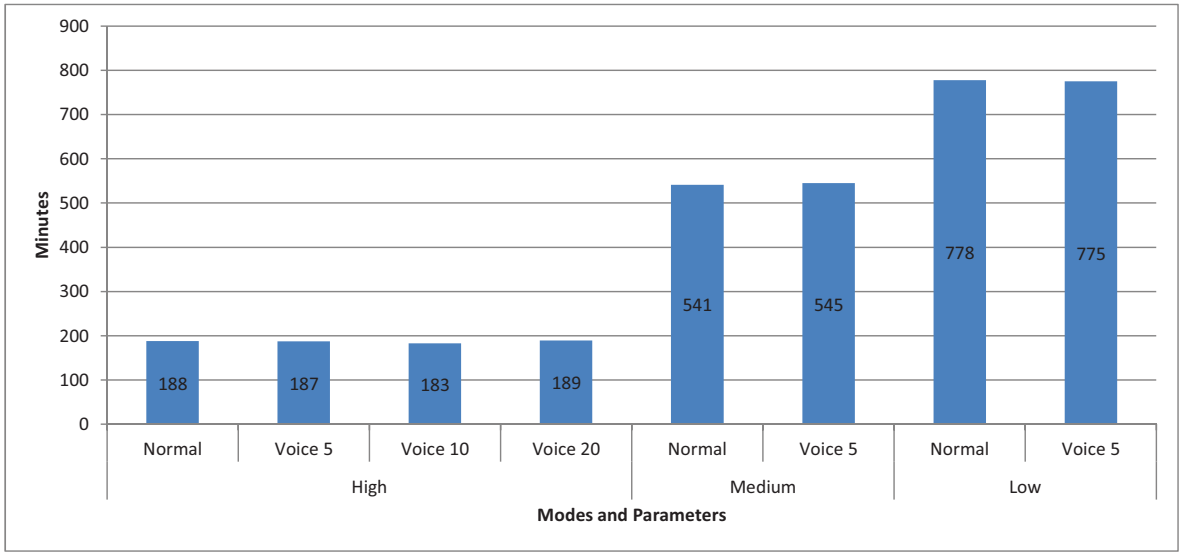


Figure 6.11: The battery life of each usage mode under different settings. Voice 5, Voice 10, and Voice 20 respectively denote the setting of 5, 10, and 20 speech commands per hour.

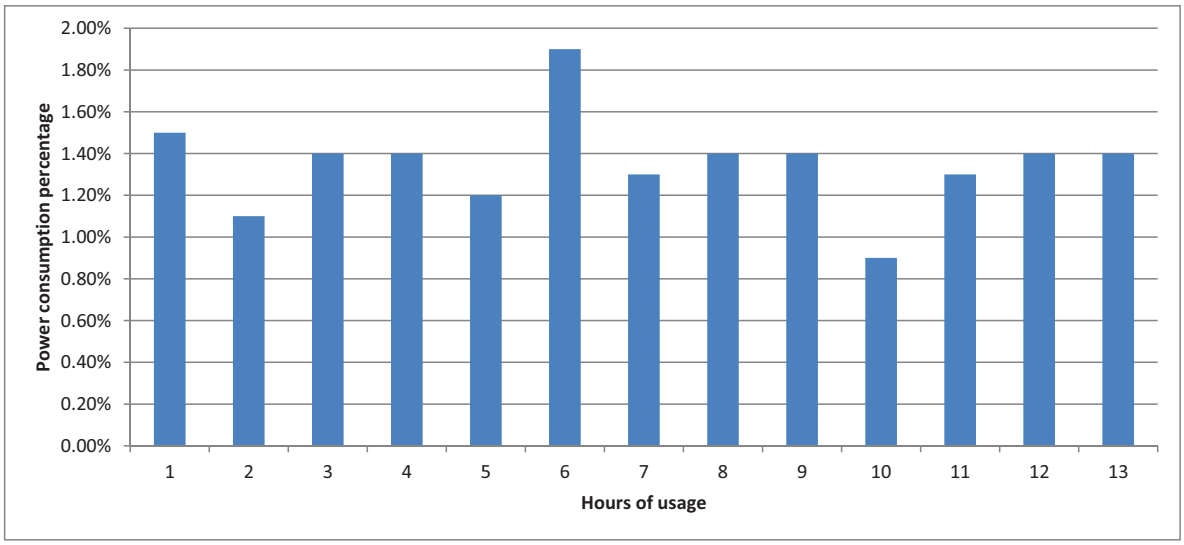


Figure 6.12: Speech-based module power consumption in the low usage mode.

