Identity Authentication and Near Field Device Authentication for Smart Devices

by

Lingjun Li

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved July 2014 by the
Graduate Supervisory Committee:

Guoliang Xue, Chair
Gail-Joon Ahn
Jieping Ye
Yanchao Zhang

ARIZONA STATE UNIVERSITY

August 2014

ABSTRACT

The widespread adoption of mobile devices gives rise to new opportunities and challenges for authentication mechanisms. Many traditional authentication mechanisms become unsuitable for smart devices. For example, while password is widely used on computers as user identity authentication, inputting password on small smartphone screen is error-prone and not convenient. In the meantime, there are emerging demands for new types of authentication. Proximity authentication is an example, which is not needed for computers but quite necessary for smart devices. These challenges motivate me to study and develop novel authentication mechanisms specific for smart devices.

In this dissertation, I am interested in the special authentication demands of smart devices and about to satisfy the demands. First, I study how the features of smart devices affect user identity authentications. For identity authentication domain, I aim to design a continuous, forge-resistant authentication mechanism that does not interrupt user-device interactions. I propose a mechanism that authenticates user identity based on the user's finger movement patterns. Next, I study a smart-device-specific authentication, proximity authentication, which authenticates whether two devices are in close proximity. For proximity authentication domain, I aim to design a user-friendly authentication mechanism that can defend against relay attacks. In addition, I restrict the authenticated distance to the scale of near field, i.e., a few centimeters. My first design utilizes a user's coherent two-finger movement on smart device screen to restrict the distance. To achieve a fully-automated system, I explore acoustic communications and propose a novel near field authentication system.

Dedicated to my wife Xinxin and my family

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Dr. Guoliang Xue for his continuous support and guidance of my study and research at Arizona State University, and for his patience, motivation, enthusiasm, and immense knowledge. The guidance, encouragement, and academic freedom that he has given me help me all the time of research and writing of this dissertation. I could not have imagined having a better advisor and mentor for my Ph. D. study.

I would like to thank my dissertation committee members, Dr. Gail-Joon Ahn, Dr. Jieping Ye, and Dr. Yanchao Zhang for their insightful advise and comments. I have been honored to co-author with, Xinxin Zhao, Dr. Gail-Joon Ahn, and Gabriel Silva. Their unselfish help, insights, and feedbacks helped me improve my knowledge in the area. I am also grateful to my colleagues and friends: Dejun Yang, Xi Fang, Xiang Zhang, Vishnu Kilari, Ruozhou Yu, Ziming Zhao, Lei Liu, Yiming Jing, Qiang Zhang, and Yashu Liu for their friendship and care, which helped me survive through these years.

Finally and most importantly, none of my achievement would have been possible without the love and patience of my wife and my family. My most sincere thanks go to my wife Xinxin Zhao for her endless support and love during the past six years. I am very thankful to my parents, Daohong Shi and Huixin Li, for their dedication and unconditional support throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

Introduction

Authentication is to verify a claim. Identity authentication verifies the claim that a given user is the owner of a given identity. Proximity authentication verifies the claim that two given devices are in close proximity. Authentication is a fundamentally important security mechanism that provides a security base for many security applications. In other words, many security applications grant user services or privileges based on the authentication result. For example, identity authentication helps applications to recognize users so as to provide personalized services. The authentication is thus the very first step to the service. Many other applications even reject to provide services when the authentication fails. Smartphones are personal devices and unlocking a smartphone requires authenticating a user's identity. If the user is not one of the expected users, the smartphone would not allow the user to get into the system. Therefore, authentication is an important security mechanism.

The emerging usage of smart devices leads to the demands for new authentication types. Proximity authentication is a new authentication type that is closely associated with smart devices. Due to the mobility of the devices, physical proximity becomes a measure of trust between two smart devices. Proximity authentication is to ensure this trust before a security application takes place. For example, fast near field file transmission application

[3] transmits files between two smart devices only when they are in the near field. Authenticating the physical proximity of the two devices is therefore essential to the success of such applications.

1.1 Password-Based Authenticated Key Exchange

In many cases, security applications want authentication not only to verify the claim but also to output an integer number. The integer number is then used as a session key to protect the follow-up conversations in the application. Therefore, we require the integer number to be as random as possible, i.e., a random integer of high entropy. Password-based authenticated key exchange (PAKE) is such an authentication protocol that generates a high-entropy random integer from the same low-entropy password shared between two parties involved in the authentication. The phrase "low entropy" means the password is chosen from a small set of possible values. This small set is called *dictionary*. We will use PAKE many times in the following chapters of this dissertation.

In the model of PAKE, two involved parties share a short, low-entropy password and they want to agree on a high-entropy random integer that can be used as a cryptographic key. They want to achieve this in the presence of a powerful and malicious attacker, who fully controls the communication channels and can perform any attack. Since a password is of low entropy, the brute force method that tries all the possible values in the dictionary succeeds after a small number of attempts during the authentication. This attack is called *on-line dictionary attack* and is inevitable. However, to limit its damage, we can adopt a policy that a password is invalidated or blocked if a certain number of failed attempts have occurred. Therefore, one of PAKE's security goals is to ensure that online-dictionary attacks can be detected. In another attack, an attacker makes active or passive attacks during an authentication, and then performs brute-force attacks on the password dictionary in a offline way, i.e., no communications with the two parties involved in the authentication. This attack is called *off-line attack* and should be fully prevented by a PAKE protocol. PAKE is generally divided into two types, two-party PAKE and group PAKE. We are about to mainly use and discuss two-party PAKE in this dissertation.

Bellovin and Merrit [15] proposed the first PAKE scheme, the so-called Encrypted KEy Exchange (EKE). Their scheme is based on an ideal cipher, which is supposed to be a random permutation over a plaintext block for each given key. An attacker cannot infer any information about the output by encrypting any other block or the same block under any other key. The ideal cipher model is a heuristic rather than a plausibly true case in practise. Many extensions and analyses [5, 14, 20, 21] have been proposed to reduce the needs for ideal cipher but none of these works eliminated the ideal cipher from the protocol.

Katz, Ostrovsky, and Yung [56] proposed the first practical scheme that does not need an ideal cipher, i.e., works in standard model. The scheme assumes a common reference string from which each party can retrieve needed random strings. The common reference string model is plausible in a practical sense because a implementation can hard code all the needed strings in the program. The protocol takes place between a *client* and a *server*, and is described below.

- **Initialization:** provided a security parameter k, the protocol selects three other parameters: 1) a multiplicative group $\mathbb{G}$ of prime order q where q is of k bits, 2) random generators $g_1,\ g_2,\ h,\ c,\ d \in \bar{\mathbb{G}}$, where $\bar{\mathbb{G}} \overset{\text{def}}{=} \mathbb{G}\backslash\{1\}$, and 3) a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$ chosen at random from a collision-resistant hash family.
- The client generates a key pair (VK, SK) for signature, where VK and SK are respectively verification key and signing key. Picking a random integer $r_1 \leftarrow \mathbb{Z}_q$, the client calculates $A = g_1^{r_1}$, $B = g_2^{r_1}$, and $C = h^{r_1} \cdot pw_c$, where pw_c is the client's pass-

word. Next, the client combines all the necessary information together by using hash function: $\alpha = H(PID|VK|A|B|C)$, where PID is a string that uniquely identifies this execution. The original protocol proposed to use the concatenation of the client's name and the server's name as PID. After obtaining α, the clients calculates $D = (cd^{\alpha})^{r_1}$. Finally, the client sends message $Client|VK|A|B|C|D$ to the sever. This message is denoted by msg_1.

- The server selects five random integers $x_2,\ y_2,\ z_2,\ w_2,\ r_2 \leftarrow \mathbb{Z}_q$. Combining his version of the client's information together: $\alpha' = H(PID|VK|A|B|C)$, the server calculates $E = g_1^{x_2} g_2^{y_2} h^{z_2} (cd^{\alpha'})^{w_2}$, $F = g_1^{r_2}$, $G = g_2^{r_2}$, and $I = h^{r_2} \cdot pw_s$. The server receives the aforementioned message, msg_1. The server combines the message with his own information together: $\beta = H(msg_1|Server|E|F|G|I)$, and calculates $J = (cd^{\beta})^{r_2}$. Finally, the server sends message $Server|E|F|G|I|J$ to the client and this message is denoted by msg_2.

- The client selects four random integers $x_1,\ y_1,\ z_1,\ w_1 \leftarrow \mathbb{Z}_q$, and calculates $\beta' = H(msg_1|Server|E|F|G|I)$ and $K = g_1^{x_1} g_2^{y_1} h^{z_1} (cd^{\beta'})^{w_1}$. The client signs all the transmitted messages $Sig \leftarrow \mathtt{Sign}_{SK}(msg_1|msg_2|K)$ and sends $K|Sig$ to the server.

- The server verifies the signature using $\mathtt{Vrfy}_{VK}(msg_1|msg_2|K, Sig)$. If it is not true, the server aborts the protocol. Otherwise, the server calculates $C' = C/pw_s$ and obtains his session key $sk_s = A^{x_2} B^{y_2} (C')^{z_2} D^{w_2} K^{r_2}$.

- The clients calculates $I' = I/pw_c$ and obtains his session key $sk_c = F^{x_1} G^{y_1} (I')^{z_1} J^{w_1} E^{r_1}$.

First of all, the two resulting session keys are identical if the client and the server have the same password, $pw_s = pw_c$. The correctness can be verified by the following equations

$$\begin{aligned} E^{r_1} &= (g_1^{x_2} g_2^{y_2} h^{z_2} (cd^{\alpha})^{w_2})^{r_1} \\ &= (g_1^{r_1})^{x_2} (g_2^{r_1})^{y_2} (h^{r_1})^{z_2} ((cd^{\alpha})^{r_1})^{w_2} \\ &= A^{x_2} B^{y_2} (C')^{z_2} D^{w_2} \end{aligned}$$

and

$$\begin{aligned} K^{r_2} &= (g_1^{x_1} g_2^{y_1} h^{z_1} (cd^{\beta'})^{w_1})^{r_2} \\ &= (g_1^{r_2})^{x_1} (g_2^{r_2})^{y_1} (h^{r_2})^{z_1} ((cd^{\beta})^{r_2})^{w_1} \\ &= F^{x_1} G^{y_1} (I')^{z_1} J^{w_1}. \end{aligned}$$

Therefore:

$$sk_c = E^{r_1}(F^{x_1} G^{y_1} (I')^{z_1} J^{w_1}) = (A^{x_2} B^{y_2} (C')^{z_2} D^{w_2}) K^{r_2} = sk_s.$$

Here, we assume that $\alpha = \alpha'$, $\beta = \beta'$, and $pw_c = pw_s$ in an if both parties are honest.

The basic idea of protecting passwords from being eavesdropped is not to use it directly in any communication message. Instead, a ciphertext of the password, i.e., A, B, C, D or F, G, I, J, is transmitted over the channel. Even if the attacker is able to capture the ciphertext, he cannot learn anything about the password. The two parties involved in the protocol do not need to decrypt the ciphertext since they only want to achieve the same integer. Each member multiplies all the intermediate integers together and cancels his own password from the multiplication result. Hence, they can obtain the same number if the two password are identical. The reason why an online-dictionary attack can be prevented is that the result session key sk_c or sk_s is generated at random by each execution and is bounded with the intermediate communication messages. Without replaying the entire protocol with an honest party, an attacker cannot carry out a brute force attack. After that, many works [4, 24, 43, 57] have been proposed to achieve security in the UC framework, to enhance round efficiency, or to depend on new assumptions.

Many efficient protocols have been proposed. Among these protocols, SPEKE is an outstanding one that has been standardized [2]. SPEKE was first described by David Jablon [47]. In its first construction, SPEKE was not secure and Jablon thus proposed a refined construction in [48]. SPEKE has been proved to be secure in random oracle model by Philip MacKenzie [70]. SPEKE works as follows.

- The client and the server agree on a randomly selected safe prime p and a cryptographic hash function H. Again, the client and the server hold their individual password pw_c and pw_s, which are identical if both parties are honest.
- The client and the server respectively construct $g_c = H(pw_c)^2 \mod p$ and $g_s = H(pw_s)^2 \mod p$. Here, squaring makes the resulting integer a generator of quadratic residue subgroup of the multiplicative group of integers modulo p.
- The client selects an integer at random $a \leftarrow \mathbb{Z}$ and sends the server $g^a \mod p$.
- The server selects an integer at random $b \leftarrow \mathbb{Z}$ and sends the client $g^b \mod p$.
- The client and the server each abort if their received values are not in the range $[2, p-2]$. This is to prevent small subgroup confinement attack [66].
- The client computes session key $sk_c = (g^b \mod p)^a \mod p$.
- The server computes session key $sk_s = (g^a \mod p)^b \mod p$.

It is obvious that the server and the client arrive at the same session key if they have the same password. The protocol protects the confidentiality of the password by using one-way hash function which gives a unique digest for a given password. The digest is supposed to not leak any information about the password. This assumption is true in *random oracle* model.

1.2 Overview and Contributions

This dissertation discusses and studies two important authentication problems for smart devices — user identity authentication and proximity authentication. For identity authentication, we study the disadvantages of password based user authentications that is widely used on current smart devices.

We explore the features of smart devices and utilize the features to construct an identity authentication that overcomes the disadvantages of password based authentications and is suitable for smart devices. Proximity authentication is a new authentication technique that emerges with the widespread use of smart devices. Most existing proximity authentication scheme either requires an extra equipment or cannot restrict the two devices in a short proximity range. We study this interesting problem and propose two authentication systems that both overcome the aforementioned disadvantages. We call an authentication that can restrict the distance within a few centimeters a near field authentication (NFA). One scheme is to use human finger movement to restrict the physical distance between the two devices. The other scheme achieves the authentication based on the physical properties of sound.

1.2.1 User Authentication for Smartphones

For smartphones, the biggest issue of traditional password based authentication is that it interrupts the user-smartphone interactions. Users have to stop current session to input the password. The small virtual keyboard on most smartphones makes this process much more annoying since people often press wrong keys. In this study, we propose a novel user authentication scheme based on human finger movement patterns. The main contributions of this study are listed below.

Contributions:

- We propose and study the unobservable smartphone re-authentication problem. We design a novel system architecture especially for smartphones to reduce the computational overhead on smartphones.
- We propose to use the finger movement as a biometric characteristic to authenticate a user. When users use smartphones, the smartphones sense the users finger movements and interpret the sensed data as different gestures. Since users have to use gestures to interact with smartphones in most cases, our proposed approach can enforce re-authentication to every user. In addition, our approach can continuously re-authenticate the current user without being noticed by the user.
- We propose an efficient biometric-based re-authentication system for smartphones using the classification method. We design the biometric features for the classification algorithm and discuss their performance. We implemented our system on a Motorola Droid smartphone to demonstrate its efficiency. Extensive experiments were performed to evaluate the effectiveness of the features and the performance of our system.

1.2.2 Finger Movement Based NFA

The purpose of this study is to provide the NFA on most off-the-shelf devices. The proposed NFA system achieves NFA by using human finger movement on the touch screens of two nearby smart devices. Human input usually contains errors and is of low entropy, which affects the usability and security of a system. For these issues, we provide efficient solutions that can execute on the smart devices of limited resources. An outstanding feature of the proposed system is that it does not need any prior secret information shared between the two devices involved in the authentication. For a successful authentication, the system generates the same high-entropy cryptographic key for both devices. Finally, we build a prototype on a Motorola Droid smartphone to demonstrate the efficiency of the system.

Contributions:

- We propose to use zigzag on-screen finger movements to perform near field authentication between two smart devices. Compared with the previous motion patterns, such as bump, shake, etc., finger movements are easier to carry out and provides better user experience. Another advantage is that finger movements are small and hard-to-catch motions. The movements are hard to be observed and emulated by a nearby attacker, which is a possible attack to the bump system [1].

- We design a robust feature so that two extracted feature data sets are similar to each other. Zigzag finger movements provide many features, such as curvatures, curvature distance, moving time, etc. We propose to use the time between the starting point and a peak point as the feature to be extracted in our system. The reason behind this choice is that people's finger usually moves slowly, or even a short-time pause, when it turns at a curve peak point. Although the time is too short to be noticed by human eyes, it is long enough to be sensed by touch screens. This makes the elapsed time of two corresponding peak points very similar to each other.

- We propose an efficient system to remove the differences between two extracted feature data sets and generate a high-entropy cryptographic key. We design an efficient approach using a private set intersection protocol to reconciliate the two feature data sets. As pointed out previously, the feature data is of low entropy. We use the encrypted key exchange technique to defend against dictionary attacks and generate a high-entropy key.

- Our system is efficient. The efficiency of our system is twofold: 1) it requires less human involvement; 2) the computation overhead of our system is not heavy, which is demonstrated in our evaluation.

1.2.3 Acoustic NFA

The above finger movement based NFA system makes a solid step toward useful NFAs and is suitable for many scenarios. However, the system suffers from its non-automation and the human assistance makes the system not suitable for highly frequent authentications. For example, if a web server adopts smartphone based two-factor authentication (SBTFA), the browser and the smartphone may want to perform NFA in a high frequency during the entire web session. In addition, relay attacks pose a serious threat to existing approaches for proximity authentications. In this study, we present a novel NFA system that restricts the distance between the two devices to the scale of several centimeters. When the authentication succeeds, the system generate an assertion that can be used as an evidence of the authentication. Our system explores acoustic communications and can prevent relay attacks. The generated assertion is a confidential binary sequence known only to the two devices. Our system is fully automated and light-weight, as demonstrated by extensive evaluations on a prototype.

Contributions:

- It asserts whether two communicating devices are in the near field (a few centimeters) of each other. We use the term “near field” instead of “proximity” to emphasize that the asserted distance can be as small as a few centimeters. A device can prove that it is in the near field by presenting the assertion to the other party.

- It can prevent relay attacks. Dolphin has an adjustable time window limiting an attacker's relay time. A prudent implementation leaves an attacker no time to relay messages.

- It requires no extra equipments and can be easily deployed on off-the-shelf devices. Dolphin is a fully auto- mated system and needs no human interactions.
- A valid near field assertion generated by Dolphin is a binary sequence and confidential to the two devices that execute Dolphin. This property is important for applications that generate a cryptographic session key based on the devices' proximity relations, such as one-time file sharing between two proximate smartphones [3].
- It is light-weight and battery friendly for smartphones.

Part I

IDENTITY AUTHENTICATION

CHAPTER 2

User Identity Authentication for Smart Devices

The past few years have witnessed an exponential growth of smartphones, in both technology and market shares. According to a research done by canalys [23], smartphones were sold about 73 million more than personal computers (PCs) in 2011. Compared with PCs, smartphones are more privately owned. People may share a desktop, but few are willing to share their smartphones.

At the same time, smartphones are becoming an important personal entrance to various networks, such as the Internet or online social networks. Many apps and websites now allow people to store their accounts, profiles, passwords, etc., in smartphones for automatic re-access. Besides, people also use smartphones to keep contact with friends and families, take pictures of special moments, and arrange schedules. No one would like to disclose such information to an untrusted person. However, due to its small size, a smartphone could be easily taken away by an attacker. The attacker can acquire a good profit from reselling stolen smartphones. It is reported by lookout.com that $2.5 billion worth of devices were lost or stolen in 2011 [67]. Besides, having a victim's private information, an attacker can steal the victim's identity and launch impersonation attacks in networks. Such attacks substantially threaten the security of the networks, especially online social networks. Impersonation attacks also threaten most current trust and reputation systems for networks.

Therefore, protecting smartphones against unauthorized usage has significant meaning to safeguarding users' privacy and network security. A smartphone can alert the owner and lock itself when unauthorized usage is detected, which will inhibit most smartphone thefts.

To prevent unauthorized usage of smartphones, a *re-authentication* system is more suitable than an authentication system. An authentication system authenticates a user for one time when he logs in, such as inputting a password to unlock a smartphone. The purpose of a re-authentication system is to continuously authenticate the current user during the whole system execution. In the absence of re-authentication, it is easy for an attacker to access a smartphone if the owner forgets to lock it and leaves it in a public place. Even if the smartphone is locked, an attacker can use operating system (OS) flaws to bypass the lock screen, which is reported to exist in Android [53] and iOS [49] systems. The continuous protection provided by re-authentication is necessary for smartphones.

A straightforward re-authentication approach is to periodically invoke an authentication system, such as asking the user to enter a password [106]. This approach interrupts user-smartphone interactions and leads to bad user experiences. For smartphones, it is preferable that the re-authentication takes place in a way that users do not "observe" its existence.

Current short unlock passwords, such as 6-digit numbers, cannot protect smartphones against a powerful attacker. However, long and complicated passwords are difficult to memorize. Hence, a re-authentication system should rely on certain "password" that is easy to memorize but difficult to forge. A good candidate for such passwords is the owner's biological data. Many works have studied biometric-based authentication, such as fingerprint recognition [27], face recognition [35], and iris recognition [83]. However, these methods are not suitable for smartphone re-authentication because they either rely on special equipments, which are not available for smartphones, or need the users to stop interactions to assist the re-authentication. In addition, continuous face recognition requires keeping the camera on all the time, which dramatically reduces a smartphone's battery life.

In this study, we propose a re-authentication system for smartphones using users' finger movements. The system first learns the owner's finger movement patterns, keeps running in the background, continuously monitors the current user's finger movement, and compares the current user's movement patterns against the owner's patterns. Our system does not need user assistance in re-authentication and users are not aware of its execution.

The main contributions of our work are as follows:

- We propose and study the unobservable smartphone re-authentication problem. We design a novel system architecture especially for smartphones to reduce the computational overhead on smartphones.
- We propose to use the finger movement as a biometric characteristic to authenticate a user. When users use smartphones, the smartphones sense the users' finger movements and interpret the sensed data as different *gestures*. Since users have to use gestures to interact with smartphones in most cases, our proposed approach can enforce re-authentication to every user. In addition, our approach can continuously re-authenticate the current user without being noticed by the user.
- We propose an efficient biometric-based re-authentication system for smartphones using the classification method. We design the biometric features for the classification algorithm and discuss their performance. We implemented our system on a Motorola Droid smartphone to demonstrate its efficiency. Extensive experiments were performed to evaluate the effectiveness of the features and the performance of our system.

The rest of this chapter is organized as follows. We introduce the background and related work in Section 2.1. The attack model is introduced in Section 2.2. We discuss the design goals for a smartphone re-authentication system in Section 2.3. We present our re-authentication system in Section 2.4. We discuss the feature design and selection in Section 2.5. We evaluate our re-authentication system in Section 2.6, and conclude our work in Section 2.7.

2.1 Background and Related Work

Compared with traditional authentications, biometric-based authentications are easy to carry out, natural to use, and invulnerable to forgery. Traditional approaches are based on possessions of secret information, such as passwords. Biometric based approaches make use of distinct personal features, such as fingerprint or iris.

A biometric-based re-authentication system involves an enrollment phase and a re-authentication phase. A user is enrolled by providing his biological data. The system learns patterns from the provided data and stores the learned patterns for future reference. During the re-authentication phase, the system compares the observed biological data against the stored data to re-authenticate a user.

Previous studies on biometric-based re-authentication concentrated on either physiological or behavioral features [103]. Physiological biometrics study static physical features of humans. Currently, there are many different physiological biometrics for re-authentication, such as fingerprint [27], face patterns [35], and iris [83]. However, physiological biometric-based re-authentication approaches usually rely on specific devices, which are unavailable on most smartphones. In addition, most approaches need human assistance in the re-authentication. For example, most face recognition systems need the users to stay still at a specific angle to the camera during re-authentication. Hence, these physiological biometric-based approaches cannot achieve continuous unobservable re-authentication.

Behavioral biometrics assume that people have distinct stable patterns on a certain behavior, such as keystrokes on a keyboard. Behavioral biometric-based re-authentication uses the behavior patterns to authenticate a user's identity. For personal computers, most previous studies concentrated on two operations: keystrokes [16, 74, 75] and mouse movements [54, 107]. Typing actions happen much less frequently on smartphones than on personal computers, because people hardly use smartphones to do heavy text input. Therefore, it is difficult to collect a sufficient number of keystrokes on smartphones for re-authentication.

Although smartphones do not have mouse input devices, previous studies [7, 76] on mouse movements help us to understand finger movements on smartphones. Hence, we give more detailed review of prior works on mouse movements.

2.1.1 Mouse Movements

When a mouse moves, the hardware captures the movement and sends the mouse events to the OS, including raw movement coordinates, button up, and button down events. The OS interprets these mouse events to a series of point data, which form a mouse movement. In the approach proposed by Ahmed and Traore, point data are aggregated as *point-and-click* or *drag-and-drop* actions [6, 7]. A point-and-click action contains a click event and a mouse movement following the click. A drag-and-drop action is a mouse movement with one button pressed down. The reason to study the two actions is that they are both performed intentionally by users. Ahmed and Traore characterized each action using action type, moving distance, duration, and direction [7]. They computed 39 dynamics related features and used a neural network to classify new observed actions.

Recently, Zheng *et al.* [107] proposed to use only the point-and-click action and three features: *direction, angle of curvature*, and *curvature distance*, to authenticate a user. The classifier they used is SVM. They aggregated the features of 20 point-and-click actions as a feature vector. Their work required 20 mouse movements, compared with 2000 mouse movements required in Ahmed and Traore's work [7]. This reduction decreases the data collection time and hence increases the re-authentication frequency. In their work, they collected 81218 point-and-click actions from 30 users in a controllable environment and one hour raw mouse events from 1074 anonymous users from an online forum. The average false rejection rate and the average false acceptance rate were both 1.3% in their tests.

In another approach, Pusara and Brodley utilized the connections between each pair of points within a window of a configurable size [82]. The features, such as angle, distance, and speed, were extracted from the points rather than the actions. They used C5.0 decision tree as the classifier in their system, which achieved an average false acceptance rate of 0.43% and an average false rejection rate of 1.75% in the experiments on an eleven-user data set. Gamboa and Fred [39] aggregated the points between two clicks. Each click is represented by 63 features. For each user, they proposed a greedy approach to reduce the feature set to a best fit subset.

2.1.2 Smartphone Features

One of the biggest differences between personal computers and smartphones is that smartphones are equipped with many sensors, such as multi-touch screen, accelerometer, and gyroscope. Although different smartphones may have different sensors, multi-touch screen, accelerometer, and compass are provided by most smartphones.

Multi-touch screen is a basic equipment on a smartphone. A multi-touch screen is able to respond to more than one finger touch. The number of supported touch points varies from device to device. Some basic screens can only support two touch points while some advanced ones are able to support up to ten touch points. The multi-touch screen records the touch position, touch area, and touch pressure, packs them as a single touch event, and sends it to the OS. A series of touch events are connected together and recognized as different gestures, such as sliding, tap, double tap, or spread.

Accelerometer measures the phone's acceleration on three axis, x, y, and z [42]. This captures a smartphone position in a three-dimensional space.

Orientation indicates whether a smartphone is held in portrait mode or landscape mode.

Compass measures the position of magnetic north in relation to the X, Y, and Z axies of the phone.

Various sensors in smartphones provide a lot of biological data of a user, which can be used in biometric-based authentication. Some previous works have studied using smartphone sensors for security purpose. Some works used accelerometer to sense a person's shake motion data to securely pair two devices [25, 72]. Mäntyjärvi *et al.* first considered using sensors to record users' behavioral patterns and to continuously re-authenticate a user [71]. They suggested to use accelerometer and orientation sensors to monitor a user's walking patterns. Their approach can successfully recognize a user at rates between 60% and 85%. Okumura *et al.* proposed to authenticate a user using the accelerometer data sensed when the user is swinging his arm [79]. Instead of using the false acceptance rate or the false rejection rate, they claimed their system's equal error rate – the error rate when the false acceptance rate is equal to the false rejection rate – was able to achieve as low as 5%. Recently, Conti *et al.* proposed to re-authenticate a user using the arm movement patterns, sensed by the accelerometer and orientation sensors, while the user is making a phone call [30]. They achieved a false acceptance rate of 4.44% and a false rejection rate of 9.33% in their tests.

Recently, Biometric Signature ID company has proposed to use gesture based signature to re-authenticate a user in the log-in phase [17]. This approach records a user's signature during the enrollment phase and compares an input signature against the recorded one during re-authentication. Luca *et al.* [33] proposed to use gesture information, such as touching area or duration, as an additional re-authentication approach on top of the current password pattern approach. The two methods are both one time re-authentication and will interrupt user-smartphone interactions if they want to achieve continuous re-authentication. Different from these works, our system aims to provide a continuous unobservable re-authentication.

Existing continuous re-authentication approaches have paid extensive attention to the accelerometer and orientation sensors and used behaviors that may not happen during an attack in our scenario. For example, an impostor may not swing arms when he uses a victim's smartphone. Therefore, we need an approach that can continuously re-authenticate a user as long as he is using the smartphone. We propose to use and monitor the gesture on smartphone touch screen, which is the most important and necessary interface between users and the smartphone OS.

2.1.3 Smartphone Gesture Patterns

Here we first give several observations on smartphone gestures, which differentiate the finger movement on smartphones from the mouse movement on computers.

Usage Intermittence: people may not always use smartphones for a long time.

Typical usages are to wake up a smartphone, click an email app, check if there is any new email, and then turn off the screen. The collected gestures are thus not temporarily continuous.

Spacial Disconnection: In the study on mouse movement patterns, each movement can be captured by hardwares and used to formulate patterns, such as the point-and-click patterns. On smartphones, not every finger movement can be captured by a touch screen. For example, a user lifts up his finger, moves in the air, and clicks a link on a webpage. In these cases, the screen cannot capture the finger movement in the air, which corresponds to the point action in mouse movements.

Orientation Dependent: Users may use smartphones in either portrait or landscape orientations. Users' gestures have different patterns in different orientations. For example, a sliding up distance becomes shorter in the landscape mode.

2.2 Attack Model

We consider an attacker who has physical access to the smartphone and wants to use the resources in it, such as applications or music. For example, an attacker may steal a victim's smartphone and enjoy the music in it without paying any money. The attacker may also steal the network account information and the personal information stored in the smartphone. For example, the attacker can post a fake message in a social network using the victim's account. The purpose of our work is to design a continuous re-authentication system running in the background. The system keeps authenticating the current user in an unobservable way, i.e., it does not interrupt the user's interactions with the smartphone. In this paper, we only consider the authentication of a user against the smartphone owner, because a smartphone is usually privately owned and not shared by multiple users. If the user is found to be a stranger, the re-authentication system alerts the OS.

2.3 Design Goals

We summarize the goals that a smartphone re-authentication system should achieve in the following.

- **Continuity**: A smartphone re-authentication system should keep authenticating the current user as long as the smartphone is being used.
- **Unobservability**: A smartphone re-authentication system should neither interrupt user-smartphone interactions nor need human assistance during re-authentication.
- **Light-weight**: A smartphone re-authentication system should not need intensive computations on smartphones.

2.4 Approach

We are ready to present our smartphone re-authentication system, which achieves the design goals discussed in the above section.

Our idea stems from the observation that users' finger movements on smartphone screens are different from person to person when they use smartphones. For example, some people like to use the index finger to slide up the content displayed on the screen while some people prefer to use the thumb. Following customs in smartphone development, we call a continuous finger movement on the screen a *gesture*. We assume that a user's gestures contain his distinct behavioral characteristics.

Our work uses such characteristics to re-authenticate users. We illustrate our system architecture in Figure 2.1. Considering the limited computational and storage resources in a smartphone, our system is divided into two modules, the re-authentication module and the training module. The re-authentication module is deployed in a smartphone and the training

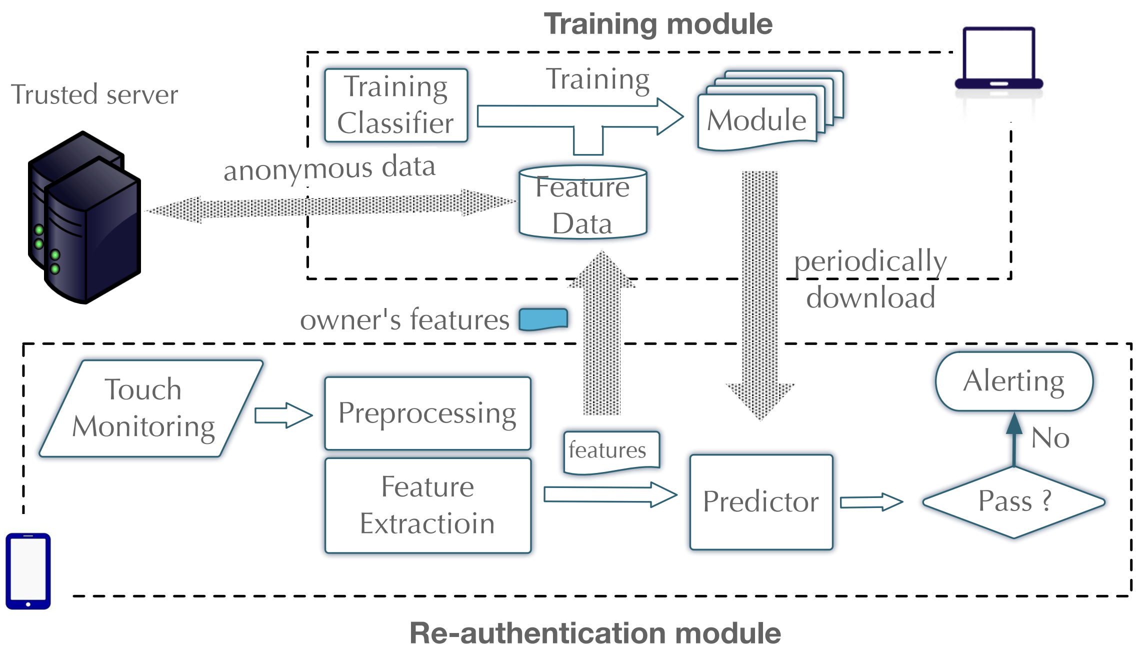


Figure 2.1: System Architecture

module is executed on a PC. To provide a better security and performance guarantee, we suggest to implement the re-authentication module as part of the smartphone OS services in practice.

The re-authentication module keeps running in the background of smartphones. It monitors a user's raw touch event data and sends it to the preprocessing component, which assembles every single raw data into different gestures and then sends them to the feature extraction component. The latter component extracts features from the gesture data, forms a feature vector, and feeds it into the predictor component. Finally, the predictor component makes a prediction. If the feature vector is predicted to be from the smartphone owner, this re-authentication is passed. Otherwise, an alert message will be sent to the OS. Different OSs may take different actions in response to the alert. One possible action is to lock the system and ask the user to input an administrator password. Another possible action is to send a message, with the current GPS information in it, to the owner's e-mail box.

The predictor component consists of a classifier and multiple classification modules, as shown in Figure 2.2. A classifier is a classification algorithm that uses an object's feature vector to identify which class the object belongs to. The classification algorithm used in our work is the support vector machine (SVM) algorithm. Each classification module is in charge of a main gesture type or a combination of a main gesture type and an auxiliary type. A classification module is a file containing parameters for the classification algorithm and determines the classifier's functionality. The basic classification algorithm is embedded in the classifier. Using different classification modules, the classifier can make predictions on feature vectors of different gesture types. When a feature vector is fed in, the classifier chooses a corresponding classification module and makes a prediction.

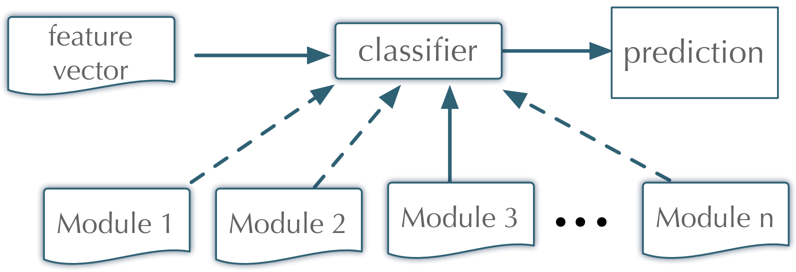


Figure 2.2: Predictor Component

The training module is executed on the owner's PC, because it requires significant computations. When a smartphone owner first enrolls in the system, the system collects the owner's gesture features by using the touch monitoring, preprocessing, and feature extraction components of the re-authentication module. Our system deploys a trusted data server to collect feature data from smartphone owners and downloads them to the training modules when necessary. To protect an owner's privacy, the data collection is done anonymously. This can be achieved by using anonymous group messaging [31, 104]. A fixed number of a user's feature data and a time stamp form a ready-to-submit feature message. Every user in our system is a group member and the server is the data collector in the anonymous group messaging system. The users collaboratively shuffle their messages before the messages are handed in to the server. Eventually, the server does not know the connection between a message and its owner. In this way, a user's training module can use other users' feature data but has no way to know the user identities. We note that a user only wants to download other users' feature data. Therefore, a user compares every downloaded feature message against his own submissions and drops the one that is the same with one of his submissions. The comparison can be based on the hash value of the messages to reduce the time and storage overhead.

The training module uses the owner's features and other people's features in the training algorithm to obtain classification modules. After training, the classification modules are downloaded onto the smartphone. The training module anonymously uploads the owner's data to the trusted server and obtains anonymous features from it. We note that this trusted data server does not participate in the re-authentication and is only needed when an owner wants to re-train his classification modules, which is done offline and on-demand. Therefore, our system does not pose a high requirement on the communication delay between smartphones and the server.

An owner's usage pattern usually stays stable. But sometimes, the owner may change his usage pattern over weeks or months, which may cause more false alarms. When this happens, the classification modules need to be re-trained. To keep the modules up to date, our system also allows an on-demand re-training. When the owner requests a re-training, the re-authentication module captures the owner's gestures, calculates and uploads the owner's feature vectors to the training module. The training module then downloads anonymous feature messages from the server, filters out his own submissions, and runs the classifier training algorithm again to obtain new classification modules.

We note that the access to the system enrollment and re-training process should be restricted to the smartphone owner only. This can be achieved, for example, by using traditional password based protection.

2.5 Characterizing Gestures

Our system monitors five types of gestures: sliding up, sliding down, sliding left, sliding right, and tap, as shown in Figure 2.3. Usually, slidings are used to move contents

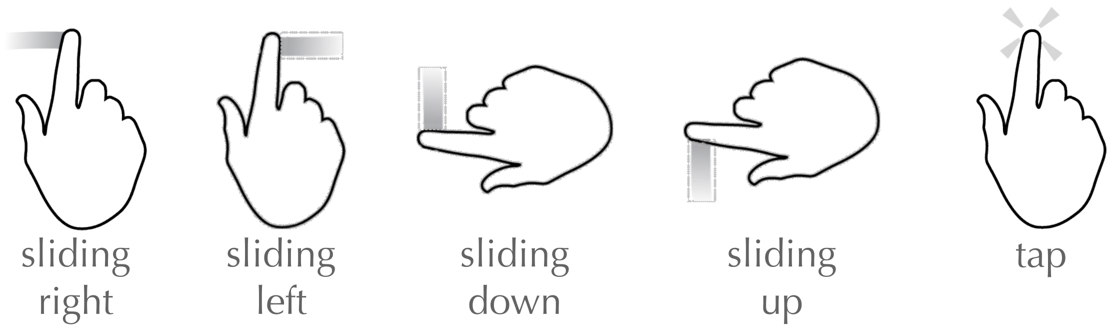


Figure 2.3: Five Essential Gestures

displayed on the screen and tap is used to click a link or a button within the contents. Although there are some other gestures, such as double tap, spread, and pinch, the five gesture types are the most often used types when users interact with smartphones. We collected 75 users' gestures when they used smartphones. We show the proportions of different gesture types in a pie chart in Figure 2.4. It shows that the above five types of gestures take a dominant part of all the gestures. In other words, most users inevitably used at least one of the above gesture types when they used smartphones. As shown in Figure 2.4, slidings and taps occupy 88.8% of all the gestures. We remark that we do not consider virtual keyboard strokes here, because they are not suitable for smartphone re-authentications. Keystroke based authentications usually need a number of continuous keystrokes, but most users do not continuously input many texts on smartphones. In addition, an attacker can use a vic-

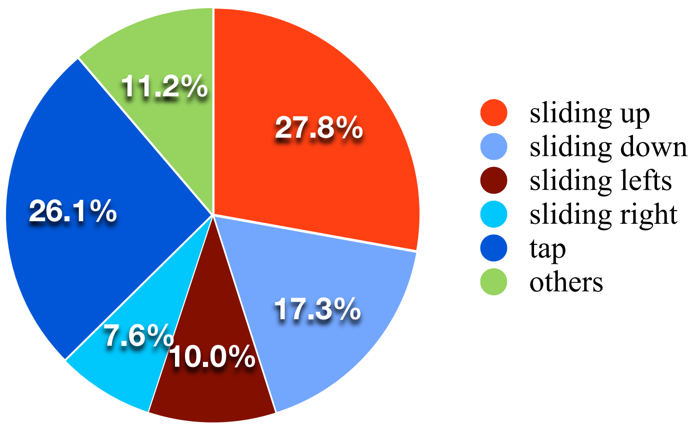


Figure 2.4: Pie Chart for Collected Gestures

tim's smartphone without many continuous keystrokes.

Users can hold smartphones in either portrait mode or landscape mode. As pointed out in Section 2.1.3, the orientation mode affects a user's gesture patterns. Hence, our system has two classification modules for every gesture to deal with each orientation mode.

2.5.1 Data Collection

The open-source Android system is selected as our implementation platform. Specifically, all of our experiments and data collections were carried out on Android 2.2.2. The first thing we need is a program that can monitor a user's finger movements in the background. However, for security reasons, Android requires that only the topmost apps can obtain touch events, dispatched from the Android system management service. In other words, we cannot enjoy the convenience that Android API provides to developers and have to work around this problem. We found that Android relies on Linux kernel for core system services, including the maintenance of hardware drivers [41]. When some touch event happens, a screen reads in raw data and sends it to the Linux kernel. The kernel then packages the raw data and sends it to the upper layer Android library. Since we cannot get input data from Android API, our idea is to read input data directly from lower layer Linux kernel.

Linux kernel uses device files to manage devices, located under the directory `/dev/`. Same as other devices, a multi-touch screen also has a corresponding device file, say `/dev/event3`. When the multi-touch screen reads inputs, the data are put in the device file by the kernel. The data orgnization follows the Linux multi-touch event protocol [86]. In the protocol, touch details, such as position, touch area, pressure, etc., are sent sequentially as Linux ABS event packets [86]. Each packet contains an ABS event indicating a specific touch data. Packets are separated by a `SYN_MT_REPORT` event (type `0002`). When all touch packets in a multi-touch action arrive, a `SYN_REPORT` event (type `0000`) is generated. A typical multi-touch ABS packet is as follows:

```
0003    0030    00000005
0003    0032    00000018
0003    0035    000002b6
0003    0036    00000296
0000    0002    00000000
0003    0030    00000003
0003    0032    00000012
0003    0035    0000024d
0003    0036    000001e4
0000    0002    00000000
0000    0000    00000000
```

The first byte indicates the event type: `0003` is an ABS event and `0000` is an `SYN` event. The second byte indicates the data type: `0030` is `ABS_MT_TOUCH_MAJOR` major axis of touch ellipse; `0032` is `ABS_MT_WIDTH_MAJOR` major axis of approaching ellipse; `0035` and `0036` are `ABS_MT_POSITION_X` and `ABS_MT_POSITION_Y`, respectively, giving the central position of an ellipse. These four basic data types are supported by all Android smartphones. Other data types include `ABS_MT_TOUCH_MINOR`, `ABS_MT_WIDTH_MINOR`, `ABS_MT_ORIENTATION`, `ABS_MT_TOOL_TYPE`, `ABS_MT_BLOB_ID`, and `ABS_MT_TRACKING_ID`. The multi-touch protocol recognizes a finger touch area as an ellipse and describes it using its major and minor axises. Some low-end devices, such as the Motorola Droid smartphone we used, recognize a touch area as a circle and omit the minor axis value. `TOUCH` type data describes the area that a finger directly contacts the screen and `WIDTH` type data describes the shape of a finger itself. The ration of `ABS_MT_TOUCH_MAJOR/ABS_MT_WIDTH_MAJOR` gives the touch pressure. The last two bytes in each line represent the data value. The above packet contains two finger data details, separated by `0000 0002 00000000`.

Our monitoring program needs the root privilege to hack into the lower layer of an Android system. Such a re-authentication system is usually integrated into the OS and can be granted the root privilege by the OS.

We carried out our data collection and all the experiments on two Motorola Droid phones, with 550MHz A8 processor, 256MB memory, 16GB sdcard, and Android 2.2.2 OS. In order to collect gesture data, 75 users were invited to take away our smartphones for days and use them freely. We did not specify any requirement on the usage and let the users use the smartphone in any way they feel comfortable. The users can browse web pages, including news, online forums, social network websites, etc., or use the installed apps, such as twitter, facebook, google reader, etc. Users were not required to continuously use the smartphone. They could lock the smartphone and resume using it later. In summary, we want the participants to use smartphones in the same way that they use their personal smartphones in their daily life.

2.5.2 Metric Design

Good features are critical to a supervised machine learning approach, which is used in our system. In this section, we design the metrics to characterize the five types of gestures. In Section 2.5.3, we test whether a metric is good and drop the bad ones. A feature is the average metric value over a *block* of gesture data.

2.5.2.1 Metrics of Sliding

First, we inspect what happens during a sliding gesture. Figure 2.5 shows the sensed data of a sliding gesture, a sliding up, recorded by our touch monitoring component. We note

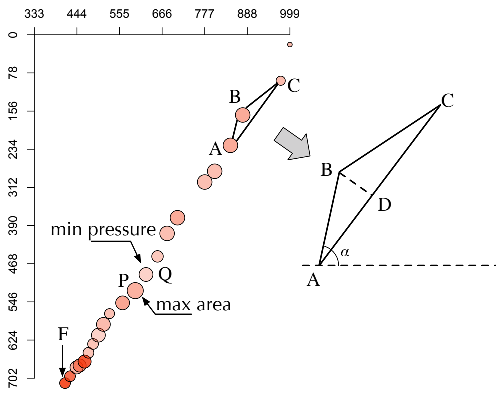


Figure 2.5: A Sliding Up Gesture

that the coordinate on the smartphone platform puts the origin at the top left corner of a screen. Each circle represents a finger touch, because Motorola Droid phone views a finger touch as a circle. The size of a circle shows the size of the touch area and the brightness of a circle shows the strength of the touch pressure. The movement starts at point F and ends at point C. The time between every pair of circles is the same. Apparently, the finger moves slowly at first, then faster, because the circles become sparser as the finger moves.

Our first interesting observation – which is different from our intuition – is that the maximum touch area may not happen at the first touch and the minimum pressure may not happen at the last touch. As can be seen from the figure, the touch point P has the largest touch area and point Q has the smallest touch pressure, both of which are neither the first touch nor the last touch. Another observation is that strong pressures happen at the beginning of a sliding. Despite the first 5 points, the variations of touch pressures are not as big as that of touch areas.

We propose the following metrics for a sliding gesture:

- **First touch position:** the coordinates, x and y, of the starting point in a sliding.
- **First touch pressure:** the pressure of the first touch.
- **First touch area:** the touch area of the first touch.
- **First moving direction:** the moving direction of a touch point is the angle between the horizontal line and the line crossing the point and its succeeding point. The angle α in Figure 2.5 is the moving direction of point A. First moving direction is the moving direction of the starting point.
- **Moving distance:** the total distance of the sliding gesture. Particularly, it is the summation of the distances between every two continuous touch points.
- **Duration:** the time duration of the whole sliding gesture.
- **Average moving direction:** the average value of all the point's moving directions. We note that the last point is not counted, because it does not have a moving direction.
- **Average moving curvature:** given any three temporally continuous touch points, such as A, B, and C in Figure 2.5, the corresponding moving curvature is angle $\angle ABC$.

The average value of the moving curvatures in the sliding gesture is selected as a metric.

- **Average curvature distance:** given any three consecutive points, such as *A*, *B*, and *C* in Figure 2.5, the corresponding curvature distance is the distance from point *B* to line *AC*. We take the average of all the curvature distances as a metric.
- **Average Pressure:** the average of all the touch pressures in the sliding gesture.
- **Average touch area:** average of all the touch areas.
- **Max-area portion:** we index all the points according to the time order, starting from 1. The max-area proportion of the sliding gesture is the index of the max area touch point divided by the total number of the points in the sliding. This metric reflects which portion of the sliding contains the maximum touch point.
- **Min-pressure portion:** Similar to max-area portion, the min-pressure portion is the index of the minimum pressure touch point divided by the total number of the points.

The final feature vector is calculated over a *block* of sliding gestures. The block size is denoted by n_s. Each feature value in the feature vector corresponds to an average metric value over the block of sliding gestures.