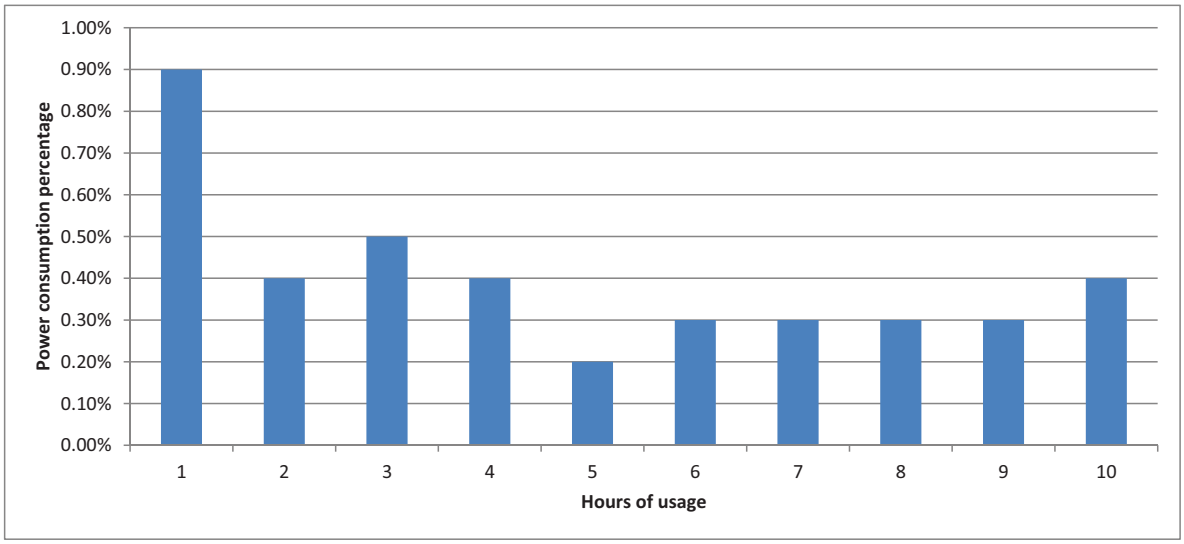


Figure 6.13: Speech-based module power consumption in the medium usage mode.

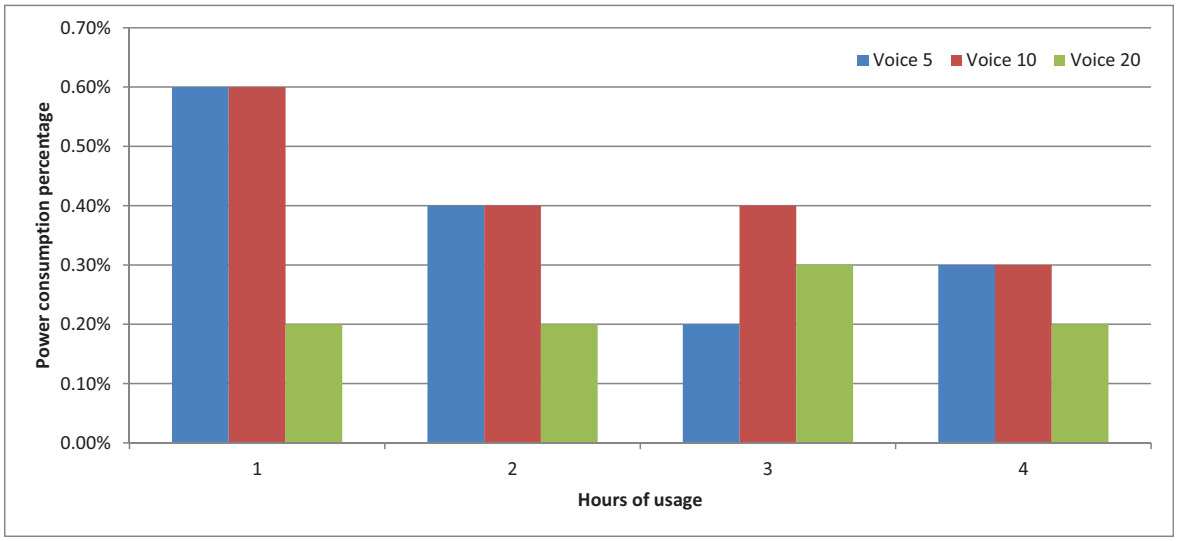


Figure 6.14: Speech-based module power consumption in high usage mode.

Process	Resource	Consumption Level
Waiting for input	CPU	Very low
Input reading	CPU, Memory	Low
Interpreting	CPU, Memory	Low
Sub-template locator	CPU, File read	Low
DTW distance calculation	CPU, File read, Memory	High
Sequential improvemnet	CPU, Memory	Low
Perform Actions	File Write	Low

Table 6.4: Resource overheads in touch-based data analysis.

6.5.2 Touch Analysis

6.5.2.1 Touch Analysis Process

The touch-based user verification and context module is implemented as a background service. When there is no touch interaction or the system is in locked state, the background service will be in idle status, which consumers very litter power. When user interacts with the smartphone using touches, the service will capture touch inputs from the touchsceen sensor. After pre-processing, the background service extracts features such as speed, contact size, curvature from the touch sensor data. A sub-template database locator is used to locate a suitable sub-template database for pattern matching. The pattern matching process executes the dynamic time warping library to calculate the distance between input and the selected templates. In addition, it accumulates the DTW distance for sequential reduction of false rejections. Then the system can act based on the touch analysis outcome. The resource usage of each step is shown in Table 6.4.

Mode	Average	Min	Max
High-Average	2%	1%	3%
Video-Average	2.6%	2%	3%
Medium-Average	4.5%	3%	10%
Low-Average	2.9%	2%	4%
High-Max	12%	8%	2%
Video-Max	17.3%	16%	18%
Medium-Max	19%	11%	47%
Low-Max	19.2%	11%	2%

Table 6.5: CPU usage statistics for the touch-based background service.

6.5.2.2 Computational Overhead

The main computational overhead can be attributed to the DTW distance calculation. Table 6.5 shows the CPU usage statistics. Suffixes Average and Max in Column Mode mean the average and maximum CPU usage in hours. High-Average and High-Max respectively mean average CPU usage and maximum CPU usage in each hour. According to the results, CPU usage on average is very low, below 5%. However, the peak CPU usage is relatively higher sometimes. The peak CPU usage occurs when there is a burst of touch inputs, in which case, touch pattern matching has to be applied to all the input within a relatively short time period. Since this kind of scenario happens only infrequently, the average CPU usage overhead is small.