2.5.2.2 Metrics of Tap

Tap is a simple gesture and does not provide much information about a user's finger movement patterns. In contrast to our intuition, many tap gestures contain more than one touch points. It is due to the screen's high sample frequency and the slight tremble of a user's fingertip when he is touching above the screen. The metrics for a given tap gesture are as follows:

- **Average touch area:** the average of all the touch areas.
- **Duration:** time duration of the tap gesture.
- **Average pressure:** the average of all the touch pressures.

Similar to the calculation of a sliding feature vector, a tap feature vector is also the average metric values over a block of tap gestures. The block size is denoted by n_t.

2.5.3 Metric Selection

According to our observations about users' behaviors of using smartphones, we proposed different metrics in Section 2.5.2, trying to characterize a user's gestures. Selecting good metrics is essential for a supervised machine learning method, such as SVM used in this work. In this section, we test the performance of each metric and drop the bad metrics. If a metric can be used to easily distinguish two users, we say the metric is a good metric. We view a metric value calculated from a person's gesture as a data sampled from an underlying distribution of the metric. For a metric to distinguish two different persons, it is necessary to require the two underlying distributions to be different. Therefore, for a metric, we construct a metric data set for each invited user in the data collection by calculating the metric value from each of his sliding gestures. Then, we tested whether two metric data sets are from the same distribution. If most pairs of the data sets are from the same distribution, the metric is bad in distinguishing two persons and we need to drop it.

We use two-sample Kolmogorov-Smirnov test (K-S test) to test if two metric data sets are significantly different. Two-sample K-S test is a nonparametric statistical hypothesis testing based on maximum distance between the *empirical cumulative distribution* *functions* of the two data sets. The two hypotheses of K-S test are:

$$H_0 : \textit{the two data sets are from the same distribution;}$$

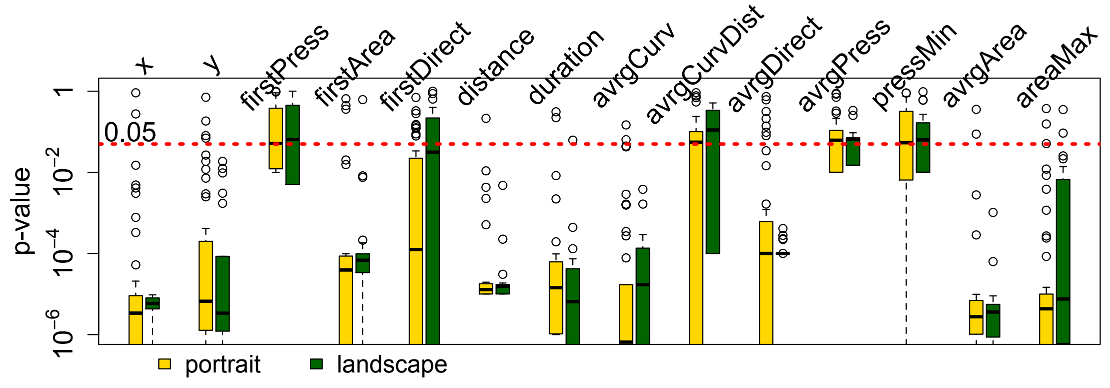
$$H_1 : \textit{the two data sets are from different distributions.}$$

A K-S test reports a p-value, i.e. the probability that obtaining the maximum distance is at least as large as the observed one when H_0 is assumed to be true. If this p-value is smaller than a significant level α, usually set to 0.05, we will reject H_0 hypothesis because events with small probabilities happen rarely. For each metric, we calculated the p-value for each pair of the metric data sets and drop the metric if most of its p-values are greater than α.

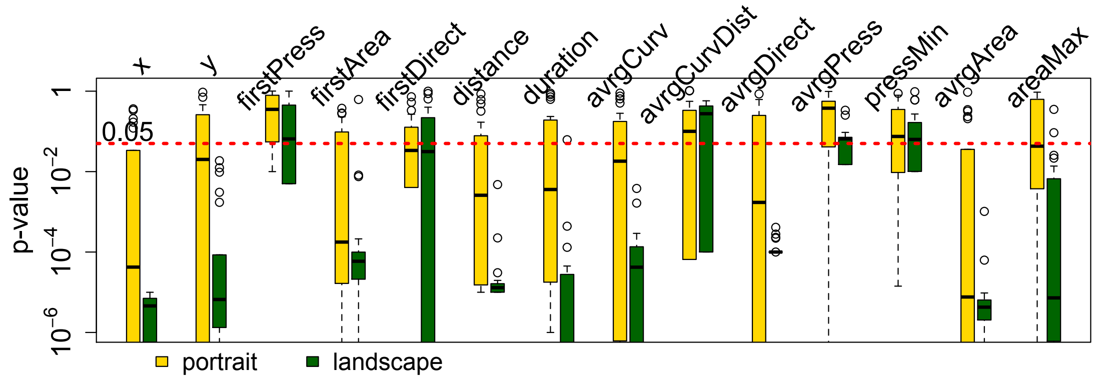
2.5.3.1 Sliding Gesture

Figure 2.5 shows the testing results for the metrics of the four sliding gestures in both portrait and landscape modes. Due to space limitation, we abbreviate some metric names in the figures. firstPress is "first touch pressure", firstArea "first touch area", firstDirect "first moving direction", distance "moving distance", avgCurv "average moving curvature", avrgCurvDist "average curvature distance", avrgDirect "average moving direction", avrgPress "average pressure", pressMin "min-pressure portion", avrgArea "average touch area", and areaMax "max-area portion".

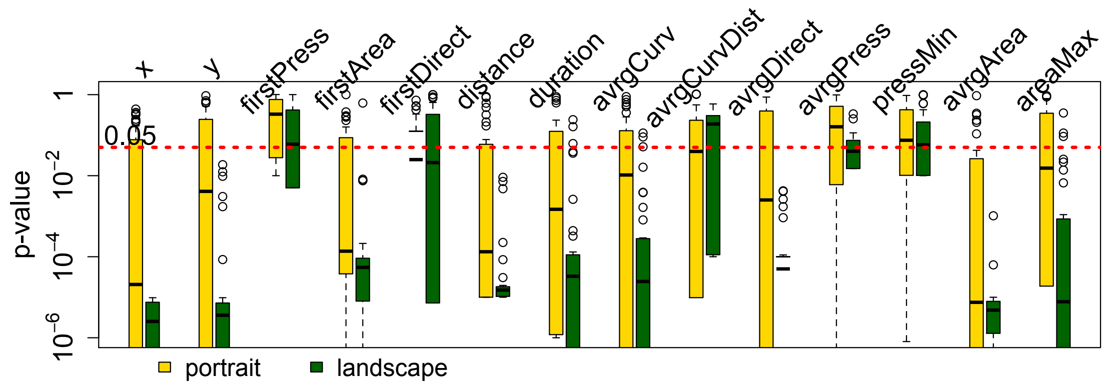
For each metric, the resulting p-values are drawn in a box plot. The bottom and the top of the box denote the lower quartile $Q1$ and the upper quartile $Q2$, defined as the 25th and the 75th percentiles of the p-values. The middle bar denotes the median of the p-values. The lowest and the highest bars outside the box denote the lower and the upper outer fences, defined as $4Q1-3Q2$ and $4Q2-3Q1$, respectively. The results from portrait orientation are represented by yellow boxes and those from landscape orientation are represented by green boxes. The y-axes in Figure 2.5 are drawn in logarithmic scale. The red dashed line in each subfigure represents the significance level α. Hence, the better



(a) Sliding up gestures



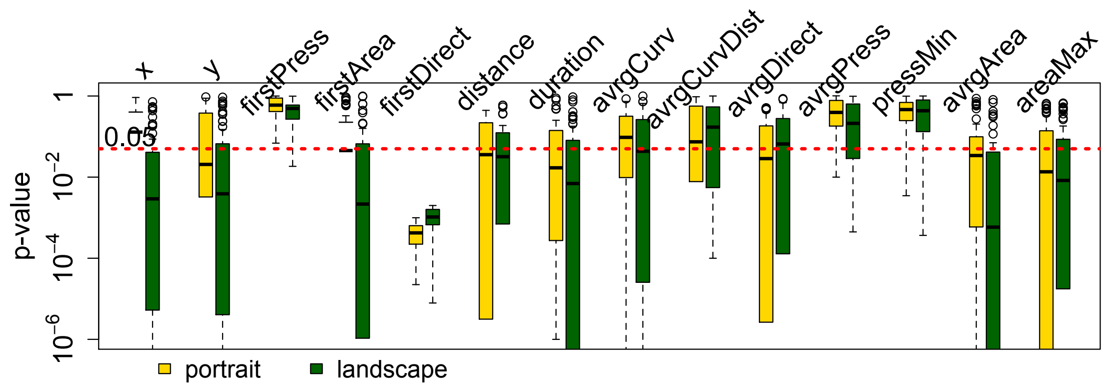
(b) Sliding down gestures



(c) Sliding left gestures

a metric is, the more portion of its box plot is below the red line. It denotes that more pairs are significantly different.

We initially thought touch pressures should be a good metric to distinguish different people. However, from Figure 2.5, we can see that none of the three pressure related

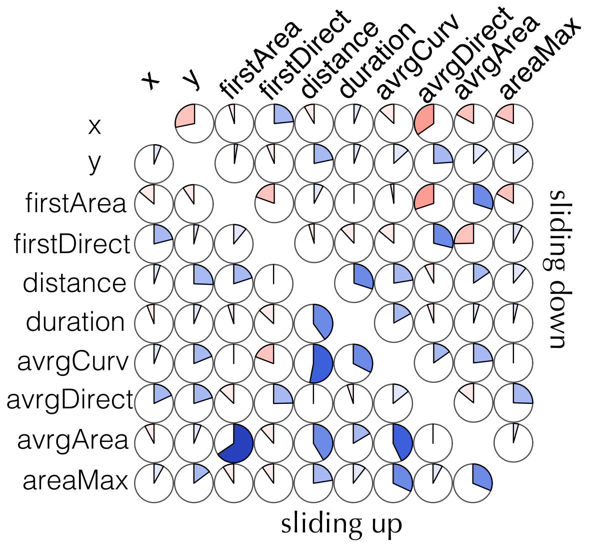


(d) Sliding right gestures

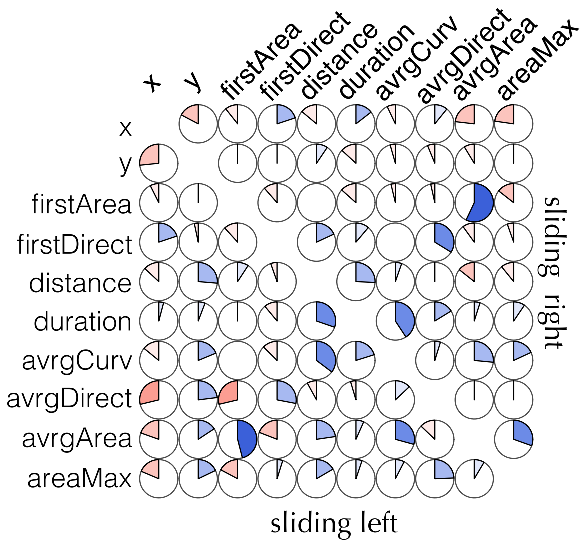
Figure 2.5: K-S Test on Sliding Metrics

metrics, first touch pressure, average pressure, and min pressure portion, is a good metric because at least half of their *p*-values are above the red line in all of the four subfigures. This means that the pressure data is bad in distinguishing two different persons. Besides, Figure 2.5 also shows that average curvature distance is a bad metric. The remaining metrics have most of their *p*-values below the red line, indicating that most data sets are significantly different to one another in the statistical sense. Therefore, we finally select first touch position, first touch area, first moving direction, moving distance, duration, average moving direction, average moving curvature, average touch area, and max-area portion as the metrics for sliding features.

Next, we tested the correlation between each pair of metrics. A strong correlation between a pair of metrics indicates that they are similar in describing a person's gesture pattern. In other words, a weak correlation implies that the selected metrics reflect the different characters of the desired gestures. For each user's gesture data set in one orientation, we calculated Pearson's correlation coefficient between each two metrics. Then, for each two metrics, we took the average of all resulting correlation coefficients between the two metrics. The average is taken over different users and different orientations. Figure 2.6 shows the resulting average coefficients. Each subfigure can be viewed as a 10 by 10



(a) Sliding up and sliding down gestures



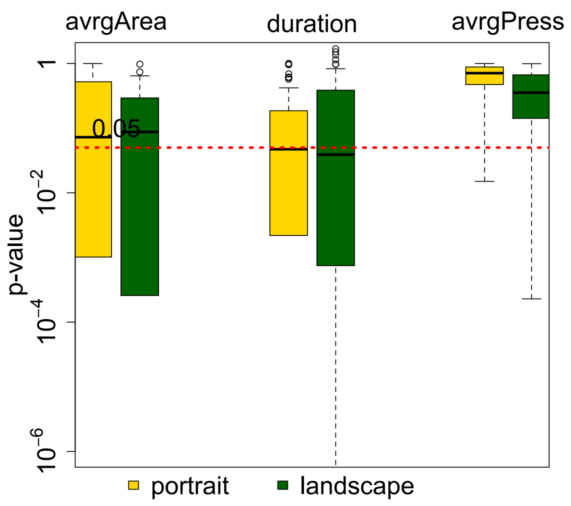
(b) Sliding left and sliding right gestures

Figure 2.6: Correlations Between Each Pair of Metrics of Sliding Gestures

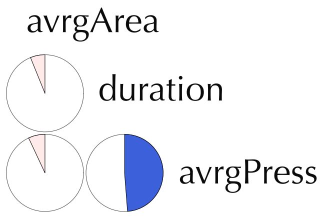
matrix and shows two sliding types using an upper triangle and a lower triangle, respectively. A pie chart in a triangle denotes the average correlation coefficient between the two metrics. The names of the metrics are listed on the top and the left sides. For a pie chart, blue represents a positive correlation and red represents a negative correlation. A larger shaded area in a pie chart indicates a stronger correlation. From the figure, we can see that most correlations are weak correlations and there are more positive correlations than negative correlations. We note that the correlation between the average touch area and the first touch area is remarkably positive in sliding up and sliding right. This is because people's first touch usually affects the remaining touches in a sliding gesture. If a person touches hard at first, it is quite possible that he will continue touching hard the rest of the sliding. However, we are not going to delete any of the two metrics because the correlation is not strong enough in sliding down and sliding left.

2.5.3.2 Tap Gesture

Tap gesture is a simple gesture. Hence, we do not design many metrics for it. Figure 2.7a shows the K-S test results on each tap metric. It is obvious that the average touch area



(a) K-S test results



(b) Correlations

Figure 2.7: K-S Test on Tap Metrics and the Correlations Between the Metrics

metric and the average touch pressure metric are not good in distinguishing users, because their medians are above the significance level. The median of *p*-values of the duration metric is just a little below the significance level. In summary, the tap metrics are not as good as the sliding metrics. The reason is that a tap gesture is usually so quick and simple that it provides few distinct features. The average correlation coefficients between every two metrics are shown in Figure 2.7b. We can see that the correlations between each pair of metrics are not strong, i.e. every coefficient is smaller than 0.5. Therefore, using tap gesture as a single gesture to re-authenticate a user is not reliable and may cause high error rates. Therefore, we propose to use tap as an auxiliary gesture. If the desired number of taps are captured, our system combines the tap feature vector together with a sliding feature vector to enhance the authentication accuracy.

2.5.4 Designing Modules

As illustrated in Figure 2.2, the predictor component contains several classification modules, each of which can independently re-authenticate a user. In Table 2.1, we list the gesture or gestures used in each module. In total, we have 16 classification modules in the

Table 2.1: Classification Modules and the Corresponding Gesture Types

PORTRAIT		LANDSCAPE	
No.	**gestures**	**No.**	**gestures**
1.	sliding up	2.	sliding up
3.	sliding down	4.	sliding down
5.	sliding left	6.	sliding left
7.	sliding right	8.	sliding right
9.	sliding up + tap	10.	sliding up + tap
11.	sliding down + tap	12.	sliding down + tap
13.	sliding left + tap	14.	sliding left + tap
15.	sliding right + tap	16.	sliding right + tap

predictor component – 8 modules for portrait mode and 8 modules for landscape mode. In each orientation mode, we use 4 modules to classify 4 sliding types. Another 4 modules are used to classify the combination of a sliding gesture and a tap gesture. Each module sends an alert to the OS if it finds the feature vector to be abnormal.

As pointed out in Section 2.5.2, a feature vector consists of the average metric values taken over a block of gestures. The block sizes are different for slidings and taps, denoted by n_s and n_t, respectively. For example, when a sliding up gesture is captured in portrait mode by the gesture monitor component, 10 metric values will be extracted and stored as a group. If there are already n_s such groups, average values are calculated by metric and fed into the classifier as a feature vector. The classifier uses module 1 to classify the feature vector. If it is an abnormal vector, an alert message will be sent out to the OS. If there is a new portrait tap feature vector ready as well, it will be combined with the sliding feature vector and the classifier will use module 9 instead. We emphasize that our system does not require the block of gestures to be temporally close to each other. In other words, any n_s sliding up gestures can be used to calculate a feature vector. This property of our system is important to smartphone usage because most people use smartphones intermittently. For example, a user may turn on his smartphone, check emails, and turn it off. There may be only a few sliding gestures in this operation cycle. Therefore, gestures are usually collected group by group and there may be a long time interval between two groups.

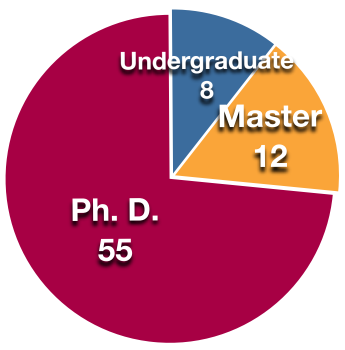
2.6 Evaluations

In this section, we are about to evaluate the performance of the proposed identity authentication scheme on a prototype. We first introduce the necessary details to set up the evaluations and show the evaluation results in the follow-up section.

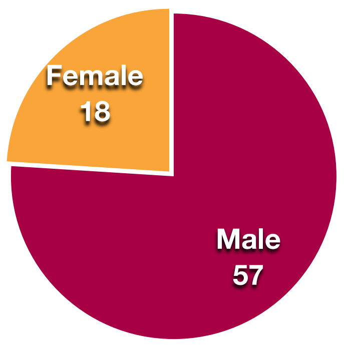
2.6.1 Setup

We used the SVM algorithm as the classification algorithm in the system and selected LIBSVM [26] as our implementation. For two-class classification tasks, the SVM finds a hyperplane in training inputs to separate two different data point sets such that the margins are maximized. A margin is the distance from the hyperplane to a boundary data point. The boundary point is called a support vector and there may be many support vectors. Sometimes, we need to map the original data points to a higher dimensional space by using a kernel function so as to make training inputs easier to separate. The kernel function used in our SVM is the Gaussian radial basis function $K(\mathbf{x}_a,\ \mathbf{x}_b) = e^{-\gamma||\mathbf{x}_a - \mathbf{x}_b||^2}$, where γ is equal to the reciprocal of the feature number. In our classification modules, we label the owner's data as a positive class and all other users' data as a negative class. SVM has been used in security area [105].

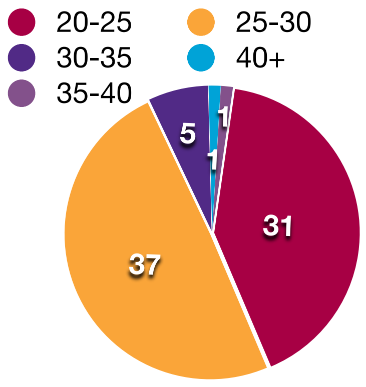
As described in Section 2.5.1, 75 people participated in our data collection. The participants are either students in or visitors to our lab building. We recorded the demorgraphics — education, gender, and age range — of the participants and show them in Figure 2.8. All our participants are older than 20 years old. Here, education is the highest degree that a participant has or is pursuing. The numbers in the pie chart are the numbers of the participants in the corresponding category.



(a) Education



(b) Gender



(c) Age

Figure 2.8: Demographics of the 75 participants

Among them, 28 people are *target users* who were asked to use the smartphones till at least 150 sliding up gestures, 150 sliding down gestures, 150 sliding right gestures, 150 sliding left gestures, and 300 tap gestures were collected. Other people, called *non-target users*, were asked to use the phone till the total using time hit 15 minutes. The target users are CSE graduate students at Arizona State University. The demographics of the target users are listed in Table. 2.2.

Table 2.2: Demorgraphics of the Target Users

	Category	# of users
Education	Master students	2
	Ph. D. students	26
Gender	Female	9
	Male	19
Age	20-25	10
	25-30	15
	30-35	3

In our experiments, we generated training and testing data sets for each target user. For a specific gesture type and a target user, the user's corresponding gesture data set was divided into two halves. In each half, we randomly selected a block of gesture data of necessary size, such as n_s for sliding up gestures, and calculated the feature vector. The vector was labeled as a positive feature vector. We generated training positive feature vectors using the first half gesture data set and testing positive feature vectors using the other half gesture data set. In order to generate negative feature vectors, we divided the remaining target users and the non-target users into two halves, respectively. The first half of the target users and the first half of the non-target users consisted of the training user pool. The remaining users consisted of the testing user pool. To generate a training (testing) negative class feature vector, we first randomly chose a user from the training (testing) user pool, then randomly selected a block of gestures from the user's gesture set, and finally calculated a feature vector, labeled as negative. We dropped a feature vector if the selected gesture block was previously used. Training feature vectors, including positive and negative ones, were put together in a training data set and testing feature vectors were in a testing data set. We remark that positive training and testing feature vectors were generated from two disjoint sets of gestures. For negative feature vectors, the users used to generate testing vectors are totally different from those used to generate training vectors.

Hence, in our experiments, the feature vectors used to test a classification module are never met by the module in its training phase.

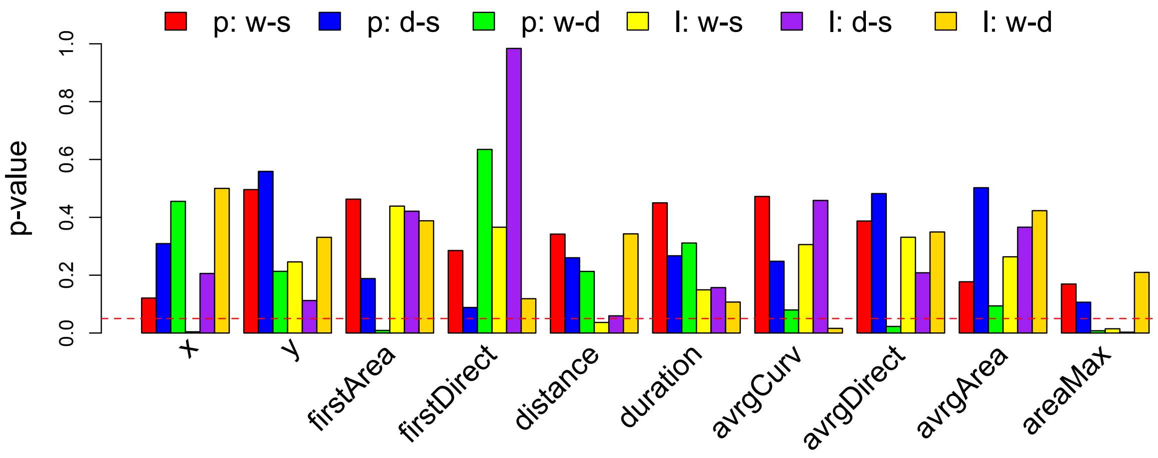
2.6.2 Experiment Results

In this section, we test the classification performance of our system under different system parameters. We also implemented the re-authentication module on a Motorola Droid smartphone to test its computation overhead.

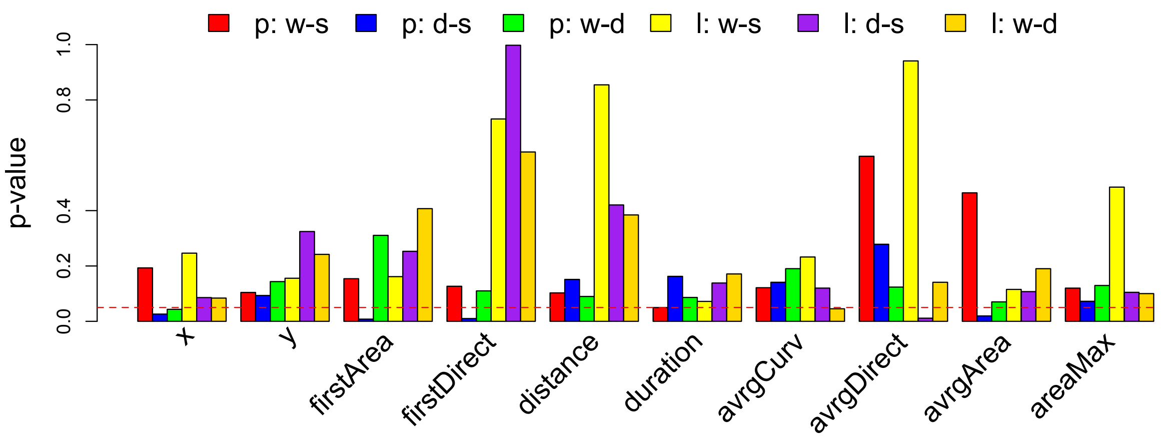
During data collection, we did not put any usage restrictions on the participants. Users were free to walk, sit, travel by vehicle, or perform other common activities, while they were using our smartphones.

2.6.2.1 Usage Environments

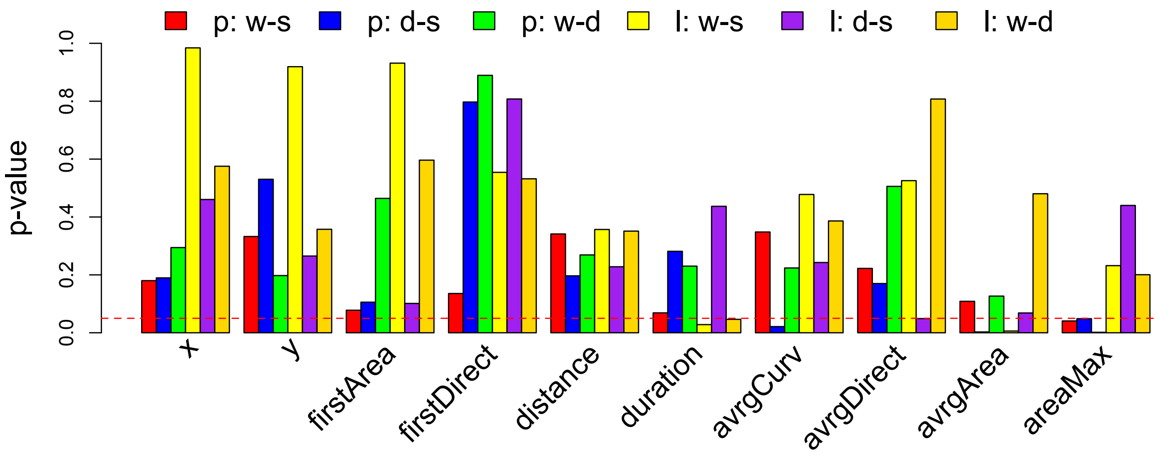
A smartphone may be used in different environments, such as in a moving vehicle. We are interested in whether a metric stays same in different usage environments, i.e., whether two metric data sets, which are obtained in two environments, are from the same distribution. We carried out the experiments in three normal usage environments, sitting, walking, and in a moving vehicle. We did not test some extreme usage environments, such as running, because users usually do not use their smartphones in such environments. We asked a volunteer to use the smartphone in each of the three environments, respectively, till the enough gestures were captured. For sitting and walking, the volunteer used the smartphone while he was sitting still or walking around. For the moving vehicle environment, the volunteer used the smartphone while he was sitting in the back seat of a moving city bus. Given a necessary metric, a gesture type, and a holing mode (portrait or landscape), we obtained a metric data set for each usage environment. We compared each pair of data sets by using K-S test, as introduced in Section 2.5.3, which results in a *p*-value for each



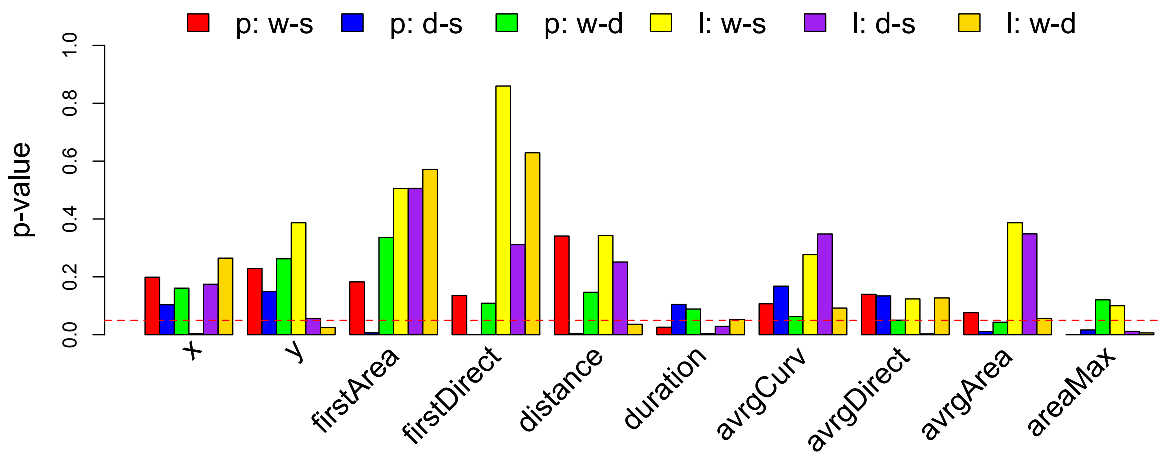
(a) Sliding up gestures



(b) Sliding down gestures



(c) Sliding left gestures



(d) Sliding right gestures

Figure 2.8: K-S Test on Sliding Metrics in Different Environments. p: portrait; l: landscape; w: walking; d: driving; s: sitting; w-s: w vs. s; d-s: d vs. s; w-d: w vs. d.

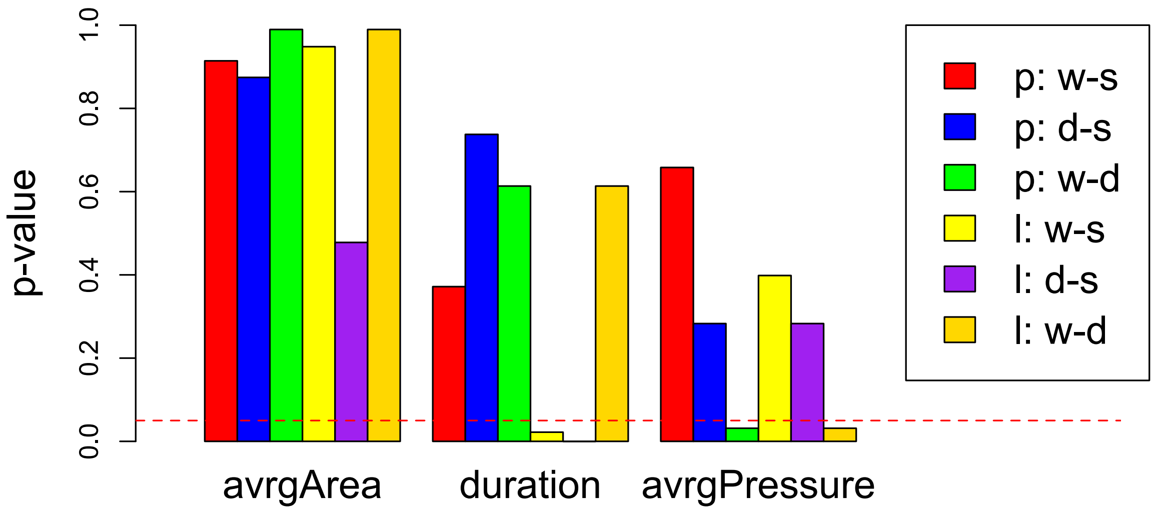


Figure 2.9: K-S Test on Tap Features in Different Usage Environments.

comparison. A higher p-value indicates the two data sets are more likely to be from the same distribution. When the p-value is smaller than a significant level $\alpha = 0.05$, the two data sets are thought to be from different distributions. The results for sliding gestures and tap gestures are shown in Figure 2.8 and Figure 2.9, respectively.

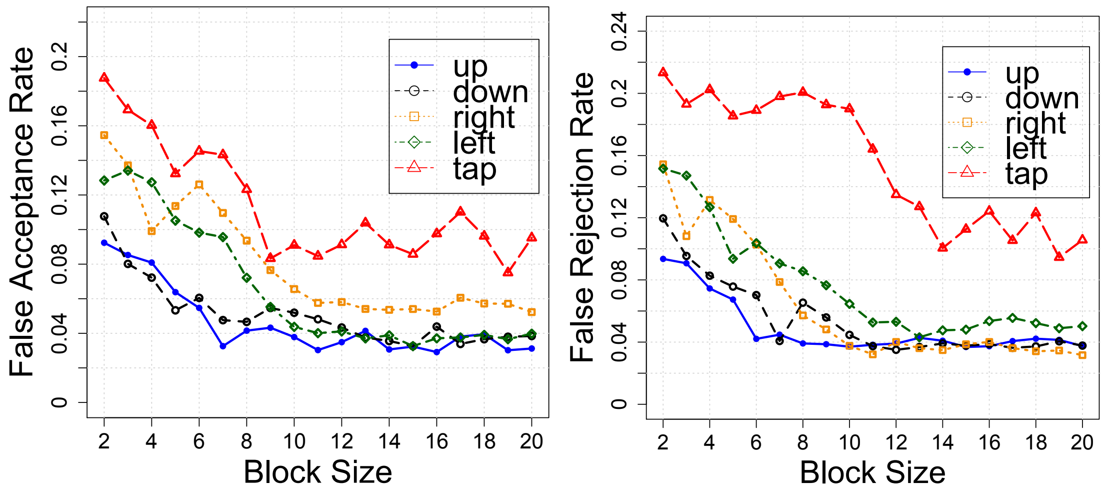
The red dotted line in each figure denotes the significant level $\alpha = 0.05$. From the figures, it can be seen that most *p*-values are greater than 0.05, indicating that the two metric data sets in those comparisons are from the same distribution. In other words, most metrics stay the same in different usage environments. This is because that users usually walk slowly while they are looking at smartphones and hold the smartphones more firmly while in a moving vehicle. Among four sliding gestures, the sliding right gesture is more vulnerable to the environment change, because it has 15 *p*-vales under the significant level. Tap gesture has only 4 *p*-values under the significant level.

In the data collection, we did not put any usage restrictions on the participants. A participant may carry out the collection in multiple usage environments. Therefore, the collected data is mixed with respect to the usage environments.

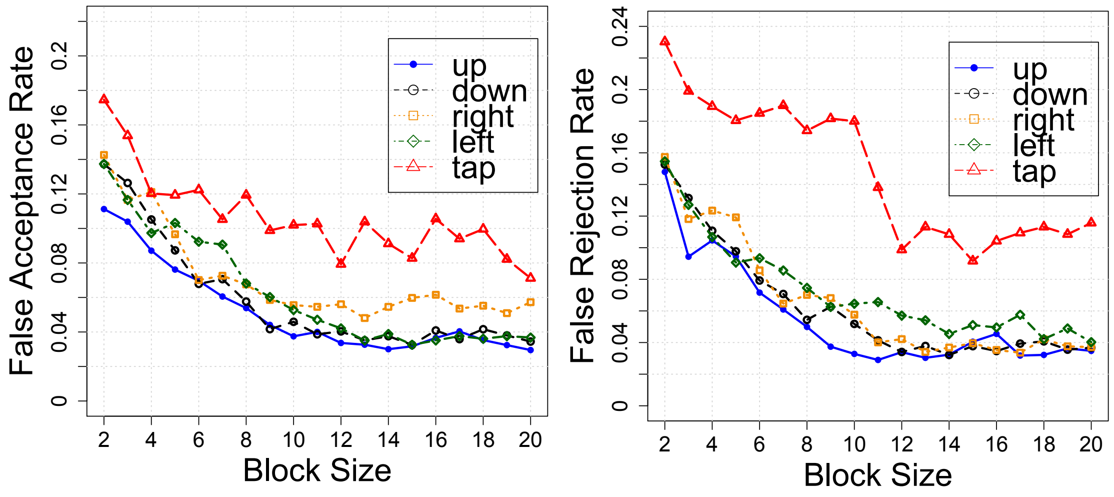
2.6.2.2 Gesture Block Size

A feature vector is calculated over a block of gestures. The block size is thus an important system parameter, which determines the number of gestures that our system needs to collect to perform a re-authentication. Hence, the size determines our system's re-authentication frequency. For each gesture type, we changed the necessary block size from 2 to 20. Given a block size and a gesture, for each target user, we generated 400 positive feature vectors and 400 negative ones in the training data set and the testing data set, respectively. We trained the classifier using a training data set, obtained a classification module, tested it using the testing set, and recorded false acceptance rates and false rejection rates. A false acceptance rate is the fraction of the testing data that is negative but classified as positive. A false rejection rate is the fraction of the testing data that is positive but classified as negative. In the sense of protecting smartphones, a false acceptance is more harmful than a false rejection. For each gesture type, we take the average false acceptance rates and the average false rejection rates over all target users' results. The results are shown in

Figure 2.10, which contains the two smartphones orientation modes, portrait mode and landscape mode. We note that, in this experiment, we took tap gesture as a single re-authentication gesture type and used its feature vectors to obtain a classification module in order to show its classifying accuracy. For each mode, we show the change of the average false acceptance rates and the average false rejection rates of each gesture type with increment of the block size. From the figure, we can see that, for a sliding gesture,



(a) Portrait mode



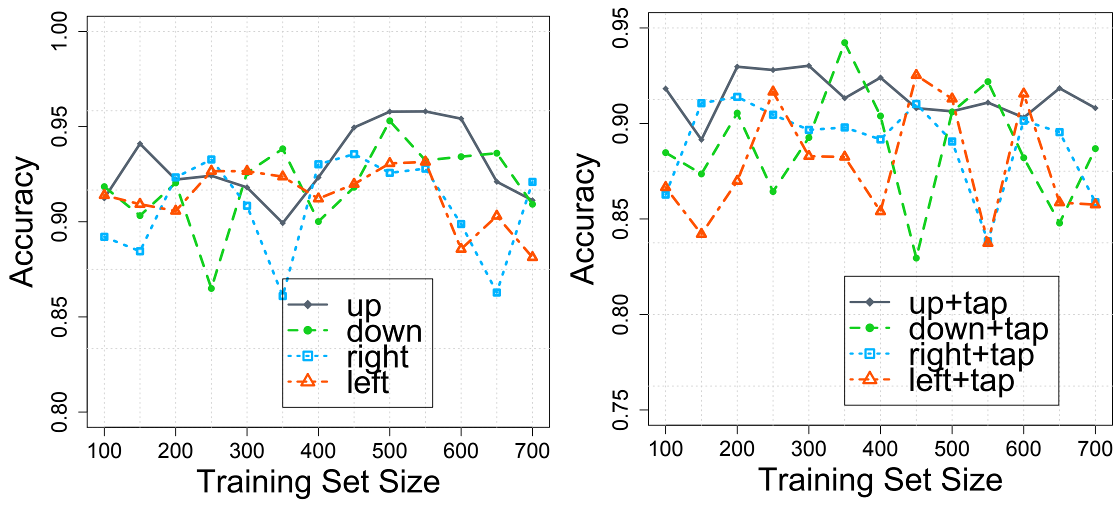
(b) Landscape mode

Figure 2.10: False Acceptance Rate/ False Rejection Rate vs. Gesture Block Size

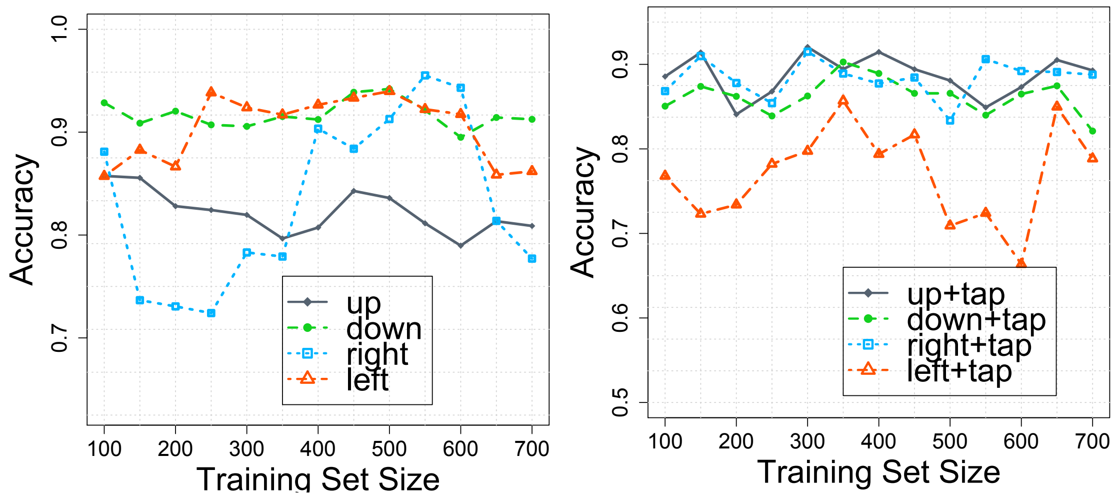
its false acceptance and false rejection rates get stable when the block size is greater than 14. In both modes, the two rates of tap gesture are approaching stable when block size is getting close to 20 although they are not as stable as sliding gestures. Among the five types, tap has the worst performance, having the highest false acceptance rates and false rejection rates in both modes. This also confirms our previous analysis of the tap metrics in Section 2.5.3.2.

2.6.2.3 Training Size

The size of a training data set affects a module's classification accuracy, because a larger training data set gives the classification algorithm more information. We tested the performance of each classification module under different training set sizes, from 100 to 700 at intervals of 100. In the feature generation, we selected block size $n_s = 14$ and $n_t = 20$ to generate feature vectors. Our system monitors both portrait mode and landscape mode in the background, using 8 classification modules for each mode (Section 2.5.4). Given a training set size and a classification module, for each target user, we used the approach introduced in Section 2.6.1 to generate a training data set and a testing data set. Each testing data set was of the same size as its corresponding training data set. For each training set size, we obtained 16 classification modules for each user. We tested each classification module and recorded the classification accuracy. Then for each module and each training set size, we took the average of all user's classification accuracies. The results are shown in Figure 2.11. From Figure 2.11a, we can see that when the training set size increases, the accuracy of a classification module first increases, approaches to a maximum point, and then decreases. We observe that the maximum point is around 500 for single gesture type modules and around 300 for combinatory gesture type modules. The same trend is observed on the results under landscape mode in Figure 2.11b. The observations indicate



(a) Portrait mode



(b) Landscape mode

Figure 2.11: Training Set Size vs. Classification Accuracy

that tap gestures provided extra useful information to a combinatory gesture type module and the module thus did not need more training data to learn a user's patterns. The accuracy decreases after the training set size passes the maximum point because a large training data set makes the module specific to the training data so that it makes more errors in prediction.

Besides, we list the average classification accuracy for each classification module in Table 2.3 when the training size is 500 for single gesture type modules and 300 for combinational gesture type modules.

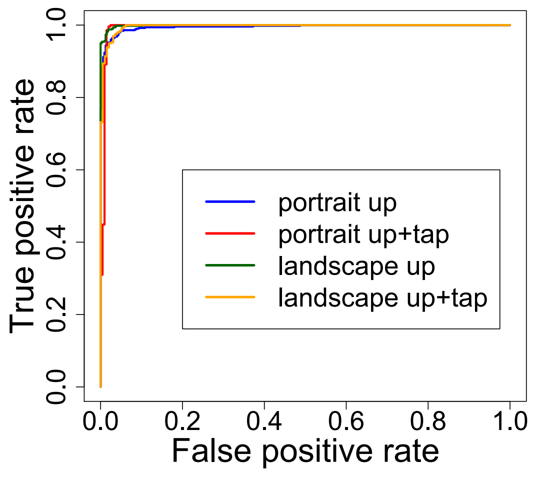
Table 2.3: Classification accuracy

PORTRAIT		LANDSCAPE	
gestures	**Accuracy**	**gestures**	**Accuracy**
sliding up	95.78%	sliding up	83.60%
sliding down	95.30%	sliding down	94.20%
sliding left	93.06%	sliding left	93.97%
sliding right	92.56%	sliding right	91.27%
up + tap	93.02%	up + tap	92.05%
down + tap	89.25%	down + tap	86.25%
left + tap	88.28%	left + tap	79.74%
right + tap	89.66%	right + tap	91.50%

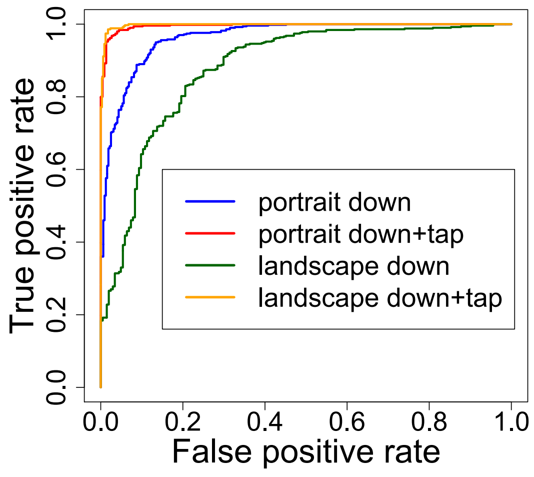
2.6.2.4 Using Tap

In the study of classification, the receiver operating characteristic (ROC) curve is a good way to graphically reflect the performance of a binary classifier. It is a plot of true positive rate $TP/(TP+FN)$ versus false positive rate $FP/(FP+TN)$. Here, TP, FN, FP, and TN are the number of true positive predictions, false negative predictions, false positive predictions, and true negative predictions, respectively. A true positive prediction is a correct prediction on a positive class. False positive, true negative, and false negative predictions are defined in the similar fashion. Generally, if a ROC curve is close to the top-left corner, it indicates the corresponding classifier can obtain a high true positive rate with a low false positive rate. Therefore, such a classifier is considered to be a good classifier.

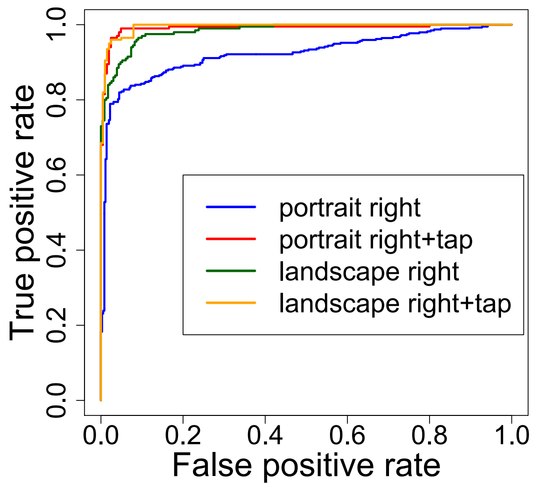
We fixed a target person and drew 16 ROC curves for his 16 classification modules. We set $n_s = 14$ and $n_t = 20$. The training data set size and the testing data set size were both 400 for all the 16 modules. The results are shown in Figure 2.12. The purpose is to test the improvement of using tap gesture as an auxiliary as well as the performance of each classification module. In all the plots, we can see that most modules having tap



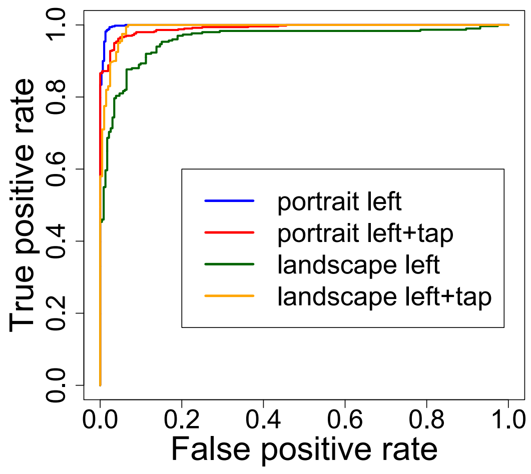
(a) Sliding up



(b) Sliding down



(c) Sliding right



(d) Sliding left

Figure 2.12: ROC Curves for 16 Classification Modules

as an auxiliary gesture perform better than the ones having only sliding gesture types. For example, in Figure 2.12a, two modules having tap gestures (red and yellow lines) are closer to the top-left corner than the other two lines. At the same time, we notice that sliding left performs better than “left+tap” combination in the portrait mode. Our single sliding gesture type modules also perform well in the two modes, since most of them are close to the top-left corner. Among the 16 modules, the classification modules for portrait and landscape sliding down gestures have the worst performance while the modules for sliding left gestures have the best performance. A possible explanation to this result is that people usually slide left for more contents while they are reading and they usually hold the contents, sliding slowly. In most cases, people slide down to get back to the top without reading. So they slide quick and slight. For a slow and “holding” sliding, the screen can sense more personally specific information, which leads to a better classification performance.