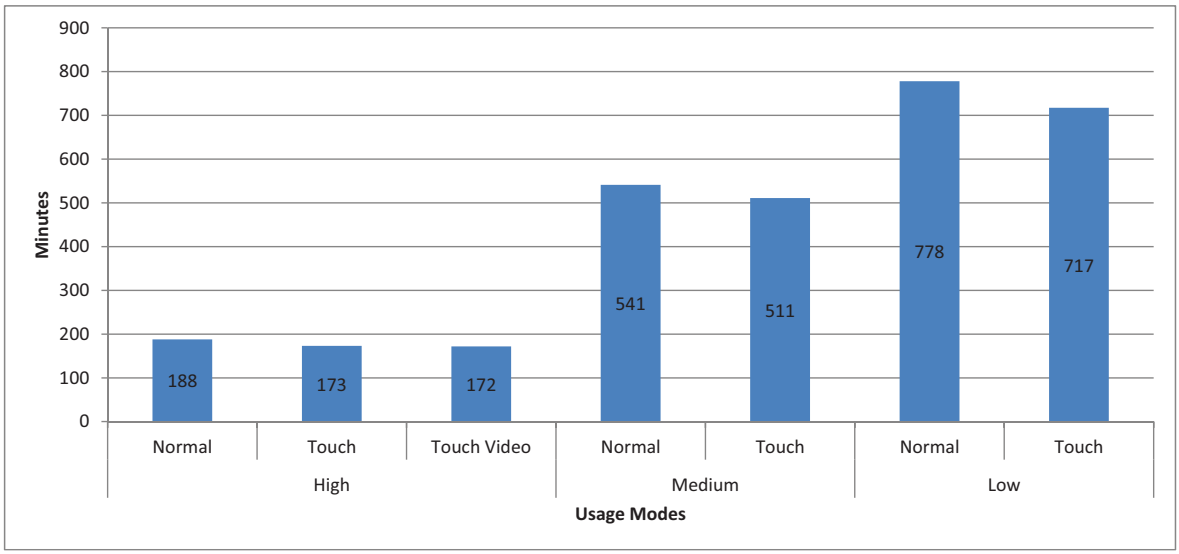


Figure 6.15: Mobile battery life statistics under different usage modes with the touch-based background service enabled or disabled.

6.5.2.3 Power Consumption Overhead

Fig. 6.15 shows battery life statistics under different pre-defined usage modes. The overall impact on battery life was below 10%.

In certain usage scenarios (e.g. watching video, GPS navigation), the mobile device may be in a state of high CPU usage but receive few or even no touch input from the user. We investigated such high usage scenario using video playback as an example. The battery life result is shown in Fig. 6.15. Comparison of power consumption between normal usage mode and video playback mode is shown in Fig. 6.16. As indicated by the results in Fig. 6.16, power consumption overhead for the touch-based background service is correlated with the frequency that a user interacts with the mobile device using touches.

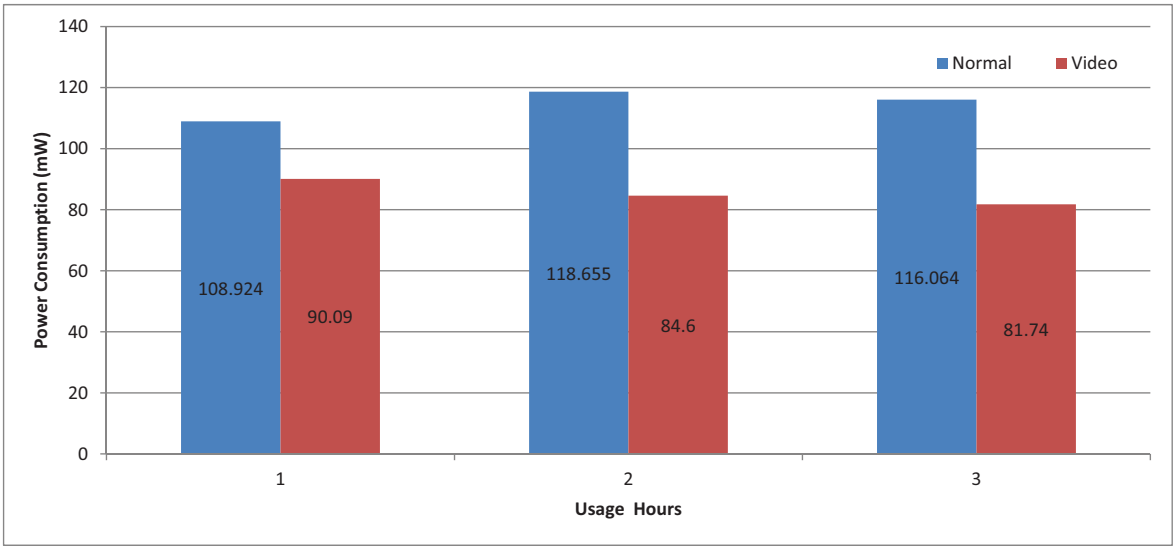


Figure 6.16: Comparison of power consumption between normal touch mode and video mode.

Details of power consumption percentage are shown in Fig. 6.17. In video playback mode, power consumption percentage dropped 2% when compared with the normal touch usage cases. Regardless of the usage modes, power consumption of the touch-based background service was relatively stable. For the high, medium, and low usage mode, power consumption percentage remained around 10%. When a mobile device was in high usage mode (increased power consumption), the impact on power consumption by the touch-based background service is relatively small because its percentage is low. On the other hand, in low usage mode (low overall power consumption), the power consumption percentage attributing to the touch-based background service is higher. However, considering that the overall power consumption is low, the actual energy consumed by the touch-based background service is still acceptable.