2.6.2.5 System Overhead

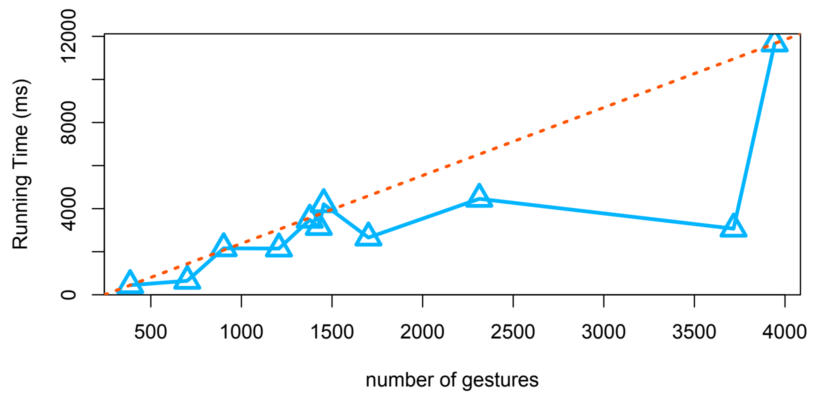
As shown previously, our system needs as less as 14 same type slidings to construct a feature vector. The following table (Tab. 2.4) shows the average time interval for each gesture type according to our collected user data. For example, on average, a sliding up gesture happens every 8.24 seconds in portrait mode. Therefore, our system can collect 14 portrait sliding up gestures in 115.6 seconds, which means the system can usually re-authenticate a user in 2 minutes using sliding up gestures. Learning an owner’s patterns is deployed on

Table 2.4: Average Time Interval for Gesture Types (Second)

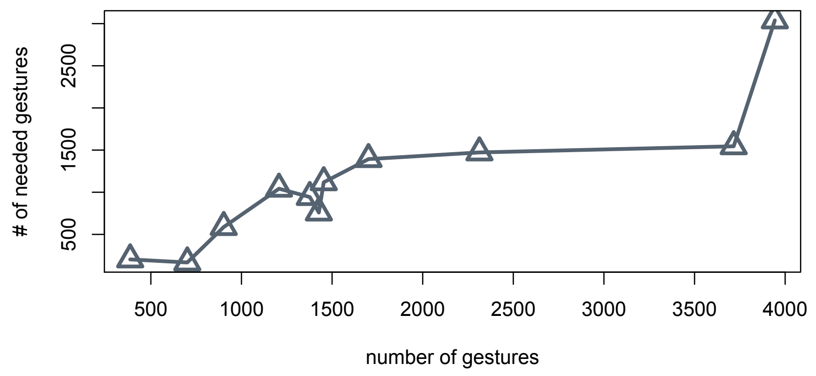
	up	down	left	right	tap
portrait	8.24	14.25	37.13	22.47	14.12
landscape	12.14	19.23	50.74	34.27	18.73

a PC in our architecture and performed offline. Feature extraction and classification is performed online by a smartphone, which directly affects the user experience of our system. Given a feature vector, the classification can be done in a short time because our feature vector is of small dimension. Particularly, given a combinatory feature with 13 feature values in it, our implementation only needed 17 milliseconds to give a prediction. The implementation used LIBSVM [26] on a Motorola Droid phone.

Feature extraction contains filtering necessary gestures and calculating each feature values, and thus takes more time. We tested our feature extraction scheme on a Motorola Droid smartphone with a single 550MHz A8 CPU, 256MB memory, 16GB sdcard, and Android 2.2 OS. We fed 386, 701, 902, 1207, 1377, 1427, 1454, 1701, 2313, 3716, and 3943 gestures to the preprocessing and the feature extraction modules on the smartphone. The running time and the number of filtered features are shown in Figure 2.13.



(a) Running time vs. total gesture number



(b) Filtered gesture number vs. total gesture number

Figure 2.13: Running Time of Feature Extraction

In practice, gestures will be fed to our monitoring component immediately after they are captured by the screen. In some cases, the OS may buffer the gestures and suspend our system for a while to run another high priority process. Since security is important in many cases, we assume that our system is not suspended for a long time so that the number of gestures it deals with at one time is within several hundreds. Looking at the second point in Figure 2.13, we can see that filtering 427 needed gestures from 701 gestures and extracting features from them takes only 648 milliseconds. We note that this process was carried out on a low-end smartphone and we can expect a dramatic performance enhancement on current main-stream smartphones.

2.6.2.6 Discussion

We carried out all our data collections and experiments on Motorola Droid phones, which are equipped with low-end touch screens. Therefore, some metrics may be dropped due to the smartphone's hardware limitations. For example, we left out the pressure related metrics because the touch screen did not provide accurate pressure measurements. The metrics may need to be carefully tested or even re-designed before deploying our system on another smartphone platform. For example, pressure may become useful and provide more user information on some smartphone platforms. However, our work provides a guideline for the metric design on other platforms and our methodology can still be adopted. Our work shows that using gestures to construct an unobservable continuous re-authentication on smartphones is practical and promising.

2.7 Conclusions

In order to prevent unauthorized usage, we have proposed a re-authentication system using user finger movement. The system performs continuous re-authentication and does not need human assistance during re-authentication. We have discussed biometric feature design and selection for finger movement. We have demonstrated the effectiveness and efficiency of our system in extensive experiments. This work has been published in [65].

Part II

NEAR FIELD AUTHENTICATION

CHAPTER 3

Finger Movement Based Near Field Authentication

This work is motivated by a common scenario of using smart devices, such as smartphones or tablets. Two people, say Alice and Bob, carry their smart devices and meet each other in a cafeteria. Alice is going to transfer some of their photos to Bob via the free cafeteria wifi. However, they want to do the transmission confidentially because the photos are private to them. Over the insecure public cafeteria wifi, Alice and Bob need to set up a one-time cryptographic session key to protect their communications, i.e. the photo transmission. In order to agree on a one-time session key, they should first invoke some key exchange (KE) protocol, such as Diffie-Hellman KE protocol. Without proper authentication, such a KE protocol is usually vulnerable to man-in-the-middle (MITM) attacks. Since Alice and Bob are meeting in person, they can carry out the KE protocol using a near field communication (NFC) system to defend against MITM attacks. An NFC system can only work within a distance less than a few centimeters. An MITM attacker is difficult to attack the communication carried out by an NFC system. However, current NFC systems rely on NFC chips, which are not available on many smart devices. In fact, only 20% of the smartphones on the market are expected to be equipped with a NFC chip by the end of 2014 [60].

The goal of this work is to design a system that is not dependent on NFC chips but able to authenticate whether two devices are in the near field via a insecure public network.

We call this kind of authentication *near field authentication* (NFA). The purpose of a near field authentication system is different from that of a traditional authentication system. Traditional authentication systems usually authenticate either the authenticatee's identity [62] or his ownership of certain secret information, such as passwords or cryptographic keys. NFA systems aim to verify whether the authenticatee, i.e. a smart device, appears in the near field. Traditional authentication systems usually leverage the information that exists or has been distributed prior to the authentication process. In contrast, an NFA process usually takes place impromptu, which requires an NFA system to bootstrap authentication from scratch.

With the widespread usage of smart devices, we will see a large number of near field applications. One example is pay-with-smartphones, which is currently one of the hottest applications on smartphone platforms. People store their credit card information in smartphones and make purchases by putting their smartphones close to a reader. Since the process is done in the near field, it is believed that the purchase is made by the smartphone owner himself. This idea has been realized by Google Inc. as Google Wallet on Android 2.3.3 or later using NFC chips. Another example is that users can store their flight tickets in their smart devices. When they check in at an airport, they just put the smart devices close to a machine that reads the ticket information and processes check-in automatically. Again, this application is only supported by devices having NFC chips. NFA plays a significant and fundamental role in these applications, because it provides authenticated and confidential communications between two devices. Therefore, in order to promote near field applications and let more smart devices benefit from this new technique, it is necessary to have a system that performs NFA on smart devices which do not have NFC chips.

The basic idea of a near field authentication is to compel two smart devices to appear together and stay close when the authentication is carried out. Many previous works

[1, 72, 96] have given initial solutions to this problem. BUMP™[1] is a representative system among these works. The purpose of BUMP is to provide a fast way to match two smartphones and then set up a secure data transmission channel. Their construction, according to [96], is first to bump two smartphones, then use the accelerometer in each smartphone to sense the force of the bump, and send the sensed results to a trusted server. The server compares every incoming data, matches the two smartphones, produces and sends the same session key to both if a match is found. However, using a centralized trusted center may not be suitable for our scenario, due to the single point of failure and the fact that Internet access may not always be available. For example, some tablets do not have cellular data services and cannot have access to the Internet in places that do not provide public Internet connections. Therefore, it is preferable to perform NFA over local networks, such as bluetooth or wireless LAN. Mayrhofer and Gellersen [72] proposed two authentication protocols with the purpose of pairing two devices. They used the accelerometer data that is sensed during the shake motion to create a session key for smart devices. In order to shake the devices for a few seconds, the device user needs to hold the two devices tightly in one hand. This limits their protocol application to the small size devices. Large or fixed devices, such as tablets and self-service check-in machines, cannot use their construction.

The idea of our near field authentication system is inspired by the observation that touch screens now are widely equipped by smart devices. Therefore, we propose to use people's on-screen finger movements to construct a near field authentication system. In order to force two smart devices to stay close to each other, we let a person move two fingers of one hand — usually the index finger and the middle finger — simultaneously on the two smart device screens, as illustrated in Fig. 3.1. The reason we use this motion is that it is easy and natural for people to perform and it produces many variations in terms of the sensed data. The more variations the data has, the more difficult it is for an attacker to

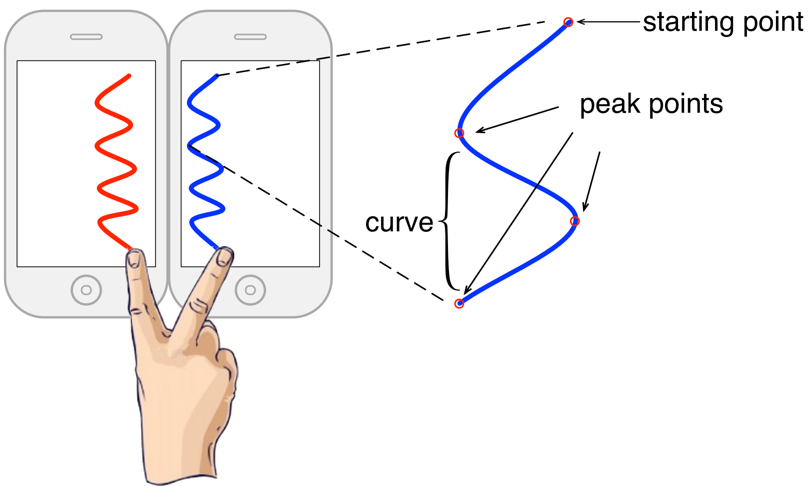


Figure 3.1: Zigzagging Finger Movements on Two Smart Device Screens

carry out a dictionary attack. Since the two finger movements are done by one hand, they are highly coherent to each other. We leverage this coherence to generate the session key for the two smart devices.

It is natural to assume that the zigzag on-screen finger movements cannot produce enough variations to fully defend against dictionary attacks. In other words, we believe that a particular person's finger movements may follow some pattern that can be used by an attacker to construct a dictionary to enhance the probability of successfully guessing the final session key. When data does not contain many variations, we say it is of low entropy. The low entropy of finger movement data gives rise to the first challenge for our system design. The second challenge is to find out which feature is suitable for finger movements so that the extracted data is robust to the small differences between two finger movements. Finally, even when the feature is robust, it is usually impossible to get two extracted data exactly the same. Therefore, a reconciliation approach between two smart devices is needed to make them agree on the same data. We make the following contributions in this study.

- We propose to use zigzag on-screen finger movements to perform near field authentication between two smart devices. Compared with the previous motion patterns, such as bump, shake, etc., finger movements are easier to carry out and provides better user experience. Another advantage is that finger movements are small and hard-to-catch motions. The movements are hard to be observed and emulated by a nearby attacker, which is a possible attack to the bump system [96].

- We design a robust feature so that two extracted feature data sets are similar to each other. Zigzagging finger movements provide many features, such as curvature, curvature distance, moving time, etc. We propose to use the time between the starting point and a peak point as the feature to be extracted in our system. The reason behind this choice is that people's finger usually moves slowly, or makes a short-time pause, when it turns at a curve peak point. Although the time is too short to be noticed by human eyes, it is long enough to be sensed by the touch screens. This makes the elapsed time of two corresponding peak points very similar to each other.

- We propose an efficient system to remove the differences between two extracted feature data sets and generate a high-entropy cryptographic key. We design an efficient approach using a private set intersection protocol to reconciliate the two feature data sets. As pointed out previously, the feature data is of low entropy. We use the encrypted key exchange technique to defend against dictionary attacks and generate a high-entropy key.

- Our system is efficient. The efficiency of our system is twofold: 1) it is easy and intuitive for people to use. 2) the computation overhead of our system is not heavy, which is demonstrated in our evaluation.

The rest of this chapter is organized as follows. We introduce related works in Section 3.1. We formulate the NFA problem and discuss the design goals in Section 3.2.

We present the design and constructions in Sections 3.3 and 3.4, respectively. We analyze the system in Section 3.5 and present the experiment results in Section 3.6. We conclude our work in Section 3.7.

3.1 Related Work

An NFA system is a kind of location limited system, but it puts more strict requirements on distance. In other words, when an NFA is passed, the two devices should be less than a few centimeters apart, i.e. they are in near field.

The importance of location enforcement in authentication was first realized by Stajano and Anderson in [94], in which users need to use a wired connection to complete the authentication process between two devices. Motivated by this work, Balfanz *et al.* [9] proposed the concept of *location-limited channel*, over which users carry out the authentication process. The transmission range of the channel in their work could be from centimeters to meters. However, the purpose of our work is to authenticate a device appearing in a distance less than a few centimeters. In addition, the authors of [9] used infrared as their location-limited channel, which is not available on most smart devices. The work of McCune *et al.* [73] and the follow up work of Saxena and Watt [88] used the smartphone's camera to scan the barcode displayed on the other screen or film a blinking light toward the other. Their protocols are directional, so a user has to execute the protocols twice to achieve mutual authentication, which can be done by our system in one execution. Claycomb and Shin [29] proposed to use audio transmission as a location limited channel and provide the key verification information through the audio channel to establish a secure channel between two smartphones.

Many other prior works used various sensors on smartphones to achieve location enforcement. The commercial app BUMP [1] uses the accelerometer to quickly match two smartphones for data transmission. As we introduced previously, BUMP system compares the data measured during a bump between two intended devices to match the two devices and provide confidential communications. The app is intuitive to use and needs less human involvement. Their system security and efficiency rely on the BUMP server, modeled as a trusted center. However, the server itself may suffer from single-point-of-failure problem [68]. For our scenario, the trusted center scheme is not a suitable choice either, because the Internet accessibility is not always available when people want to perform authentication on smart devices. Some works [25, 46, 52, 72, 96] proposed to use a different motion pattern — shake by a person — so as to achieve location enforcement. The accelerometer sensor was used to sense the motion once again. Mayrhofer and Gellersen [72] used a signal processing approach to remove the differences between sensed data sets and a key exchange protocol to generate a session key. The protocols in [72] do not need a trusted center. Although shake may provide the sensed data with more variations, it is difficult to hold and shake a big smart device, such as a tablet. Some other works proposed to use vibrations to transmit secret [45, 88]. The authors used the vibration that is produced by one device to encode the secret and the accelerometer of the other to sense and decode the secret. Recently, Studer *et al.* [96] proposed an MITM attack against those motion based approaches. They assumed that there is a powerful adversary who can observe the user's motion, such as shake or bump, so that he can emulate a similar motion pattern to carry out an MITM attack. While the success rate of their attack was sensitive to the delay induced by the attacker, their work did suggest that a smaller motion pattern is preferred than shake and bump when a third person is standing nearby.

In order to defend against MITM attacks, some prior studies suggested to use human comparison after the secret exchange, because an attacker cannot change the output on the device screen. Many works studied the comparison of a string or hexadecimal digits

[18, 59, 100]. To improve the usability, other works proposed to encode hexadecimal digits into a sentence [40] or an image [81]. However, string comparisons or picture comparisons require human intelligence and are error prone for people. Compared with these works, our system does not need manual comparison and is easy for people to use.

Establishing secure channels for two mobile devices without any pre-shared secret information has been widely studied in recent years. Usually, such a system needs an out-of-band (OOB) channel to transmit some secret information. While some applications seek to migrate or remove such an OOB channel, by using BAN logic, Claycomb and Shin [28] have shown that device authentication using a single channel is impossible.

3.2 Problem Formulation

In this section, we formulate the near field authentication problem.

3.2.1 System Model

The purpose of a near field authentication system is stated in the following definition.

Definition 1. *A near field authentication (**NFA**) system is a mutual authentication system between two parties. When the authentication is successfully passed, the system convinces both parties that they are separated in a distance less than a few centimeters. In addition, at the end of a successful authentication, an NFA system assigns the same cryptographic session key to both parties.*

The parties considered in this work are smart devices equipped with touch screens, such as smartphones, tablets, etc. For convenience, we use Alice and Bob (two human names) to denote these two parties, which are actually two devices.

An NFA system usually needs an approach to achieve location enforcement, forcing two devices to stay close. Some systems use the physical limit of special communication channels, such as NFC channels. Many other systems ask a person, called *conductor* in this paper, to achieve the enforcement. The conductor is one of the device owners and assumed to be trusted in this paper. This is a reasonable assumption because two persons who are willing to do confidential communications must trust each other in our scenario. BUMP system [1] lets a conductor use one smartphone to hit the other one to ensure that the two devices are both in the vision of the conductor. Our construction asks a conductor to slide two fingers of one hand over the two screens to achieve location enforcement.

3.2.2 Attack Model

The two smart devices involved in the authentication, including their executing applications, are considered trusted and not compromised. A near field authentication usually takes place in the presence of device owners. They will not perform the authentication process if they do not trust each other. Safeguarding a smart device against being compromised by an outside attacker has been well studied in the area of intrusion detection and is out of the scope of this work.

In this paper, we consider a fully malicious adversary, who controls public communication channels. The adversary is able to eavesdrop any communication. He is also able to tamper, delay, replay, inject and block any message. During a protocol execution, the adversary can carry out an MITM attack, impersonating one party to communicate with the other honest one. Such a powerful adversary is always able to destroy an ongoing authentication process by blocking all messages. However, such a rash attack exposes the presence of the adversary and does not bring back much benefit to the adversary. Observing the presence of an adversary in NFA, the device owners can stop further communications and move to a more secure place. The main purpose of an NFA system is to protect follow up communications, which are of more interest to the adversary. Therefore, **the purpose of the adversary** in this work is to obtain the session key generated by an NFA system without being captured, so that he can decrypt and obtain the succeeding communication messages.

3.2.3 Design Goals

In addition to defending against the adversary modeled in Sec. 3.2.2, we expect an NFA system to achieve the following design goals.

- *It needs less human involvement.* System users prefer to do simple and intuitive tasks. An NFA System will not be widely accepted, if it needs people to do more involved works, such as string comparisons.
- *It is intuitive to use.* A system should be designed in a way that is intuitive and easy for people to learn how to use.
- *It does not rely on prior knowledge.* Near field authentication and follow up communications usually take place impromptu. Hence, the two involved parties cannot be assumed to share any prior information.
- *It is decentralized.* An NFA is expected not to rely on any trusted center, which suffers from a single-point-of-failure problem.
- *It is localized.* An NFA system should still work when the Internet is not accessible. For example, some companies limit the Internet access for safety reasons. In such cases, an NFA system should make use of local networks, such as bluetooth

or WLAN. This goal also makes the centralized architecture unsuitable for an NFA system.

3.3 Design of An NFA System

3.3.1 System Overview

Fig. 3.2 shows the architecture of our NFA system. Our system uses human finger move-

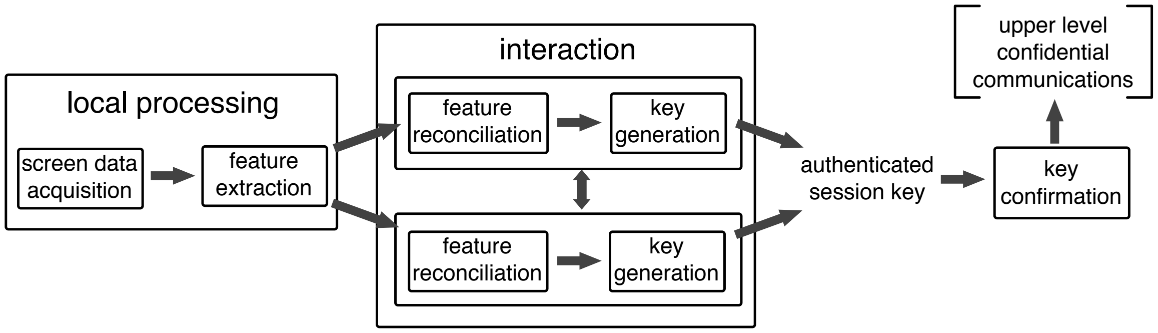


Figure 3.2: The Architecture of Our NFA System

ment on touch screens to assist authentication, taking advantage of the physical closeness of two fingers to enforce the spatial closeness in NFA. To the best of our knowledge, we are the first to use finger movement or touch screen input as a location limited channel. Compared with other motion patterns, such as bump, shake, etc., finger movement has the following merits.

- It provides more robust similarities.
- It is easy to carry out, especially on devices that are difficult to hold in one hand.
- It is hard to be observed and copied by a nearby attacker. Current studied motion patterns, such as bumps and shakes, require a quite noticeable motion, which can be

easily captured by a nearby attacker. Observing these motion patterns, an attacker can launch a better dictionary attack, as described in [96].

In our system, when a conductor triggers an authentication process, the two devices sense finger movements on their individual touch screens and extract the feature data locally. Using the extracted data, the two devices interact with each other to generate the same session key. Finally, the two devices verify that the generated session keys are identical before they use them to protect succeeding communications.

3.3.2 Feature Design

Our system asks a conductor to use two fingers, from the same hand, to do a "zigzag" movement on two touch screens (see Fig. 3.1). A zigzag movement forms a series of curves, which provide many features to be extracted. The features of a curve have been studied by Zheng *et al.* in [108] for mouse movement. They recommended to use angle of curvature, curvature distance, and moving direction to characterize a curve movement. However, we found, by experiments, that these spatial features are sensitive to finger movements. A small deviation of two finger movements may cause a big difference between the resulting feature values.

We propose to use a new temporal feature — the time between a peak point and the starting point, i.e., the peak point's *elapsed time*. As shown in Fig. 3.1, the starting point is the point where a finger first touches the screen and peak points are the points where the finger moving direction changes. The robustness of this feature is due to the fact that people usually make a pause when their fingers turn above the screen. Although the pause usually takes only about 0.04 second, it is long enough to be captured by a touch screen and can be easily detected during a follow up data processing. Another observation is that two fingers may slide at slightly different speeds on two screens but they usually turn at the same time. This makes peak point's elapsed time a robust feature for an NFA system. A time value collected by touch screen is accurate to 10^{-6} second, but such a high accuracy also causes high sensitivity and low robustness. Hence, we round a time value to the nearest decimal fraction with 2-digit fractional part. Finally, we drop off the decimal separator to make the value an integer. For example, given a time value of 1.426478, we use 143 as a feature value. Hereafter, we mean the processed time value when we say "time value".