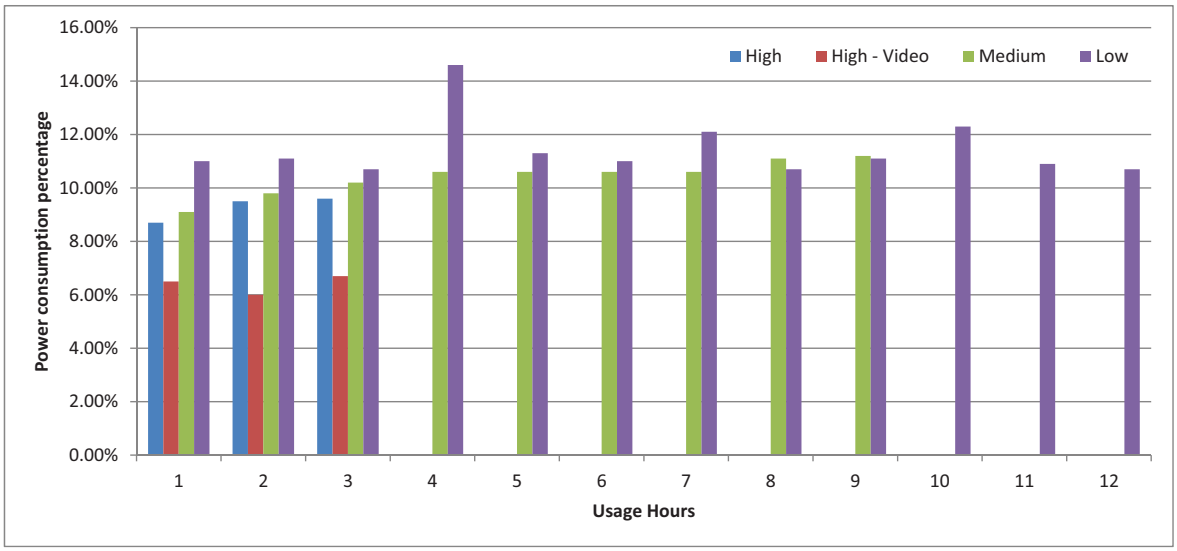


Figure 6.17: Touch module power consumption percentage in each mode

6.5.3 Motion Analysis

6.5.3.1 Motion Analysis Process

The motion sensor based context analysis is also a background service. For situation awareness and device state monitoring, the motion-based service continuously records and analyzes the motion sensor data. The first step is to acquire the sensor readings through system API and write the collected sensor data into a vector. After the vector is filled, the system applies feature extraction to the sampled sensor data according to a pre-defined sliding window. Then the extracted feature vector is matched using a simple decision tree. The system uses the analysis results for monitoring the smartphone state, determining context and detecting switch of user identity. The resource usage of each step is shown in Table 6.6.

Process	Resource	Consumption Level
Recording	CPU, Memory	Medium
Feature extraction	CPU, Memory	Low
Identity change detection	CPU	Low
Perform Actions	File Write	Low

Table 6.6: Resource overheads of the motion-based sensor data analysis service.

Mode	Average	Min	Max
High-Average	1%	1%	1%
Medium-Average	1%	1%	1%
Low-Average	1%	1%	1%
High-Max	3%	3%	3%
Medium-Max	3.2%	3%	5%
Low-Max	3%	3%	3%

Table 6.7: CPU usage percentage of the motion-based sensor data analysis service.

6.5.3.2 Computational Overhead

The CPU usage is mainly caused by feature extraction and decision tree processing. Table 6.7 shows CPU usage statistics. The average and max suffixes in Column Mode are used to indicate the average and maximum CPU usage scenario in hours. As suggested by the results, CPU usage of the motion-based sensor data analysis service was relatively stable. Although this background process was active most of the time, it was lightweight. The peak CPU usage was around 3%.

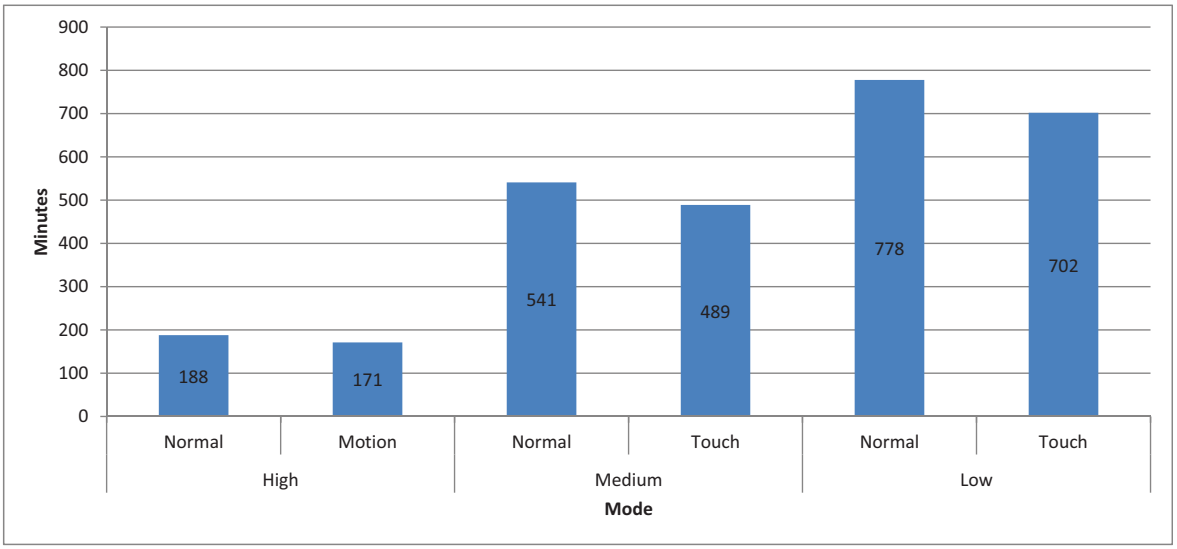


Figure 6.18: Power consumption percentage of the motion-based sensor data analysis service in each mode.

Mode	Average	Min	Max
High	6.7%	6.6%	6.8%
Medium	8.8%	8.6%	8.9%
Low	8.8%	8.6%	9.0%

Table 6.8: Power usage percentage of the motion-based sensor data analysis service.

6.5.3.3 Power Consumption Overhead

Battery life statistics for the motion-based sensor analysis service are shown in the Fig. 6.18. The impact on battery life was around 10% of the overall battery life.

Table 6.8 shows the min and max value of the power consumption usage caused by the motion-based background service. The power consumption was relatively stable at an acceptable level, below 10% of the overall power usage.

Process	Resource	Consumption Level
Waiting for input	CPU	Very Low
Security level check	File read, CPU	Low
Identity change check	File read	Low
Perform Actions	CPU	Low

Table 6.9: Resource overheads of the application manager.

6.5.4 Overhead Analysis for the Overall System

6.5.4.1 Background Services

The implemented SenGuard system consists of multiple sensor based analysis components mentioned previously, as well as an application manager. The application manager is a background service. It is in idle status for most of the time and is activated only when a user attempts to access an application. When a smartphone user tries to open an application, the application manager will first check the security setting of the application. If the application does not require user authentication, the application manager will return to the idle state. Otherwise, the application manager will evaluate the current user identity based on the previous sensor inputs, contexts, and user authentication history. If the system is uncertain about the user identity or there is a high likelihood that the current user identity has changed, the user needs to be re-authenticated before access to the application is granted. The resource usage of each step is shown in Table 6.10.

Mode	Average	Min	Max
High-Average	4.3%	4%	5%
Medium-Average	8%	7%	9%
Low-Average	2%	2%	2%
High-Max	23.3%	22%	24%
Medium-Max	52%	52%	52%
Low-Max	20.5%	18%	23%

Table 6.10: Overall CPU usage percentage.

6.5.4.2 Computational Overheads

The overall CPU usage is shown in Table. 6.10. The average CPU usage of the implemented system is below 10%, which is in the same range as common commercial apps with medium overhead. Note that peak CPU usage only represents infrequent burst user interaction scenarios.

6.5.4.3 Power Consumption Overheads

The overall statistics on impacts to battery life are shown in Fig. 6.19. As indicated by the results, the battery life was affected within 10%. Details of the power consumption data are shown in Table 6.11. In the worst case scenarios, the power consumption overhead was 17.5%. Given the definition that in the low usage mode, a users interacts with their smartphone less than 5 minutes per hour, the overall impact of SenGuard to battery hours is within acceptable range.

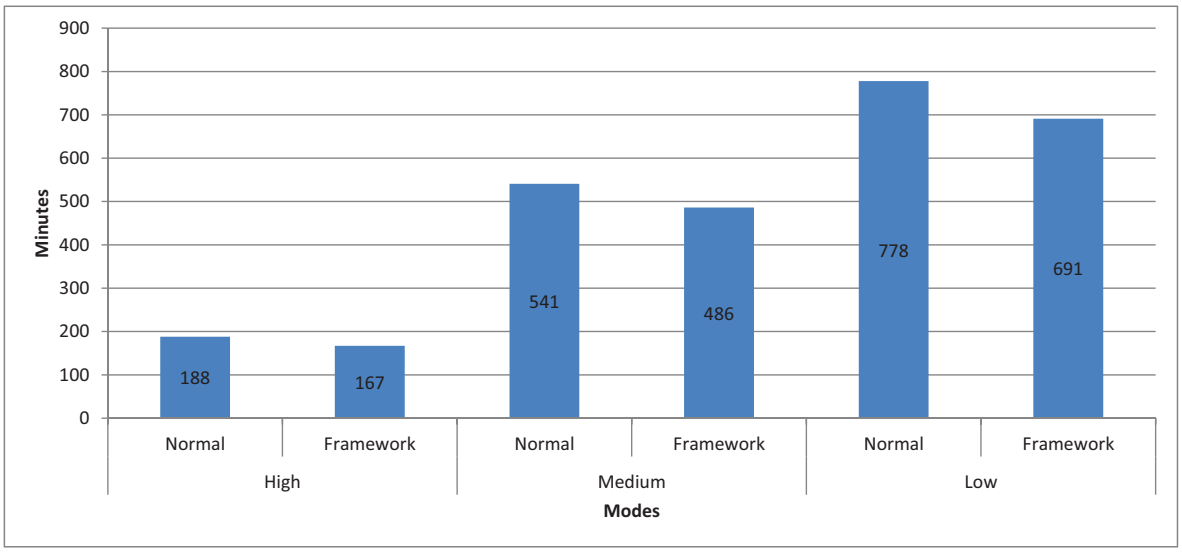


Figure 6.19: Power consumption percentage in each mode.

Mode	Average	Min	Max
High	9.3%	8.9%	9.8%
Medium	13%	12.4%	13.5%
Low	16.73%	15.2%	17.5%

Table 6.11: Overall power usage percentage.

Chapter 7

Conclusion

7.1 Summary of Work

I have investigated the feasibility of employing on-board mobile sensors to perform user-identity authentication, and based on the performance and experiment results of each individual sensor modality, I proposed an implicit user-identity approach integrating fingerprint sensor and Trustzone, which considering the trade-off between security and usability. The key contributions are summarized in this section.

7.1.1 Overall Research Contributions

The concept of on-demand user-identity verification and device-leaving-hands event on the mobile system is been proposed and evaluated in this thesis. I compared the novel schema with the previous time-out based identity management design,

and proves the effectiveness of our approaches using quantitative approaches and experiments. When designing the security system, I evaluate the usability of the system and design the approaches based on the usability design theory "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use." I employ quantitative approaches such as questionnaire to test the effectiveness, efficiency and satisfaction of the proposed approaches and interaction flow. I am the first three research teams that trying to use touch interaction data to identify user's identity and the first research team that investigated employing it to identify user's identity in the natural use. With quantitative approaches, I proved the feasibility of proposed and designed algorithms. These algorithms include transforming the touch input into images to prevent information loss for pattern matching, sliding window and threshold schema to improve performance, and multi-level template database to lower the consumption and improve the accuracy.