3.3.2.1 Variations

A feature with more variations is more difficult to be guessed by attackers. The variations of our feature can be reflected by the distribution of time intervals between each pair of contiguous peak points. Fig. 3.3 shows the histogram of the time intervals in a person's zigzag finger movement on one screen. The top-right sub-plot shows a "zoom-in" histogram of

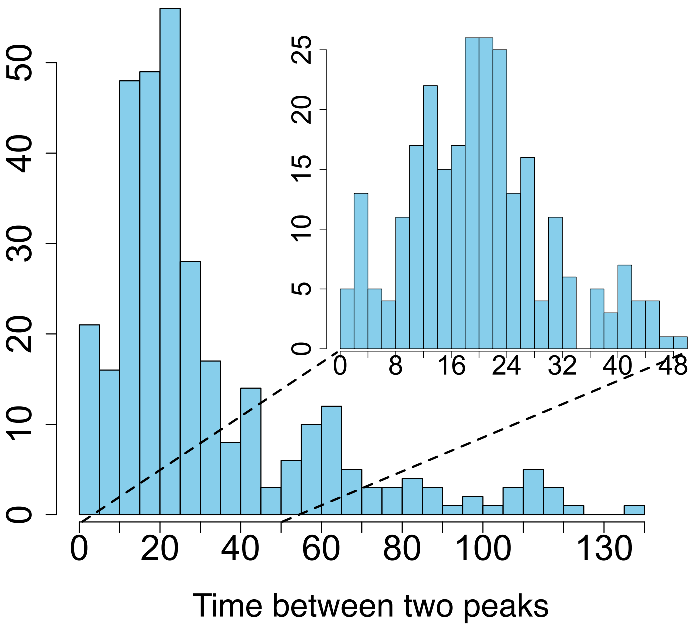


Figure 3.3: Features From One Person

the time intervals between 0 and 50. We note that the bin width is different in the zoom-in histogram. The bin width in the big plot is 5 units and the one in the zoom-in plot is 2 units. The data collection was carried out on two Motorola Droid smartphones running Android 2.2 OS. From the figure, we can see that the time intervals distribute from less than 0.1 second (10×10^{-2}) to around 1.4 seconds. This shows that the variation of a person's feature value is large. We can also see that many values congregate into the interval between 10 and 25. The sub-plot shows that the values distribute almost evenly on each bin between 8 and 24. This shows that this person's finger movement has some pattern: the time that he uses between two peak points is mostly between 0.08 and 0.24 second.

3.3.2.2 Similarity

Feature values collected from two devices are expected to be very similar to each other, because it saves the time of removing the differences between two sensed feature data sets in the feature reconciliation phase. A person was invited to slide fingers zigzag on two devices, which recorded two sequences of peak points, respectively. We call a peak point in one sequence and its counterpart in the other a *pair*. We computed every peak point's elapsed time. Finally, we calculated the absolute difference between the two elapsed time values in each pair. A set of such absolute difference values can be viewed as a measurement of the similarity between the two sensed feature data sets. The distribution of the absolute differences is shown in Fig. 3.4. The experiments represented by the yellow bars were carried out on two Motorola Droid smartphones, while the experiments represented by the green bars were carried out on two different smart devices — a Motorola Droid smartphone and an HP TouchPad™tablet. For comparison purpose, we also calculated the difference sets, as done previously, for another two spatial features, curvature angle and curvature distance, proposed by and defined in [108]. Due to space limitation, we refer readers to [108] for details. We can see that most differences between two corresponding

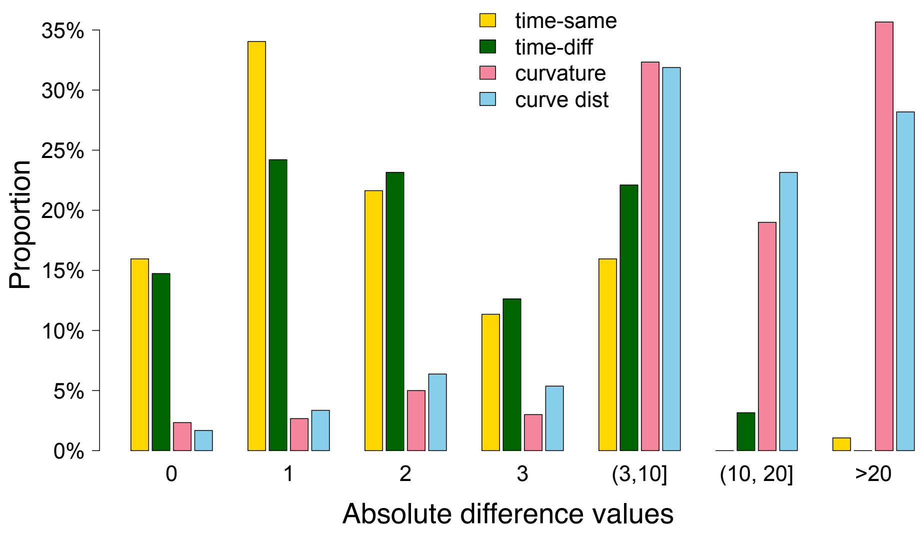


Figure 3.4: Similarity of Our Designed Feature

elapsed time values are less than 3, indicating that the two feature data sets are very similar. Specifically, 82.97% of all absolute differences obtained from the experiments on the same device type (yellow bars) are less than 3. This percentage is 74.73% on the different device types (green bars). At the same time, most absolute differences of another two feature sets are greater than 3 and many of them are greater than 20.

3.4 System Construction

We are now ready to present the construction of our system in details. Our system consists of a conductor and two smart devices. In the following description, one device is denoted by Alice and the other is denoted by Bob. There is no difference between these two roles, because our authentication is mutual. In our motivating case, the data sender is Alice and the data receiver is Bob. We assume that Alice and Bob are able to set up a communication connection through a local network, such as wifi LAN or bluetooth network. This connection is public and insecure. Our system helps Alice and Bob to authenticate each other in near field and establishes a secure connection by assigning each of them the same session key.

We say the two smart devices are *ready*, when our NFA system is launched on both devices and a communication connection has been launched between them. The two devices then stay monitoring the screen touch and waiting for the conductor's finger movements.

Our system consists of three phases: system setup and feature reconciliation, key generation, and key confirmation. Figure 3.5 shows the system setup and feature reconciliation phase. In this phase, we use two hash functions: $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$ and $H_2 : \mathbb{G}^2 \rightarrow \{0,1\}^l$, where l is determined by the security level. Here $\mathbb{G}$ is a multiplicative group of a prime order. Hash function H_2 takes two group elements and outputs an output of fixed length. In practice, H_2 can be constructed from any cryptographic hash function H', which takes an input (of arbitrary length) and hashes to a fixed length output. For example, SHA-1 is such a hash function, if the security level is less than 128 [11]. To construct H_2, we concatenate the two inputs together and pass into H'. Construction of H_1 is special, since it hashes the input to a group $\mathbb{G}$, not a fixed length output. It is constructed from a cryptographic hash function $H'' : \{0,1\}^* \rightarrow \{0,1\}^{l_1}$, where $l_1 = \log q$ and q is the group order. The idea is to hash the input to an element h in $\mathbb{Z}_q$ and the output of H_1 is g^h, where g is a generator of $\mathbb{G}$. However, an output of H'' may exceed the range of $\mathbb{Z}_q$. A standard technique is to re-hash $H''(x||1)$ and verify whether the new output is in $\mathbb{Z}_q$. Repeat for k times ($H''(x||k)$ for k-th re-hashing) till the output is accepted. Since the probability that an output exceeds $\mathbb{Z}_q$ is less than 1/2, the failure probability decreases exponentially with the repeat times. For example, we get an overall failure probability less than 0.1% if we choose $k = 10$.

Feature reconciliation is to remove differences between Alice's and Bob's sensed data sets. First, Alice and Bob choose m pairs of elapsed time values. Since the difference

When the two devices are ready, a conductor puts the two smart devices side by side, slides his two fingers of one hand zigzag on the two screens, and then triggers the authentication process. The two smart devices, Alice and Bob, sense and calculate two elapsed time sets $\mathscr{A} = \{a_1, a_2, \cdots, a_n\}$ and $\mathscr{B} = \{b_1, b_2, \cdots, b_n\}$, respectively. Here, n is the number of peak points.
Feature Reconciliation

1. Given a security level κ, Alice generates a cyclic group $\mathbb{G}$ of prime order q, written in a multiplicative notation, and sends it to Bob. Alice randomly selects m elements $a_{t_1}, a_{t_2}, \cdots, a_{t_m}$ and sends $t_1, t_2, \cdots, t_m$ to Bob.

2. For every t_i, $1 \leq i \leq m$, Bob selects $\alpha_i \leftarrow_R \mathbb{Z}_q$, calculates $h_i = H_1(b_{t_i}), y_i = (h_i)^{\alpha_i}$, and finally sends y_i's to Alice.

3. Alice selects a random number $k \leftarrow_R \mathbb{Z}_q$. For every y_i, $1 \leq i \leq m$, Alice calculates $z_i = (y_i)^k$. Alice prepares a non-interactive zero knowledge proof π for the knowledge of k s.t. $\forall_{i=1,\cdots,m}\ z_i = (y_i)^k$. Alice sends z_i's to Bob along with proof π.

4. Bob aborts if π is not verified. Bob calculates $x_i = H_2(h_i, (z_i)^{1/\alpha_i})$ for $1 \leq i \leq m$. Let $\mathscr{X} = \{x_1, \cdots, x_m\}$.

5. Alice extends $\{a_{t_1}, a_{t_2}, \cdots, a_{t_m}\}$ to set $\mathscr{A}' = \{a_{t_i} + 3, a_{t_i} + 2, a_{t_i} + 1, a_{t_i}, a_{t_i} - 1, a_{t_i} - 2, a_{t_i} - 3\}_{1 \leq i \leq m}$. For each element a'_j in the set, Alice calculates $u_j = H_2(H_1(a'_j), (H_1(a'_j))^k)$. Alice randomly permutes all u_j's and sends the final set $\mathscr{U}$ to Bob.

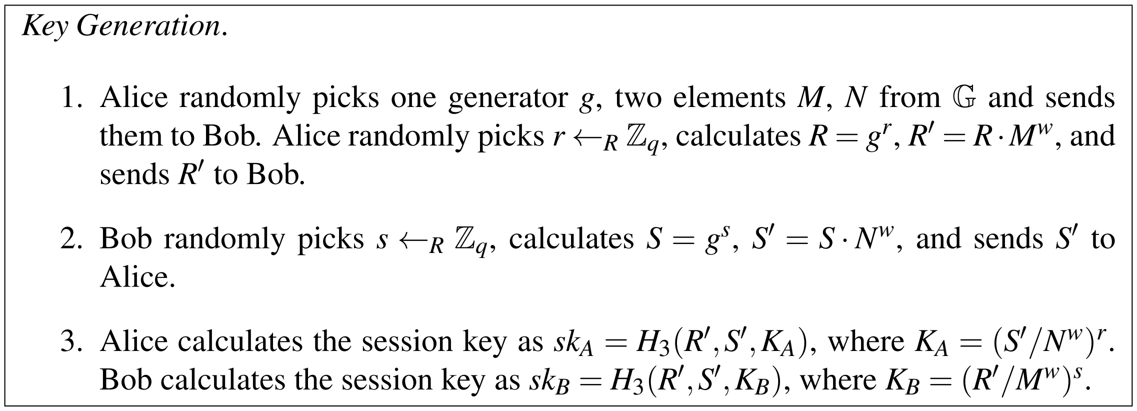
6. Bob calculates set $\mathscr{V}_b = \{b_{t_i} | \ (1 \leq i \leq m) \wedge (x_i \in \mathscr{U} \cap \mathscr{X})\}$. If $|\mathscr{V}| < 4$ and it is their first execution, Bob informs Alice and they re-execute step 1-6. If $|\mathscr{V}| < 4$ and it is their second execution, this authentication fails; Bob informs the conductor the failure reason and asks the conductor to start over the authentication. Otherwise, Bob sends set $\mathscr{V}_x = \{x_i | b_{t_i} \in \mathscr{V}_b\}$ to Alice. By comparing $\mathscr{V}_x$ against $\mathscr{U}$, Alice also learns set $\mathscr{V}_b$.

Alice and Bob now have the same data set $\mathscr{V}_b$. Alice (Bob) sorts the elements in the non decreasing order, represents them in binary strings, and concatenates them together in a string, which is denoted by w hereafter.

Figure 3.5: System Setup and Feature Reconciliation

between the two values in most pairs is less than 3, we let Alice extend her selected set by including $a \pm 1$, $a \pm 2, a \pm 3$ for every a in the set (step 5). Then, Alice and Bob privately find the set intersection. We use a private set intersection protocol proposed by [51] to let both parties find the same intersection set $\mathscr{V}_b$ (step 6). To get enough randomness, we need at least 4 feature values since each value can provide nearly 6 bits. If the found set size is too small, we simply let Alice and Bob redo the reconciliation phase. If they still cannot find enough values, this authentication fails and we let the conductor start over the whole authentication again.

In step 3 of the first phase, Alice needs to generate a non-interactive zero knowledge proof π of knowledge of k s.t. $\forall_{i=1,\cdots,m} z_i = (y_i)^k$. Although general zero-knowledge proof systems are currently thought to be inefficient, the zero knowledge proof system on discrete logarithm has been well studied and there exists an efficient non-interactive proof system [8]. Alice randomly chooses $t \leftarrow_R \mathbb{Z}_q$ and calculates $c = H(z_1||\cdots||z_m||y_1||\cdots||y_m||y_1^t||\cdots||y_m^t)$ and $s = (t - ck) \mod q$, where H is a cryptographic hash function. The proof π is tuple $<c, s, z_1, \cdots, z_m, y_1, \cdots, y_m>$. To verify the proof, Bob checks $c = H(z_1||\cdots||z_m||y_1||\cdots||y_m||y_1^s z_1^c||\cdots||y_m^s z_m^c)$.


Key Generation.

1. Alice randomly picks one generator g, two elements M, N from $\mathbb{G}$ and sends them to Bob. Alice randomly picks $r \leftarrow_R \mathbb{Z}_q$, calculates $R = g^r$, $R' = R \cdot M^w$, and sends R' to Bob.
2. Bob randomly picks $s \leftarrow_R \mathbb{Z}_q$, calculates $S = g^s$, $S' = S \cdot N^w$, and sends S' to Alice.
3. Alice calculates the session key as $sk_A = H_3(R', S', K_A)$, where $K_A = (S'/N^w)^r$. Bob calculates the session key as $sk_B = H_3(R', S', K_B)$, where $K_B = (R'/M^w)^s$.

Figure 3.6: Key Generation

When the same intersection set is obtained by both parties, the binary strings of its elements are concatenated together in a string w. Since w may be of low entropy, we propose to use encrypted key exchange approach [5] to leave attackers no choice but doing online dictionary attacks, which are easily captured by honest parties. It is a variation of the original Deffie-Hellman key exchange protocol [95]. The improvement is for defending against MITM attacks, which can break down the original DH protocol. In the protocol, a session key is the hash of all the previous messages and the secret string K_A or K_B. In this way, an MITM attack will be effectively defended because two honest parties obtain different messages in an MITM attack. Also, an offline dictionary attack becomes hard because all the intermediate messages are randomized and the final session key is also randomly selected through the hash function.

Key Confirmation.

1. Alice generates a nonce c and sends $\mathscr{C}_A = Enc_{sk_A}(c)$ to Bob. Bob generates a nonce d and sends $\mathscr{C}_B = Enc_{sk_B}(d)$ to Alice.
2. Upon receiving $\mathscr{C}_A$ from Alice, Bob decrypts it under key sk_B, obtains c', increases it by one, re-encrypts the result $\mathscr{C}'_A = Enc_{sk_B}(c'+1)$, and sends the ciphertext back to Alice. Similarly, Alice sends a ciphertext $\mathscr{C}'_B$ back to Bob.
3. Upon receiving $\mathscr{C}'_A$, Alice decrypts it, obtains c'', and passes the key confirmation if $c'' = c+1$. Similarly, Bob checks whether $d'' = d+1$.

Figure 3.7: Key Confirmation

Fig. 3.7 shows the construction of the key confirmation phase of our NFA system. Here, we use another hash function $H_3 : \mathbb{G}^3 \to \{0,1\}^l$. The construction of H_3 is the same as that of H_2. This phase is to let two parties explicitly confirm that the generated keys are same. We note that a nonce is a random number, the length of which is determined by the security level. For example, if the security level is 80 bit, we will choose a 80-bit random number. In some upper protocols, such as confidential instant message protocols, this phase is not necessary and the confirmation can be done in a implicit way. Bob sends Alice a hello message that is encrypted using his session key. If Alice decrypts the ciphertext and finds the plain text is of no meaning, she will be aware that they must have different session keys. However, this phase becomes necessary for those applications whose goal is solely to perform a near field authentication.

3.5 System Analysis

3.5.1 Performance Analysis

Our NFA system is a probabilistic algorithm because it is based on users' biologic data and may encounter some outliers. Although our design can remove some small differences, our system may still fail when most selected data are very different from its counterpart. Given a pair of peak points, if their value difference is no more than 3, we call them a *valid pair*. We have shown in Sec. 3.3.2.2 that it is with high probability to select a valid pair. Particularly, more than 70% pairs in our experiments are valid. The failure probability of our system is also affected by the number of the selected elements, m. Generally speaking, the more elements we select, the more probably we can find enough elements.

Theorem 1. *The feature reconciliation phase finishes successfully with a high probability, if $m \geq 7$ and the proportion of valid pairs is more than 70%.*

Proof. Because the pairs are randomly selected, testing the validity of a selected pair is a Bernoulli trial with success probability p. Therefore, the probability that 4 out of m pairs are valid is

$$P = 1 - \sum_{k=0}^{3} \binom{m}{k} (1-p)^{m-k} p^k$$

Plug in $m = 7$ and $p = 70\%$, we obtain $P = 87.40\%$. If the first attempt fails, the phase lets two parties redo the reconciliation using another set of randomly selected pairs. Therefore, the overall success probability of the feature reconciliation phase is $P + (1-P)P = 98.4\%$.

□

Our system is based on a conductor's zigzag finger movement and thus does not need any prior knowledge. The conductor only needs to slide fingers and trigger the authentication process. Therefore, our system is intuitive for people to use and does not need any involved human assistance, such as comparing two strings. Our system is fully decentralized and all computations are done between two devices. It does not need any third trusted center or server.

The feature reconciliation phase performs $O(m \log q)$ group multiplications, where q is group order. The key generation phase needs $O(m \log q)$ group multiplications. It is spent on computing M^w and N^w because the length of w is $O(m)$. The key confirmation is done in constant time.

3.5.2 Security Analysis

Most security threats to an NFA system come from MITM attacks and dictionary attacks. In the first phase, the private set intersection protocol guarantees that only a set provider can learn the final intersection set and nothing beyond the intersection. In the second phase, if an MITM adversary changes any intermediate message, Alice and Bob will derive different session keys. This alerts both parties to the existence of an adversary. Further more, if the adversary does not have the intersection set, he cannot get the keys derived by Alice and Bob.

Someone may be concerned with replay attacks. If a conductor uses his finger movements to generate the session key for a communication between his and an attacker's smart devices, the attacker may use the elapsed time set and carries out an MITM attack when the conductor tries to generate another session key between his smartphone and a third user's. The success of such an attack is based on the assumption that the conductor's finger movements in two sessions are very similar. However, this assumption can be easily broken when the conductor intentionally does not use the same finger movement pattern. For example, the conductor can randomly pause during the motion to break the pattern. Or, we can compare the current elapsed time set against the previous ones and ask the conductor to re-do the authentication and change the pattern if a similarity is found.

Mimicry attack is such an attack in which an attacker observes the conductor's finger movement in some way, such as video, and tries to guess the generated session key. First, this attack has to be taken on the fly, because any off-line dictionary attack will be defended by the password based authentication key exchange protocol (PAKE). Secondly, the attacker has only one chance to carry out the attack, since any more attempt will be noticed by the PAKE protocol. Besides, the elapsed time of the mocked finger movements have to be as precise as 1 milliseconds. This limitation makes mimicry attacks extremely difficult.

In the traditional private set intersection protocol, an adversary may impersonate one party in the protocol and generate a set of all possible values so that he can infer the honest party's whole input set. However, the feature reconciliation phase restricts the set size, m for Bob and $7m$ for Alice, to make it difficult for an adversary to enumerate all possible values. Offline dictionary attacks to the key generation phase is also impossible because r, s, M, and N are randomly selected. For an example, R' can be viewed as an encryption of message M using key w. Given $R' = R \cdot M^w$, an adversary iterates every possible key w to decrypt the ciphertext. However, since the message M itself is chosen randomly and of no meaning, the adversary cannot verify the validity of the obtained plain text. An adversary can carry out online dictionary attacks, guessing a w and interacting with an honest party. But every failure will be captured by an honest party.

Our system uses two built-in blocks, a private set intersection protocol [51] for feature reconciliation phase (Fig. 3.5) and an encrypted key exchange protocol [5] for key generation phase (Fig. 3.6). The two protocols have been proved to be secure against a malicious adversary in random oracle model by [5] and [51], respectively. The random oracle model is a security proving framework that assumes the existence of a random oracle mapping each request value to a random value. This oracle can be accessed by all protocol participants, including the adversary. The hash function is usually modeled as the random oracle in the proof. Following the same idea, we will see that the feature reconciliation phase and key generation phase of our system are secure. One thing we need to point out is step 6, where we connect the two built-in blocks. Different from the protocol in [51], we let Bob send back the hash values of the intersection set. If Bob is impersonated by an adversary, our system gives him chance to maliciously change the returned values such that Alice will obtain a intersection set different from his own one. We remark that the adversary cannot arbitrarily manipulate the set intersection in the sense that he can only remove elements or add new elements that he does not know (chosen from Alice's published hash values). While this breaks the fairness in private set intersection protocols, we argue that it does not harm our system. It is because the intersection set itself is actually of no interest to the adversary, whose goal is to get the final session key and the follow up communication contents. If the adversary modifies the intersection set, he will definitely obtain a session key different from Alice's. Alice will then abort in the key confirmation phase. Therefore, an adversary has no incentive to do such a modification.

3.6 Evaluations

In this section, we demonstrate the performance of our system. We made a proof-of-concept implementation of our NFA system on Motorola Droid smartphones, which have a 550MHz ARM A8 processor, 256MB memory, a 16GB SD card, and Android 2.2 OS.

This bland specification makes Droid a good representative of the low-end smart devices in today's market.

We first tested whether our feature reconciliation can finish successfully with a high probability. In other words, out of *m* randomly selected pairs, is it of high probability that two devices can find at least 4 valid pairs? To answer this question, we did 100 experiments on two Droid smartphones and another 100 experiments on a Droid smartphone and a HP TouchPad tablet. Each experiment collected at least 10 peak points. We tested the success rate under different *m* values. Fig. 3.8 shows the proportion of the experiments in which our system can finish successfully. From the figure, we can see that even if *m* is only 5, our system can succeed with probability close to 80%. When *m* is increased to 7, data collected from two Droid phones showed a success rate of 97.8% while the rate from two different devices was 90% . When *m* is 10, the experiments on two different devices also showed a very high success probability, 97%.