7.1.2 Touch-Based User-Identification

We presented FAST, GTGF, and TIPS, three different touch input based implicit user-identity recognition approach. FAST as a preliminary investigation of touch-based user-identification shows promising result. We further enhance the performance of touch-based user-identification in a controlled environment by employing GTGF features. However, we found these two approaches are not suitable for touch data in an uncontrolled environment. So we finally design and implement TIPS, an implicit user-identity recognition service that employs touchscreen data

in purely uncontrolled environments. TIPS analyzed real-life touch data as well as underlying contextual information for continuous user authentication. With extensive evaluation on the naturalized touch data collected from 23 phone owners and service deployment to 13 of them with 100 guest users, TIPS showed its effectiveness not only on offline simulation but also for on-device practical testing. TIPS is always running in the background and checking touch data continuously. In the future, we will leverage more contextual information to enhance TIPS' sequential model to build optimal duty-cycling, i.e., keeping "continuous" authentication as well as decreasing the computational cost maximally.

7.1.3 Voice-Based User-Identity Management

For the voice-based user-identity management, we presented the USR framework, a novel framework for integrating speaker-recognition based identity management into current human-mobile speech interaction framework. The USR framework employs many factors to perform seamless identity-based application management, including user-identity, application privacy level, usable app access control, and application function class. As a part of this study, we also contributed an open-source Android library for speaker-recognition. To investigate the effectiveness and efficiency of the technique, a controlled experiment was conducted comparing USR framework with an established speech recognition technique, Google speech recognizer, and in addition, a representation of current app access control approaches, Applock. The comparison study of these two frameworks shows that our unified

speech-speaker recognizer framework can significantly improve upon existing practical problems in mobile privacy protection and improve user experience to some extent.

7.1.4 Mobile Device Picking-Up Motion-Based User-Identification

We define a specific motion, mobile device picking-up motion, to detect user's identity by employing motion-sensors' data. We investigate the feasibility of a new behavioral biometric modality based on MDP motion. We propose two novel methods, a Statistical Method that intuitively applies classification algorithms on the smoothing sensor data; and a Trajectory-Reconstruction Method that reconstructs the MDP motion trajectories, to perform user identification. We evaluate our proposed methods on a multi-session MDP motion dataset from 31 subjects. From the results, we found that the accuracy of the methods is declined in inter-session tests. Furthermore, user movements(e.g. walking) highly impacted the accelerometer data, and further decreased the inter-session results. To solve this, we have to emphasize gyroscope and magnetometer data in case extra motion is detected. Last, experimental results of MDP motion-based verification are encouraging and point to the possibility of MDP motion-based biometric systems in real world applications.

7.1.5 An Integration of Context, Sensors and TrustZone

Throughout we have introduced and further explained IdentityTracker: a framework that not only tracks the user's identity through the fingerprint sensors, it combines these results with data from both the motion-sensors and interaction inputs, including touchscreen usage and speech inputs, while concurrently managing app-level access on smartphones in post-login stages. The idea focuses around motion and touch data to detect device-leaving-hand events based upon a predefined set of phone status, user status, and subtle gestures. To evaluate the performance of said system, we conducted two sessions of data collection to collect and clean the training data in both a controlled and uncontrolled environment. We then tested the trained model during the user's natural usage. Results have shown that our approach improves user security by not granting app-level access to unauthorized guest users while at the same time promoting usability by greatly reducing the amount of unnecessary authentications for the smartphone's owner. We have extended it as a novel method, Secure Session Service, for multi-stage verification of identities in sensitive payments or otherwise sensitive sessions. Using this method, we are able to isolate the sensitive data and processing functions from the regular (normal mode) operating system and effectively isolate these processes from any malware or malicious software present in the normal operating system. Because of the monitor, we can safely and reliably trust the secure mode with sensitive data and the processes of user verification. This three-stage verification method: The first stage being the fingerprint, the second which is continuous user verification, and the third, given the correct context, certificates used for signing which is

more secure than single-stage verifications that are found in the majority of current session-based implementations. Moreover, if a phone is stolen while not in session, the hacker is no longer able to retrieve certificates and other sensitive data or easily replicate the fingerprint as if a password and no TrustZone was used. However, if inter-session the mobile device is still secure because a transfer will automatically close the session. This session can only be reopened using a fingerprint. This method is further strengthened by the use of a Unauthorized Access Accountability Protections method that will record all instances of unauthorized attempts at accessing a secure session, recording the picture of the imposters as well as the location and time, and emailing this to the appropriate person. This framework allows for protections for pre, during, and post events that may lead to unauthorized accesses. However, there is much room to grow in terms of how secure the process is as a whole with emerging technology in mobile devices as well as advancing technology such as finger vein scanning and the addition of more sensors for more accurate readings. While not impenetrable, this approach strengthens security in todays information-stealing age.

7.2 Future Recommendations

To further improve the performance of the implicit user-identity management on mobile devices, following are some of the future directions:

1. One of the potential approach is to investigate some other sensors on smartphone device, such as GPS, and signal sensor, to investigate the feasibility

of employing location data, signal strength, and wifi access point to perform user-identification or provide context information to support other modalities user-identity verification.

2. With the rapid development of technology, wearable sensor now becomes another new hot topic in ubiquitous computing. New wearable devices are mostly sensor centered instead of computing centered, which provides potential for integration with the mobile device. The data from wearable sensors could bring interesting new sensor data, such as electrocardiogram (ECG), heart rate, pressure, and other specific features of human body, for implicit user-identity management.

3. Better machine learning and pattern recognition algorithms are another method to improve the performance. Online learning and Hidden Markov Chain could be a good test for better exploiting the collected sensor data under context-aware framework.