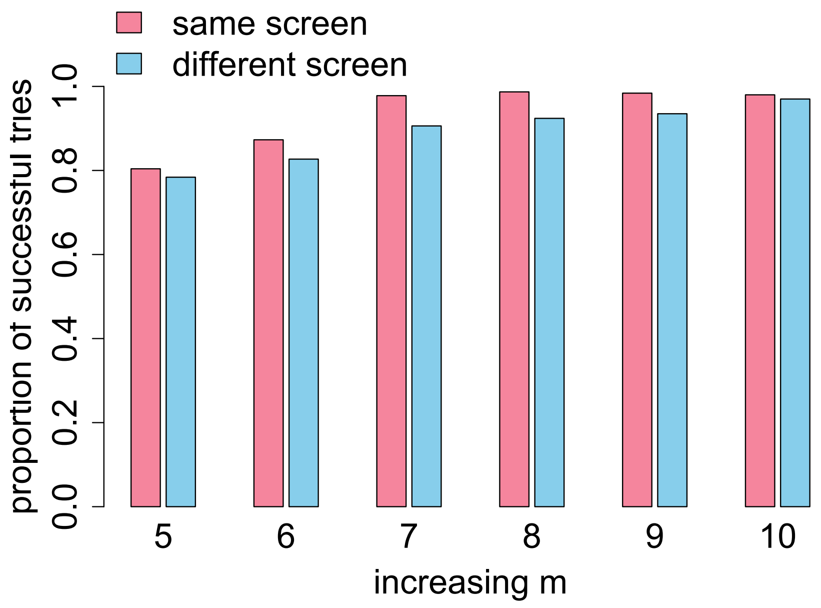


Figure 3.8: Success Rate of Feature Reconciliation

In the feature reconciliation phase, we use a hash-into-group hashing function H_1, which is constructed from a cryptographic hash function as introduced in Sec. 3.4. The construction is a probabilistic algorithm. We tested the success rate of this hash function. For test purpose, we hashed 1000 random numbers, from $(0, 200]$, into four different order groups: 128, 160, 512, and 1024 bit groups. The first two group orders are popular selection for the elliptic curve groups (ECC) and the last two are for the quadratic residue groups over the Galois field modulo a safe prime, which is called DL groups in this section. 160-bit and 512-bit output were obtained by using SHA-1 and SHA-512. In order to obtain a 128-bit output, we used SHA-1 and took the 128 least significant bits as output. To obtain 1024-bit output, we divided the binary string of the input number into two halves, hashed each half into a 512-bit string, and concatenated the both together. For each number, we recorded how many attempts needed to hash the number into the desired group. Each experiment tried 10 re-hashes before it reported "`fail`". Fig. 3.9 shows the results. We can see that, in

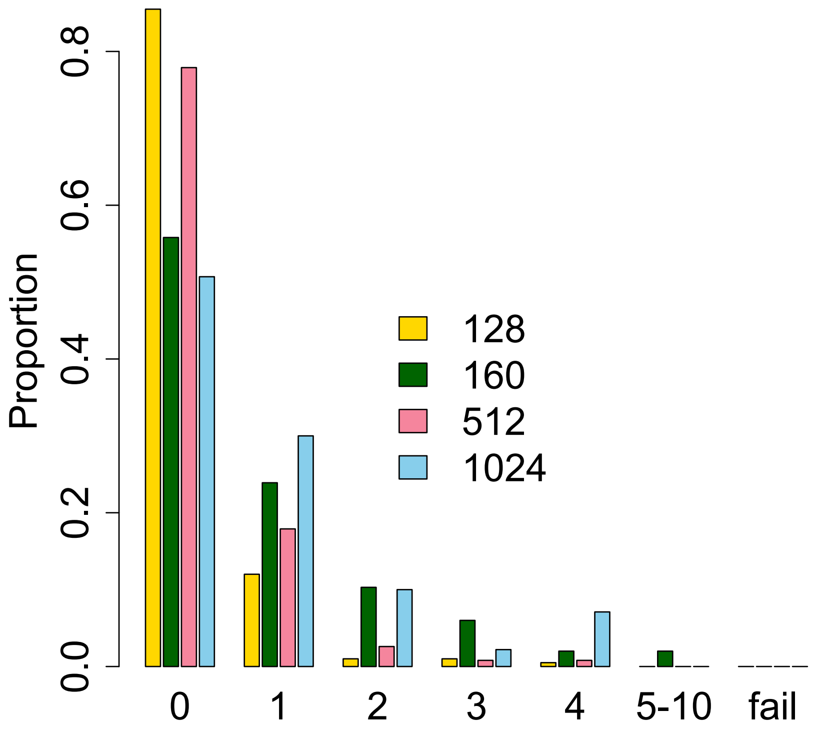
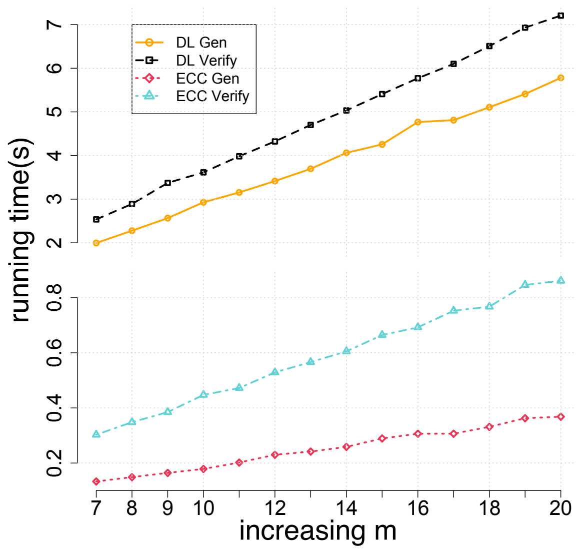


Figure 3.9: Hashing Into Group

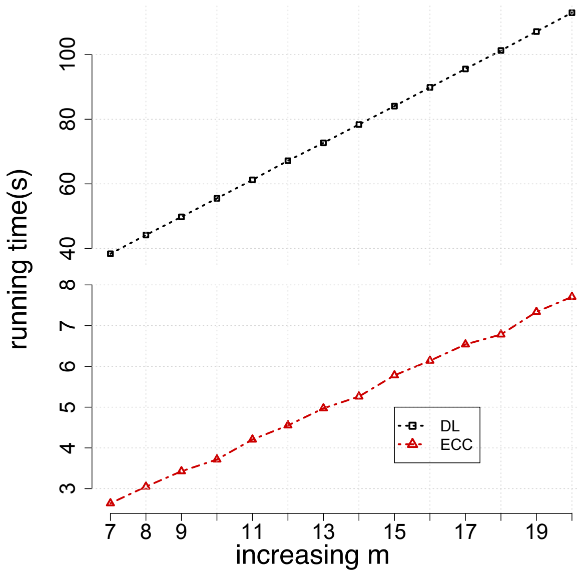
our experiments, to hash a number into the 128-bit 512-bit, and 1024-bit groups, we needed to try no more than 4 times. For the 160-bit group, we needed to re-hash a small portion of numbers for more than 5 times. We note that, in our experiments, all 1000 random numbers were successfully hashed into the groups in 10 attempts.

For a smartphone protocol, the execution time is critical in the sense that smartphones usually have limited computational and storage resources. As pointed out previously, all our simulation were carried out on a low-end Motorola Droid smartphone. We tested the running time of the zero-knowledge proof generating and verifying, the feature reconciliation phase, and the key generation phase. We did not test the key confirmation phase since it does not take up much proportion in the system execution time and actually always costs the same time when the security parameter is determined. Our system was implemented in two types of groups: a 1024-bit quadratic residue subgroup of a Galois field modulo a safe prime p and a 160-bit elliptic curve group. According to NIST's guidance [78], the two groups both have 80-bit security level. The results are shown in Fig 3.10a, 3.10b, and 3.10c. We remark that the time of the feature reconciliation phase contains the time of generating and verifying zero knowledge proof. We note that the y-axis in the figures are not continuous and we skipped in the middle because there was a big gap between the execution time of the two implementations. To show the details of the time changing in the plots, we stretched the y-axis and cut off the middle blank area. We executed the two implementations under different m values introduced in step 1.

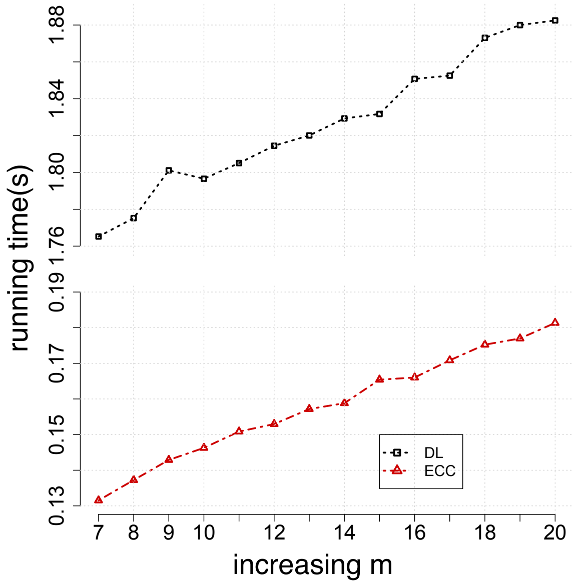
All the execution time increased linearly with the increase of m, since we have to process larger sets in generating the proof and performing private set intersection. A larger set also leads to higher probability of generating a long password w, which causes more execution time in generating the session key. Another obvious observation is that the ECC implementation was overwhelmingly faster than the DL implementation in zero knowledge



(a) Time of zero knowledge proof



(b) Time of feature reconciliation



(c) Time of key generation

Figure 3.10: Evaluation Results

proof, feature reconciliation, and key generation. For an example, if we set $m = 18$, the DL implementation took almost 100 seconds to privately find a set intersection, while the ECC implementation needed less than 7 seconds. The same performance difference was also observed when we perform zero knowledge proof and key generation. The difference is due to the different group size used by the two implementations: the DL implementation used a 1024-bit group and the ECC implementation used a 160-bit group. The reason for the different order choice is that the security of a DL group is easier to be cracked when its order is the same as an ECC group. For a larger group size, a basic group operation, such as multiplication and exponentiation, costs more time. Therefore, it is not surprising to see such a big gap between the two execution time values. We thus recommend to use ECC group as the real implementation of our NFA system. In practice, $m = 10$ is sufficiently large to select 4 pairs of valid peak points. In this case, the ECC implementation took 3.71 and 0.14 seconds to finish first phase and the second phase. Totally, the system can terminate in less than 4 seconds for an authentication.

3.7 Conclusions

In this chapter, we have proposed a new concept, named near field authentication, to help smart devices authenticate each other in a near field. We have designed a near field authentication system which uses a novel and natural human motion, zigzag finger movement, to enforce the spatial closeness. We have shown that the finger movement have high variations and stable similarities. In order to remove the differences between sensed feature data and generate high-entropy session key, we have proposed to use private set intersection and encrypted key exchange in our system. Finally, we have simulated our system on a real Android smartphone with a bland hardware specification and demonstrated its efficiency. This work has been published in [63, 64].

CHAPTER 4

Secure Acoustic Near Field Authentication

Many smartphone assisted security applications grant user services or privileges based on the physical proximity between a smartphone and a terminal, e.g., a computer. Hence proximity assertion is an important security mechanism. In this paper, we present a novel *near field assertion* system that generates valid assertions only when the two devices are within a few centimeters.

Our work is motivated by smartphone based two factor authentication (SBTFA), which verifies both a user's username/password and a pre-loaded secret in the user's smartphone. During authentication, the smartphone's proximity indicates that the operator is the user himself. However, this indication is suspicious in the presence of *relay attacks* [55]. This type of attack utilizes two communicating devices, a *leech* and a *ghost*. The leech is placed physically close to the smartphone and the ghost close to the terminal. The attacker simply relays challenge and response messages between the terminal and the smartphone via ghost-leech intercommunications and thus creates the illusion of physical proximity between the two honest devices even if they are very far apart. Relay attacks void the security protection provided by the smartphone. For example, a user is waiting at a coffee shop counter with his smartphone in his pocket and leaves his laptop unattended on a table. Even though the operating system on the laptop is protected by SBTFA, an attacker who can perform relay attacks only needs the user's password to crack into the system. Relay attacks raise a greater threat against applications that use smartphones as the only authentication factor [13, 22].

Relay attacks make SBTFA no securer than password only authentication. Even worse, people may reduce their security vigilance due to the installation of a "securer" protection. Many SBTFA system designers have realized the threat of relay attacks and confined the system communications in a short range wireless network. However, recent works have demonstrated positive feasibility studies for performing relay attacks in the real world over short range wireless networks, including Bluetooth [61], RFID [36], and NFC [37]. To prevent relay attacks and enforce proximity communications, researchers have developed two main techniques. One is *distance bounding protocol* [19] that cryptographically bounds the inter-device distance by measuring the response time. The other is *contextual co-presence* approach [44, 69, 89] that enforces the proximity by comparing the ambient information (e.g., GPS, temperature, etc.) sensed by the two co-present devices. However, the distance bounding protocol is difficult to adopt on main-stream devices. The contextual co-presence approach can only confine the two devices in a large area, such as a coffee shop. A relay attack is still possible if the attacker also stays in the same area.

Therefore, there is a great demand for a system that can restrict devices in a small area and fully resist relay attacks. However, relay attacks are due to an essential flaw of existing communication system — there is an unpredictable wait between a message and its response. This makes relay attacks very challenging to defend against.

In this paper, we present a surprisingly simple solution to this problem. The main idea is illustrated in Figure 4.1: if two people communicate by whispering, it is hard for a third person at a distance to hear the conversation.

Figure 4.1: Whisper Restricts Communication Distance

While the main idea is easy to understand, building *a working system* that can prevent relay attacks is challenging. Our main contribution is a design and implementation of such a system, named *Dolphin*. Dolphin generates near field assertions via acoustic communications, manipulates the sound power to restrict the communication distance, and uses full-duplex communication [50] to prevent relay attacks. In particular, Dolphin has the following properties.

- It asserts whether two communicating devices are in the near field (a few centimeters) of each other. We use the term "near field" instead of "proximity" to emphasize that the asserted distance can be as small as a few centimeters. A device can prove that it is in the near field by presenting the assertion to the other party.
- It can prevent relay attacks. Dolphin has an adjustable time window limiting an attacker's relay time. A prudent implementation leaves an attacker no time to relay messages.
- It requires no extra equipments and can be easily deployed on off-the-shelf devices. Dolphin is a fully automated system and needs no human interactions.
- A valid near field assertion generated by Dolphin is a binary sequence and *confidential* to the two devices that execute Dolphin. This property is important for applications that generate a cryptographic session key based on the devices' proximity relations, such as one-time file sharing between two proximate smartphones [3].

- It is light-weight and battery friendly for smartphones.

Although Dolphin is designed to provide relay-resistant near field assertions, it also has an interesting “by-product” — valid assertions are confidential to the two communicating devices. This property helps improve the security of applications that base their system security solely on the physical proximity relations.

The rest of this chapter is organized as follows. In Section 4.1, we review prior works on proximity assertions. We present our design goals and thereat model in Section 4.2. In Section 4.3, we illustrate the key ideas used in Dolphin and present an overview of the system. In Section 4.4, we present the details of the entire system. We present evaluations of Dolphin in Section 4.5, and conclude the paper in Section 4.6.

4.1 Related Works

The problem of restricting the distance between two communicating devices has been studied in pairing systems [29] and relay resistant systems [98].

One promising approach is to transmit data over an out-of-band (OOB) channel and use the channel’s physical properties to restrict the distance. Stajano and Anderson [93] proposed to use a physical cable. Balfanz *et al.* [10] used infrared light to restrict distance. Near field communication (NFC) technique can be employed to construct a near field assertion system. However, these approaches require an additional equipment (e.g., a cable or an NFC chip), which may not be available when an assertion is needed. *It is preferable to have a system that requires no extra equipments and can be executed on most off-the-shelf devices.*

A few works took advantage of visual OOB channel [73, 87]. McCune *et al.* proposed SiB [73] that displays a barcode on one screen and “reads” it using another device.

Saxena *et al.* [87] proposed to let the devices encode and transmit data via blinking LED. Other works explored user's motions [1, 25, 72, 96] and used the sensed force during motion to restrict the communication distance. BUMP [1] uses the bump force during a bump between two devices. Shaking two devices together has been widely used in many works [25, 72, 96]. Some works let a user manually compare the information displayed on the two device screens [59, 81, 100]. However, most of these approaches are not automated and require extensive user involvement. *Such approaches become unsuitable when assertions are frequently needed.*

Recently, Nandakumar *et al.* [77] achieved confidential data transmission over acoustic channels in the near field. They used low signal power and signal interference to protect the data. However, their system is not designed for distance restriction and is vulnerable to relay attacks.

Distance bounding protocols [19] cryptographically restrict the distance of two communicating devices: a verifier sends a series of rapid-fire challenges and requires a prover to respond immediately. The round-trip time of an interaction is measured and used to estimate the distance. The distance bounding protocol is provably secure but requires a no-latency communication channel and complicated implementations. The inconveniences impede this approach's industry adoption. *Therefore, a desired approach should be easily deployed and adopted.*

Alternatively, contextual co-presence is a widely-adopted approach which lets two devices sense and compare the ambient information to make a proximity assertion. Various systems have been proposed using different information, including GPS raw data [69], WiFi signal strength and access point address [58], ambient acoustic finger print [44, 89], cell broadcast information [38], and user's sitting posture [13]. Usage of a combination of multiple environment information has been studied in [90, 98]. Although such approaches restrict two devices in the same area, the restricted distance cannot be at the accuracy of a few centimeters. In addition, keeping sensors awake for a long time drains smartphone battery. *It is necessary to design a system that restricts distance in the near field and is battery friendly.*

4.2 Design Goals and Threat Model

4.2.1 Design Goals

The main objective of Dolphin is to generate valid assertions only when the two devices are in the near field. We do not include any identity authentication in the system and leave the usage of generated assertions to application developers. The design goals of a desired system are outlined below.

- **Functionality**: the system asserts whether two devices are in the near field and outputs an assertion to each device. The two assertions are identical if the devices are in the near field, but different otherwise.

- **Credibility**: when the resulting assertions are identical, the two devices are in the near field. Attackers cannot force two devices to generate the same assertion if they are not in the near field. The attacks include but are not limited to relay attacks. For example, if a system is based on sensing ambient sound, a sophisticated attacker can break credibility by playing the same music loudly and close to the two devices.

- **Confidentiality**: valid assertions are not known to any other party other than the two communicating ones. This is useful in security applications that create a session key based on the devices' proximity relations. The resulting, confidential assertions can be used as a shared secret and to derive a session key.

- **Usability**: 1) a fully automated or zero-interaction system is preferable; 2) the system should need no extra equipments and be compatible with most off-the-shelf devices; 3) the system should be easy to develop and adopt.
- **Sustainability**: the system should not drain smartphone battery fast.

4.2.2 Threat Model

Dolphin consists of a terminal, a smartphone, and a local network connection between them. A terminal is not necessarily a big-size machine and could be a smartphone or a tablet. The two devices are both honest. The connection is assumed to be insecure and controlled by the attackers. We do not consider jamming or DDoS attackers, whose purpose is to destroy the communications and assertion generation.

Dolphin may be used in two scenarios, assertion only and authentication, and can cope with different attackers.

Assertion Only is a scenario where the two devices only want to make a near field assertion. The parties care about the functionality and credibility rather than the confidentiality of the assertions. SBTFA [32] fits this scenario. In this scenario, we assume the terminal and the smartphone have their own signing-verification key pairs and have exchanged the verification keys. Also, the two devices have each other's public key for encryption purpose. Dolphin allows attackers to eavesdrop the entire local network and to manipulate any message. Attackers may carry out relay attacks between two devices that are not in the near field.

Authentication is a scenario where two devices want to derive a session key from valid assertions. This scenario requires that the resulting assertions are confidential. Pairing [29] and secure near field file sharing [3] are two examples that fit this scenario. In this scenario, Dolphin does not require the two devices to exchange any secret in advance. An attacker controls the local network but stays outside the near field during the system execution. We assume that the attacker cannot perform man-in-the-middle (MITM) attacks. Relay attacks are different from MITM attacks in that an MITM attacker actively modifies the relayed messages but a relay attacker does not. We argue that this assumption is reasonable, because an MITM attack cannot be prevented without any trust base (e.g., a pre-exchanged secret). In addition, this scenario usually happens with smartphone owners' attendance. Performing an MITM attack via acoustic communications in the near field would arouse the owners' suspicions. The owners can either exam and clear their environment before executing the system or manually compare the final assertions afterward, in a human friendly way [34].

4.3 Key Ideas of DOLPHIN

Dolphin explores sound as an OOB channel, because the power of sound fades very fast with the increase of transmission distance. A real life example is that when someone is whispering, we can hear it clearly if we stand very close. If we step away from the speaker, the voice becomes weaker and weaker and is finally buried in the ambient noise. Similarly, the idea of Dolphin is to transmit acoustic signals with power equal to the ambient noise power plus a small threshold. Using the threshold, Dolphin ensures that the signal cannot be decoded beyond a desired distance. Another nice property of sound is that it requires no special equipment, making Dolphin work on most off-the-shelf devices.

Relay resistance is a necessary but challenging part of Dolphin. Relay attacks are due to the fact that a communicating party does not know how long it needs to wait for the response message. Why is the waiting time unknown? The answer is that the communications are half-duplex! After one party sends out its message, the other one does not respond until the message has been delivered by the network. The delivery could be influenced by unpredictable network latency and attackers can take this chance to relay messages, as illustrated in Figure 4.2a. To fully resist relay attacks, we propose to use *full-duplex* communications. The two devices simultaneously transmit outgoing messages and receive incoming messages. Dolphin has a time window of an adjustable size. Starting at its message transmission, an honest party only accepts a message arriving within the time window, as illustrated in Figure 4.2b. Therefore, a small time window makes relay attacks impossible.

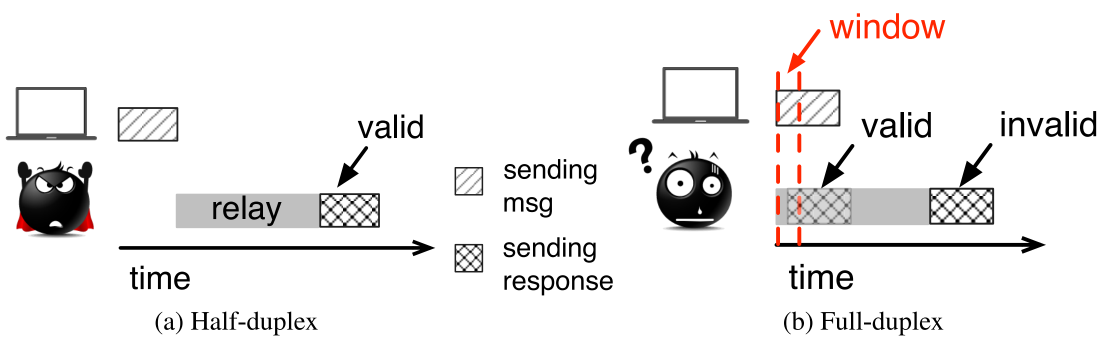


Figure 4.2: Half-duplex vs. Full-duplex

While signal interference is what communication system designers usually want to avoid, Dolphin *explores signal interference* to protect the confidentiality of transmitted messages and the resulting assertions. Dolphin simultaneously transmits two message signals in a way that the interference between them makes it extremely difficult for an attacker to separate and decode them. While it is difficult for an attacker to decode the two (combined) messages, Dolphin uses self-interference cancellation to make the decoding possible for the two parties involved.

Overview of Dolphin: A user places his smartphone close to a terminal and starts Dolphin on both. Once started, Dolphin keeps running and generates assertions on demand. If a device wants to make an assertion, it sends the other device a request message to trigger the following process. The two devices each choose a b-bit random binary sequence, where b is a positive integer. A larger value for b incurs more communication overhead but provides stronger security. In our implementation, we set b to 16. Both devices measure the ambient noise, determine the transmission power, modulate the sequence to an acoustic signal, and transmit their signals simultaneously. After the transmission completes, each device cancels self signal interference, demodulates and obtains the binary sequence of the other. Each device obtains a $2b$-bit near field assertion by appending the smartphone's sequence to the terminal's. Except the acoustic signal, all messages are transmitted over the local network.

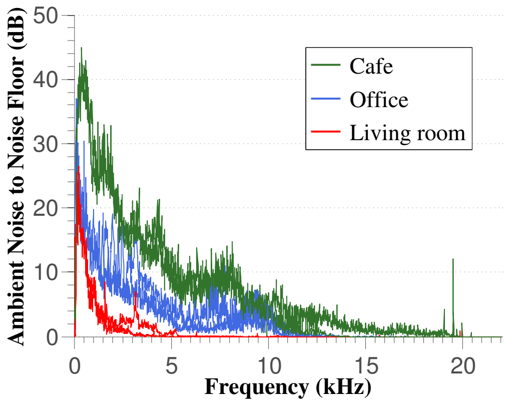
4.4 Design Details of DOLPHIN

For demonstration purpose, we implemented a prototype of Dolphin on an iPhone 5s and a 13-inch mid 2009 Macbook Pro over WiFi connections. We chose Apple platform for two reasons: 1) it provides similar APIs for iOS and Mac OSX, accelerating the development; 2) iOS and Mac OSX have reputations in audio programming — providing low latency, flexible APIs, and powerful native libraries. We emphasize that our framework and design methodologies can be applied to any platform and need no special hardware support.

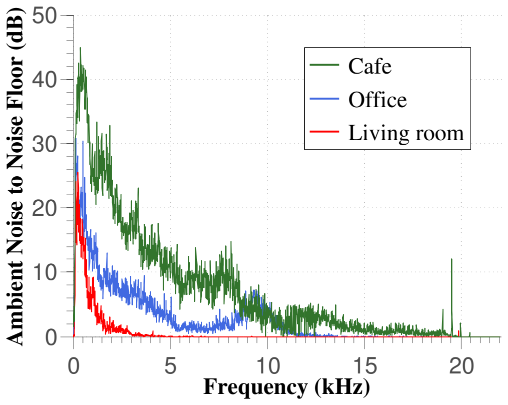
We first explain some terms that will be used in the paper. A **signal** is a sound transmitting a message. Digitally, a signal is stored in a sequence of discrete **samples**, that denote the signal amplitudes at a fixed sampling rate. A **symbol** is a group of consecutive samples carrying one message bit. A signal contains a sequence of symbols. A **tone** is a sine wave signal of small length.