4. Finally, to have a better usability in implicit user-identity management, an integration approach employing Dempster-Shafer theory(DS theory) could be a possible way. The DS theory could provide an approach based on risks of different scenarios and context, and makes decisions not based on single or multiple biometric or behavioral recognition module. So the false alarm rate could be well managed to make the whole system a usable approach.

Bibliography

[1] CMU Sphinx – Speech Recognition Toolkit. http://cmusphinx.sourceforge.net.

[2] Linguistic Data Consortium: LDC. http://www.ldc.upenn.edu/.

[3] More top worst passwords. http://xato.net.

[4] Openears: free speech recognition and speech synthesis for the iphone. http://www.politepix.com/openears/.

[5] Security flaw! google voice actions usable on lock screen! http://forum.xda-developers.com/showthread.php?t=806923.

[6] Stereo bluetooth - no longer a broken record. http://www.imsresearch.com.

[7] Worldwide smartphone markets: 2011 to 2015 - analysis, data, insight and forecasts. http://www.researchandmarkets.com.

[8] Apple. IPhone 5S Tech Specs. https://www.apple.com.

[9] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, and C. Wenk. Frechet distance for curves, revisited. In *Proceedings of the Fourteenth Annual European Symposium on Algorithms,* 2006.

[10] H. Bae, S.-W. Kim, and C. Yoo. Building the android platform security mechanism using trustzone. In *International Symposium on Embedded Technology*, 2013.

[11] L. Bao and S. Intille. Activity recognition from user-annotated acceleration data. In *The Second International Conference on Pervasive Computing*, 2004.

[12] C. Bo, L. Zhang, and X. Li. Silentsense: Silent user identification via dynamics of touch and movement behavioral biometrics. In *Proceedings of the Nineteenth Annual International Conference on Mobile Computing and Networking*, 2013.

[13] T. Brezmes, J.-L. Gorricho, and J. Cotrina. Activity recognition from accelerometer data on a mobile phone. In *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*. 2009.

[14] S. Carroll, D. Carroll, and A. A. of School Administrators. *Statistics Made Simple for School Leaders: Data-Driven Decision Making*. Rowman & Littlefield Education, 2002.

[15] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *The ACM Transactions on Interactive Intelligent Systems*, 2011.

[16] S.-H. Chen and Y.-R. Luo. Speaker verification using mfcc and support vector machine. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2009.

[17] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, 2000.

[18] K. R. Cuntoor, A. Kale, A. N. Rajagopalan, N. Cuntoor, and V. Krger. Gait-based recognition of humans using continuous hmms. In *Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, 2002.

[19] G. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 2012.

[20] F. Eyben, M. Wollmer, and B. Schuller. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the International Conference on Multimedia*, 2010.

[21] Z. Fang, Z. Guoliang, and S. Zhanjiang. Comparison of different implementations of mfcc. *Journal of Computer Science and Technology*, 2001.

[22] L. Feng and L. K. Hansen. A new database for speaker recognition. Technical report, 2005.

[23] T. Feng, N. DeSalvo, L. Xu, X. Zhao, X. Wang, and W. Shi. Secure session on mobile: An exploration on combining biometric, trustzone, and user behavior. In *Sixth International Conference on Mobile Computing, Applications and Services*, 2014.

[24] T. Feng, Z. Gao, D. Boumber, T.-H. Liu, N. DeSalvo, X. Zhao, and W. Shi. Usr: Enabling identity awareness and usable app access control during hand-free mobile interactions. In *Sixth International Conference on Mobile Computing, Applications and Services*, 2014.

[25] T. Feng, Z. Liu, K. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. Nguyen. Continuous mobile authentication using touchscreen gestures. In *IEEE Symposium on Technologies for Homeland Security*, 2012.

[26] T. Feng, J. Yang, Z. Yan, E. M. Tapia, and W. Shi. Tips: context-aware implicit user identification using touch screen in uncontrolled environments. In *Proceedings of the Fifteenth Workshop on Mobile Computing Systems and Applications*, 2014.

[27] T. Feng, X. Zhao, N. DeSalvo, X. Wang, and W. Shi. Security after login: Identity change detection on smartphones using sensor fusion. In *IEEE Symposium on Technologies for Homeland Security*, 2015.

[28] T. Feng, X. Zhao, and W. Shi. Investigating mobile device picking-up motion as a novel biometric modality. In *IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems*, 2013.

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

[30] D. Gafurov, K. Helkala, and T. Soendrol. Biometric gait authentication using accelerometer sensor. *International Journal of Information Security*, 2006.

[31] T. Hasan and J. Hansen. A study on universal background model training in speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 2011.

[32] J. Hashimoto. Finger vein authentication technology and its future. In *Symposium on VLSI Circuits. Digest of Technical Papers*, 2006.

[33] M. Indovina, U. Uludag, R. Snelick, A. Mink, and A. Jain. Multimodal biometric authentication methods: A cots approach. In *Workshop on Multimodal User Authentication*, 2003.

[34] M. Jakobsson, E. Shi, and R. Chow. Implicit authentication for mobile devices. In *Fourth USENIX Workshop on HotSec*, 2009.

[35] I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter, and A. D. Rubin. The design and analysis of graphical passwords. In *Proceedings of the Eighth Conference on USENIX Security Symposium*, 1999.

[36] L. Jing, J. Chunhua, and Y. Xia. Design and implementation of security OS based on TrustZone. In *the IEEE International Conference on Electronic Measurement and Instruments*, 2013.

[37] B. Johnson. Google voice. *Computers in Libraries*, 2010.

[38] A. K. Karlson, A. B. Brush, and S. Schechter. Can i borrow your phone?: understanding concerns when sharing mobile phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009.

[39] J. Kittler, M. Hatef, R. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Learning*, 1998.

[40] O. Klein. Generalization of einstein's principle of equivalence so as to embrace the field equations of gravitation. *Physica Scripta*, 1974.

[41] J. Koreman, A. C. Morris, D. Wu, S. Jassim, H. Sellahewa, J. Ehlers, G. Chollet, G. Aversano, H. Bredin, S. Garcia-salicetti, L. Allano, B. L. Van, and B. Dorizzi. Multi-modal biometric authentication on the securephone pda. In *Workshop on Multimodal User Authentication*, 2006.

[42] A. Kumar, P. Reddy, A. Tewari, R. Agrawal, and M. Kam. Improving literacy in developing countries using speech recognition-supported games on mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012.

[43] A. Kumar, A. Tewari, S. Horrigan, M. Kan, F. Metze, and J. Canny. Rethinking speech recognition on mobile devices. In *Intelligent User Interfaces for Developing Regions*, 2011.

[44] L. Latha and S. Thangasamy. A robust person authentication system based on score level fusion of left and right irises and retinal features. *Procedia Computer Science*, 2010.

[45] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *Proceedings of the Nineteenth International Joint Conference on Artificial intelligence*, 2005.

[46] Y. Liu, A. Rahmati, Y. Huang, H. Jang, L. Zhong, Y. Zhang, and S. Zhang. xshare: supporting impromptu sharing of mobile phones. In *Proceedings of the Seventh International Conference on Mobile Systems, Applications, and Services*, 2009.

[47] H. Lu, A. Brush, B. P. A. Karlson, and J. Liu. Speakersense: Energy efficient unobtrusive speaker identification on mobile phones. In *IEEE Pervasive Computing*. 2011.

[48] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the Seventh International Conference on Mobile Systems, Applications, and Services*, 2009.

[49] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the Eighth ACM Conference on Embedded Networked Sensor Systems*, 2010.

[50] A. D. Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012.

[51] J. Mantyjarvi, M. Lindholm, E. Vildjiounaite, S. Makela, and H. Ailisto. Identifying users of portable devices from gait pattern with accelerometers. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005.

[52] P. Marcus, M. Kessel, and C. Linnhoff-Popien. Securing mobile device-based machine interactions with user location histories. In *Security and Privacy in Mobile Information and Communication Systems*. 2012.

[53] C. Marforio, N. Karapanos, C. Soriente, K. Kostiainen, and S. Capkun. Secure enrollment and practical migration for mobile trusted execution environments. In *Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, 2013.

[54] V. Panchal. A review on finger print recognition systems. *International Journal of Emerging Technologies in Computational and Applied Sciences*, 2013.

[55] M. Pirker and D. Slamanig. A framework for privacy-preserving mobile payment on security enhanced ARM TrustZone platforms. In *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2012.

[56] N. Ratha, R. Bolle, V. Pandit, and V. Vaish. Robust fingerprint authentication using local structural similarity. In *Fifth IEEE Workshop on Applications of Computer Vision*, 2000.

[57] J. Ravi, K. B. Raja, and K. R. Venugopal. Fingerprint recognition using minutia score matching. *arXiv.org*, 2010.

[58] D. Reynolds and R. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 1995.

[59] M. Rofouei, A. Wilson, A. Brush, and S. Tansley. Your phone or mine?: fusing body, touch and device sensing for multi-user device-display interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012.

[60] N. Sae-Bae, K. Ahmed, K. Isbister, and N. Memon. Biometric-rich gestures: a novel approach to authentication on multi-touch devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012.

[61] N. Sae-Bae, N. Memon, and K. Isbister. Investigating multi-touch gestures as a novel biometric modality. In *IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems*, 2012.

[62] M. Shahzad, A. Liu, and A. Samuel. Secure unlocking of mobile touch screen devices by simple gestures: you can see it but you can not do it. In *Proceedings of the Nineteenth Annual International Conference on Mobile Computing and Networking*, 2013.

[63] W. Shi, J. Yang, Y. Jiang, F. Yang, and Y. Xiong. Senguard: Passive user identification on smartphones using multiple sensors. In *IEEE Seventh International Conference on Wireless and Mobile Computing, Networking and Communications*, 2011.

[64] Symantec. Internet security threat report. Technical report, 2013.

[65] A. F. Syukri, E. Okamoto, and M. Mambo. A user identification system using signature written with mouse. In *Proceedings of the Third Australasian Conference on Information Security and Privacy*, 1998.

[66] J. Thorpe and P. C. van Oorschot. Graphical dictionaries and the memorable space of graphical passwords. In *Proceedings of the Thirteenth Conference on USENIX Security Symposium*, 2004.

[67] C. Varenhorst. Passdoodles; a lightweight authentication method. In *Proceedings of the Nineteenth International Conference on Multimodal Interfaces*, 2007.

[68] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, 1995.

[69] R. V. Yampolskiy and V. Govindaraju. Behavioural biometrics: a survey and classification. *International Journal of Biometrics*, 2008.

[70] J. Yang. Toward physical activity diary: motion recognition using simple acceleration features with mobile phones. In *Proceedings of the First International Workshop on Interactive Multimedia for Consumer Electronics*, 2009.

[71] C. H. You, K.-A. Lee, and H. Li. Gmm-svm kernel with a bhattacharyya-based distance for speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 2010.

[72] Z. Zeng, M. Pantic, G. Roisman, and T. Huang. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[73] X. Zhao, T. Feng, and W. Shi. Continuous mobile authentication using a novel graphic touch gesture feature. In *IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems*, 2013.

[74] X. Zhao, S. K. Shah, and I. Kakadiaris. Illumination normalization using self-lighting ratios for 3d2d face recognition. In *European Conference on Computer Vision. Workshops and Demonstrations*, 2012.

[75] J. Zhu, P. Wu, X. Wang, and J. Zhang. Sensec: Mobile security through passive sensing. In *International Conference on Computing, Networking, and Communications*, 2013.