4.4.1 *The Acoustic Channel*

We start our designs by studying Dolphin's OOB channel — the acoustic channel, which is significantly interfered by the ambient noise. To characterize the interference, we measured the ambient noise power at three typical places where Dolphin may work — an apartment living room, a six-people office with A/C working, and a noisy coffee shop. First, we measured the noise floors of the Macbook and the iPhone, respectively, in an isolated silent environment. Next, we measured the ambient noise power spectrum at different places. These results are shown in Figure 4.3.



(a) Mac



(b) iPhone

Figure 4.3: Spectrum of Ambient Noise

Most digital electronics today can support an acoustic sampling rate of up to 44.1kHz, indicating that the largest operating frequency is 22.05kHz, i.e., the Nyquist frequency. As seen from the figure, the ambient noise at the office and the coffee shop are notably high — around 40 dB above noise floor for both Macbook and iPhone at low frequencies. The noise power in the living room reduces to a small value above 5kHz, while the office and the coffee shop still have a plenty of noise power until 10kHz and 15kHz, respectively. Human voice is usually below 5kHz. Noise around 10kHz is due to the A/C noise in the office and the music in the coffee shop. A key observation is that the noise powers at all three locations decrease quickly to a small value above 15kHz.

Dolphin uses four speaker-microphone acoustic channels — Mac to Mac, iPhone to Mac, Mac to iPhone, and iPhone to iPhone. A device's microphone receives acoustic signals from the other device's and its own speakers. Figure 4.4 depicts the frequency response for the four channels. The frequency responses were measured by sending a

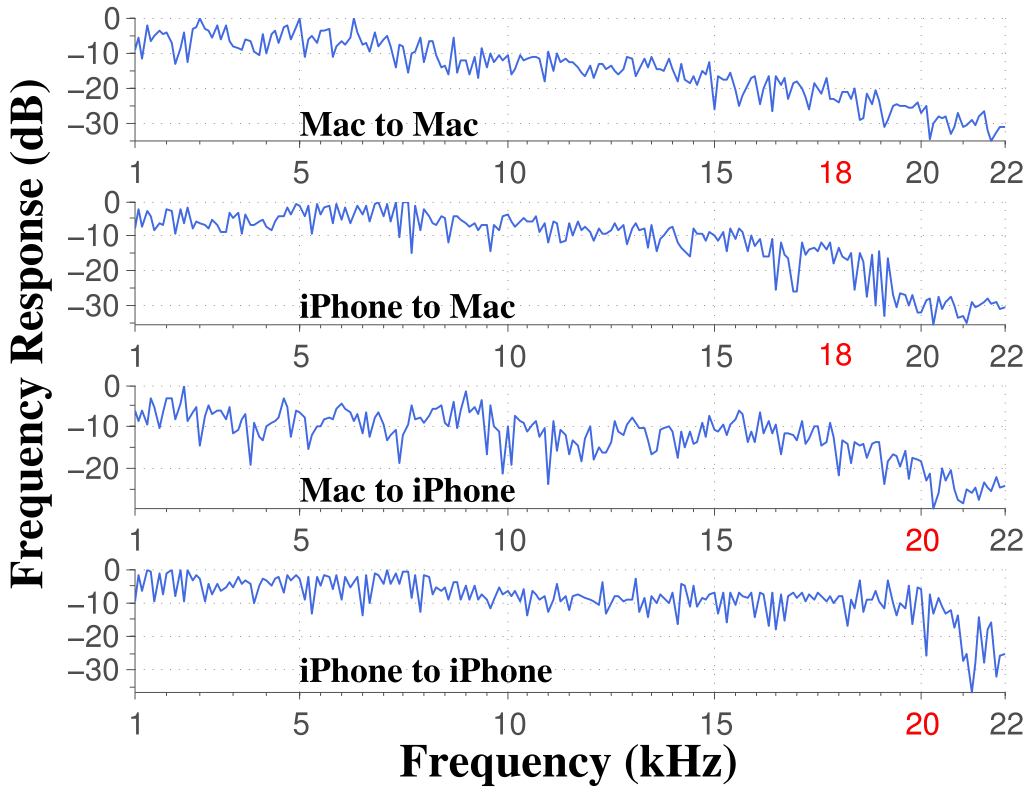


Figure 4.4: Frequency Responses for "Speaker to Microphone" Pairs

series of tones from a speaker and recording the received signal power at a microphone. The tones were of frequencies between 1kHz and 22kHz at intervals of 100Hz and at the same power level. Each plot of Figure 4.4 uses the maximum power of any received tone as the reference. We point out that during the course of the measurement, the iPhone and the Macbook were placed in the way as described in Section 4.4.6. From Figure 4.4, we observe that a remarkable power attenuation occurs at around 18kHz for Macbook microphone and 20kHz for iPhone microphone. More information will be lost if signals are transmitted at frequencies beyond the attenuation point.

We also notice that human hearing sensitivity dramatically drops at frequencies higher than 10kHz. This is reflected by the absolute threshold of hearing (ATH) graph, Figure 4.5 (taken from [109]), where the two dotted lines represent age 40 group and age 60 group, respectively. The ATH graph depicts a curve such that given a frequency, we can hear a tone only if its sound pressure level (SPL)[1] is above the curve. The ATH graph implies that an acoustic system should use high frequency signals to be less annoying.

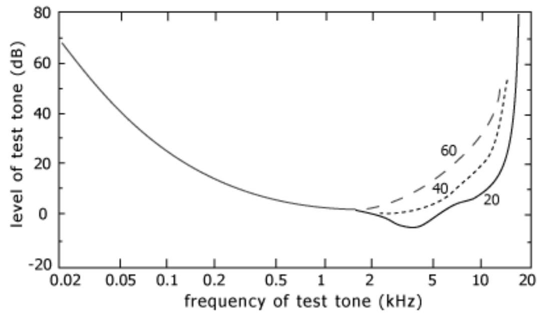


Figure 4.5: ATH Graph

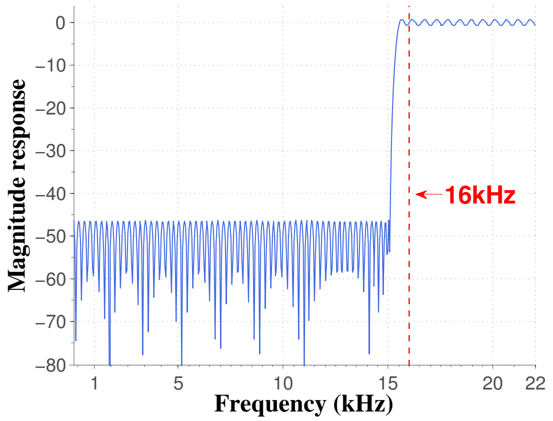


Figure 4.6: Filter Response

Given these observations, we decide to use 16kHz acoustic channel in Dolphin. Fixing the channel frequency, Dolphin uses a high-pass Finite Impulse Response (FIR) filter, depicted in Figure 4.6, to reduce noise power. Figure 4.3 shows that ambient noise is as high as 45 dB over the noise floor. The filter suppresses the frequencies below 15kHz by

[1]Sound pressure is the pressure caused by a sound wave, describing the sound loudness [84]. SPL is its logarithmic measure (in decibels) relative to a reference pressure, 20μPa.

an average of 55dB while preserving the frequencies above 15.5kHz.

Despite noise interference, a symbol of a signal is also interfered by its predecessor and successor, i.e., Inter-Symbol-Interference (ISI). We transmitted a 256-sample sine wave tone (at 16kHz) in each of the four channels and plotted the received signals in Figure 4.7. We observe a notable *rise* phase in the two channels where Macbook is the sender.

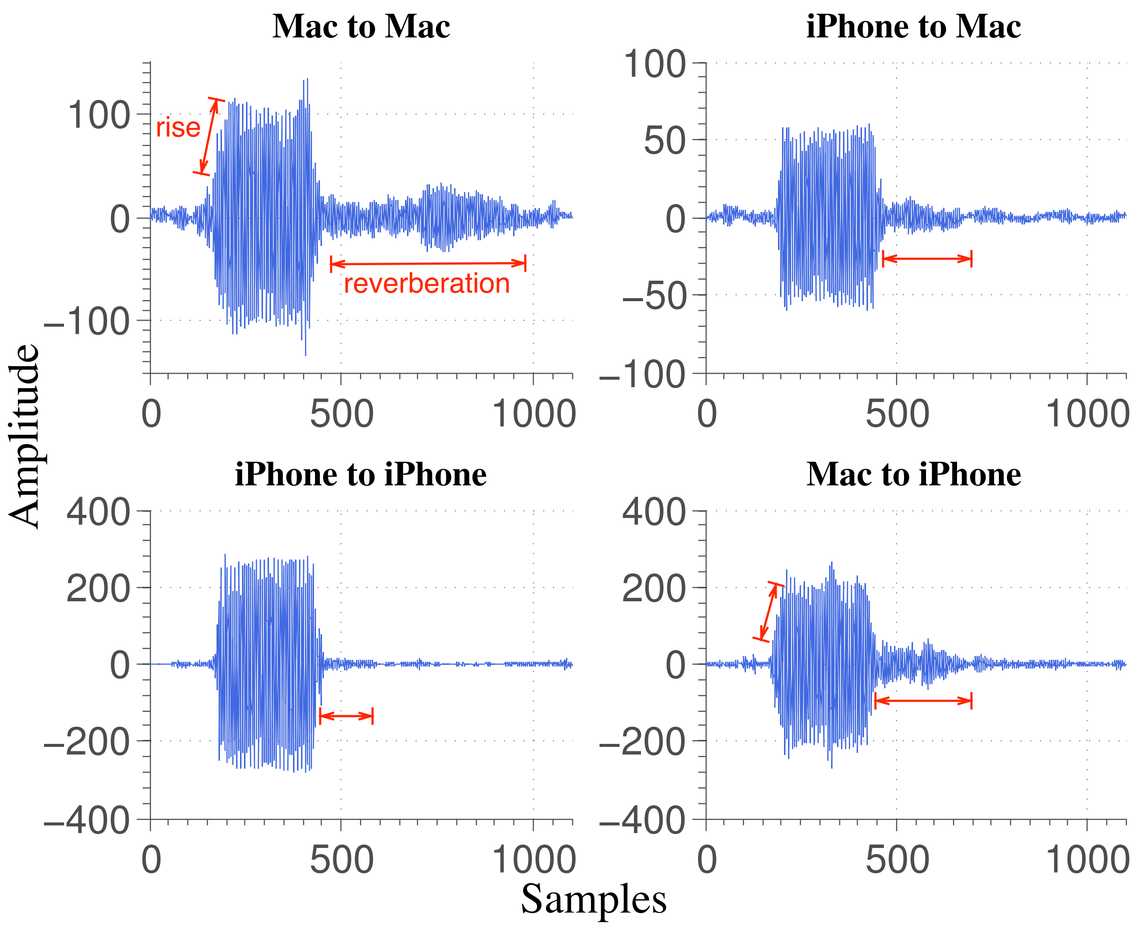


Figure 4.7: Rise and Reverberation Time in the Four Channels

In audio programming, a speaker sounds according to what a program sets in the audio buffer. Ideally, the diaphragm of a speaker immediately switches to the desired amplitude in the buffer. However, speakers are analog devices and need time to "warm up", which is reflected as a rise time. An undistinguished speaker makes more notable rise time than a nice one does. In the meantime, we observe a *reverberation* time in all four channels, with the Mac-to-Mac channel having the longest one. We transmitted tones of different lengths to see the change of the reverberation time. The results are listed in Table 4.1. The ISI

Table 4.1: Sine Wave and Reverberation Time

Sine Wave	64	128	256	512
Reverberation	122	210	348	617

is caused by the rise and reverberation interferences. *To avoid the interferences, Dolphin selects 700 as the symbol length and the first 256 samples are reserved for the sine wave tone.*

4.4.2 Modulation

In Dolphin, modulation is to transform a binary bit to a high-frequency sine wave tone that can be transmitted over the acoustic channel. There are many modulation approaches, including amplitude-shift keying (ASK), phase-shift keying (PSK), and frequency-shift keying (FSK). Dolphin uses a simple on-off keying (OOK), in which the presence or absence of a tone denotes bit 1 or 0. OOK is vulnerable to noise interference. Figure 4.8 shows the theoretical bit error rate (BER) versus signal-to-noise ratio (SNR) for four modulations: OOK, quadratic PSK (QPSK), binary PSK (BPSK), and binary FSK (BFSK) [12]. The channel is an additive white Gaussian noise channel. With the decrease of the SNR (to the left of the figure), OOK quickly gets to a high bit error rate. Particularly, the BER of OOK gets to 0.32 when SNR is 5dB. This vulnerability harms a communication system but helps Dolphin to restrict distance. Increasing acoustic signal's travel distance decreases the SNR and dramatically hampers the demodulation. In the meantime, OOK is easy to demodulate and is also used by NFC standard [85].

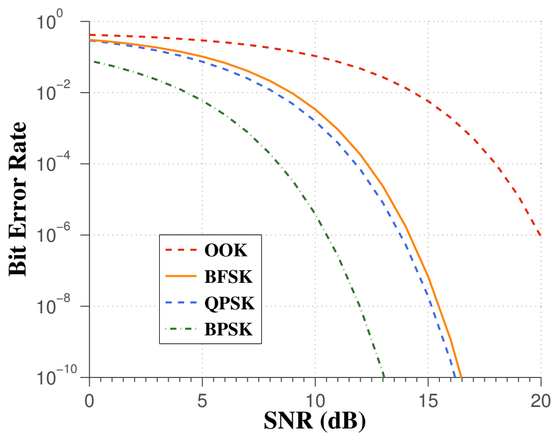


Figure 4.8: BER vs. SNR.

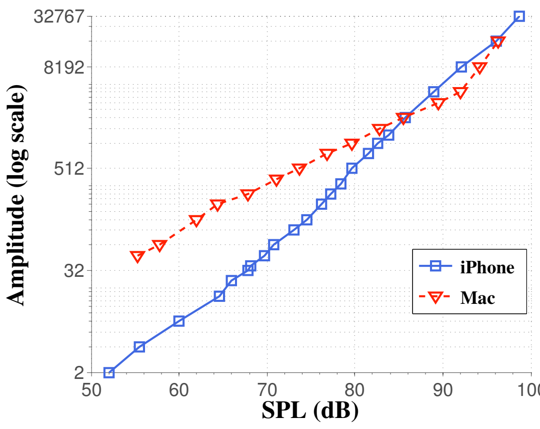


Figure 4.9: Measured SPLs and Their Amps

Dolphin transforms a binary sequence to a series of *symbols*. A symbol carries a bit "1" if it has a 16kHz sine wave tone in the first 256 samples and a bit "0" otherwise.

4.4.3 *Calibration and Safe Distance*

Dolphin restricts distance by manipulating the signal amplitude and thus changing the emitted SPL. We need to know how much amplitude can produce a given SPL. We let Macbook and iPhone play a 16kHz sine wave signal at various amplitudes under full volume and measured the SPLs at a reference distance d_0. We set $d_0 = 1$cm in our implementation. The results are shown in Figure 4.9 where the markers are the measured data points. We can see that the logarithm of the amplitudes are linear in the SPLs, which conforms to the definition of SPL [84]:

$$L_p = 20 \cdot \log_{10}(\frac{P_{rms}}{P_{ref}}) = 20 \cdot \log_{10}(\frac{A/\sqrt{2}}{P_{ref}}).$$

Here, L_p is the SPL at point p. P_{ref} is 20 μPa in air. P_{rms} is the root-mean-square of the signal's amplitudes and equals to $A/\sqrt{2}$ because the signal is a sine wave.

The *inverse distance law* [84] states that sound pressure falls inversely proportional to the distance away from a sound source. The sound pressure levels L_{p1} and L_{p2} at distance d_1 and d_2 conform to the following equation.

$$L_{p2} - L_{p1} = -10 \cdot \log_{10}(\frac{d_2}{d_1})^2 = -20 \cdot \log_{10} \frac{d_2}{d_1}. \tag{4.1}$$

Dolphin achieves distance restriction by preventing an honest party's signals being demodulated beyond a *safe distance d*. Let L_a be the SPL of ambient noise. For OOK modulation, $SNR = L_d - L_a \leq 5$dB is sufficiently low to cause a high bit error rate. According to (4.1), we can calculate the needed SPL L_0 at the reference distance

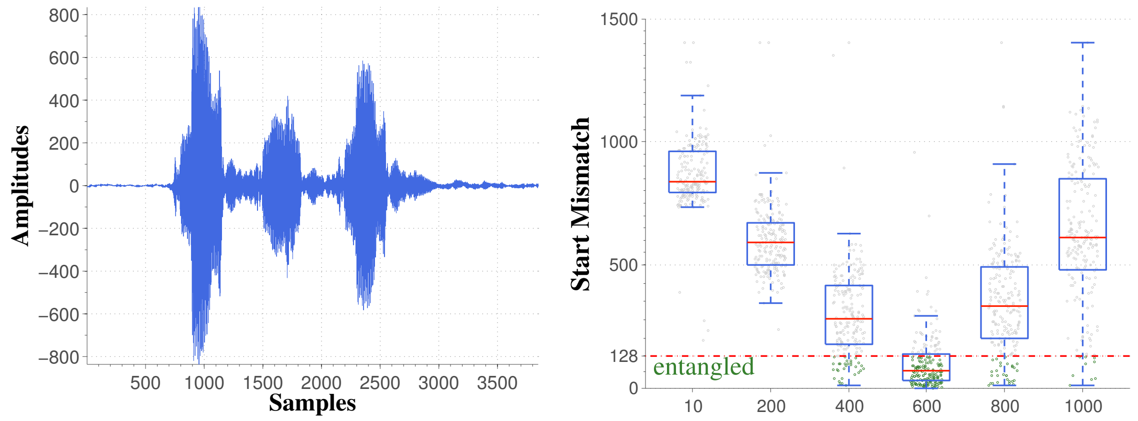
$$L_0 \leq L_a + 20 \cdot \log_{10} \frac{d}{d_0} + 5.$$

In our implementation, we choose $L_d - L_a = 0$ and $d = 15$cm, which yields $L_0 = L_a + 23.52$. Using Figure 4.9 and interpolation, we can calculate the needed amplitude for L_0.

4.4.4 Synchronization

For confidentiality purpose, Dolphin requires the two devices to transmit signals simultaneously to interfere the two signals. For example, Figure 4.10a shows a received signal, when iPhone sends "101" and Macbook sends "111". Although the first time slot and the third time slot both have symbol interference, they have different combined signal shapes. Without knowledge of any original message, it is very difficult to separate and recover the two signals.

Synchronizing two devices' actions is challenging, because network status and a device's internal status are time-varying. We attempted to use an Internet time server, `time.apple.com`, but the accuracy turned out to be not acceptable. There were nearly 0.05-second difference, resulting in a start mismatch of 2205 samples. To achieve synchronization, Dolphin estimates the message delivery time and balances start time on the



(a) Signal interferences (b) Mismatch vs. buffer sizes

Figure 4.10: Signal Interferences and Synchronization

two devices. We remark that considering its easy adoption, we are not giving up the time server based approach but looking for more precise solutions, e.g., GPS Time server or the approach in [97]. We leave the improvement to our future work.

A start mismatch is the time difference, measured in number of samples, between the start points of two signals. The start mismatch of two signals need to be less than one tone length, i.e., 256 samples in Dolphin, to fully interfere them. Furthermore, we say that two signals are *entangled* if their start mismatch is less than half a tone length.

Start mismatch is caused by system latency and audio buffer size. System latency includes network latency and internal latency caused by a device's internal procedures, e.g., job scheduling. Dolphin measures and estimates system latency as follows. First, Dolphin enhances its own instance's priority to the highest possible level, to reduce internal latency. In iOS and Mac OSX, this is achieved by using `p_thread` library and `NSThread` API. The smartphone sends a probe message to the terminal, which sends an acknowledgement message immediately after the probe arrives. Upon receiving the acknowledgement, the smartphone calculates the message round trip time rtt. Repeating the process a few times, the smartphone obtains an estimated message delivery time $\overline{rtt}/2$, where $\overline{rtt}$ is the average round trip time. The estimation is performed each time before signal transmission. Once the smartphone sends a "start transmission" message, it holds for $\overline{rtt}/2$ to start transmission and the terminal starts transmission upon receiving the message.

In audio programming, the speaker does not sound until the entire audio buffer is fulfilled. Hence, audio buffer size also affects the latency. To test the impact, we used system default buffer size for iPhone and measured start mismatches by changing Macbook's buffer sizes to 10, 200, 400, 600, 800, and 1000 samples. A measurement is done by transmitting two messages, "1000" and "0001" and counting samples between the starts of the two tones. The start mismatch is the absolute difference between the counting result and the expect number, 2100 ($3\times$ symbol length). For each buffer size, we used 4 different WiFi networks and performed 50 measurements in each network. The networks' average ping times are 7.639ms, 3.777ms, 4.770ms, and 25.546ms. The standard deviations are 3.171, 1.275, 5.451, and 10.332. Figure 4.10b shows the start mismatches in box plots. In a box plot, the box encloses the points from the 25 percentile to the 75 percentile, and the red line in the box represents the median. The red dashed line in the figure denotes the entanglement boundary. A point below the line, colored dark green, indicates that the two measured signals are entangled. The remaining points are colored light grey. Obviously, a 600-sample long buffer performs best since most of its points are below the entanglement boundary. Specifically, it has 149 mismatches less than 128, and 187 ones less than 256. Increasing the buffer size, we can see that the 6 medians show a "V" shape in Figure 4.10b. This is because a small buffer size makes Macbook respond faster than iPhone while a large one makes Macbook respond slower. *By setting buffer size to 600, Dolphin can synchronize most transmissions and thus protect confidentiality of the final assertions.* If a synchronization fails, Dolphin just re-executes the protocol until a synchronized transmission is achieved. Synchronization failure causes decoding failure, which is detected by CRC code in Dolphin, see Section 4.4.6.

4.4.5 Self-Interference Cancellation

The biggest challenge for full-duplex communication is to cancel self signal interference and extract the other party's message, which is called self-interference cancellation (SIC) [50]. Radio wireless communication researchers achieved SIC by accurately placing an additional antenna [50], recording the oppositely phased version of self signal, and using it to cancel self signal on flight. This design is for distant transmission where the arriving signal is much weaker than the self signal. However, Dolphin works in the near field and both signals are equally strong. In addition, an additional antenna is not available on most devices.

Dolphin's SIC approach is to subtract a device's self signal from its received one so as to obtain the arriving signal. Different from the aforementioned SIC, our approach is an off-line process and has a smaller throughput. Note that maximizing throughput is not the aim of Dolphin. The goal of our SIC approach is to reconstruct the self signal, which has been distorted by the arriving signal.

Dolphin reconstructs the self signal using a pre-learned *reference symbol*. Before signal transmission, each device sends itself a symbol carrying bit 1. This phase is called *beep*, and the symbol is called reference symbol. Next, the device arranges the reference symbol according to its own message's binary sequence: putting the symbol at where bit 1 appears to cancel the self signal.

Directly using the reference symbol cannot obtain a good SIC result because we have not considered distortions caused by transmission. *Sampling distortion* is caused by

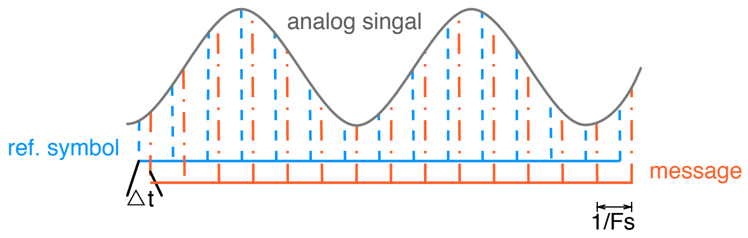


Figure 4.11: Sampling Distortion

the offset between two digital-to-analog procedures — recording reference symbols and recording message. This is illustrated in Figure 4.11, where Fs is the sampling frequency.

The offset Δt is always smaller than $1/Fs$. Fixing Δt, we can use discrete Fourier transform (DFT) to shift a sample sequence (blue lines) by Δt. Fourier transform states that a signal $x(n)$ can be represented by the summation of a series of sinusoids

$$x[n] = \sum_{k=0}^{N/2} (ReX[k]\cos(2\pi kn/N) + ImX[k]\sin(2\pi kn/N)),$$

where N is the number of samples, $X[k]$ is the k-th frequency domain sample, and $ReX[k]$, $ImX[k]$ are its real and image parts. The signal can be losslessly reconstructed if its frequency is less than $Fs/2$. Therefore, we use equation

$$x'[n] = \sum_{k=0}^{N/2} (ReX[k]\cos(2\pi kn/N + 2\pi k\Delta tFs/N)$$
$$+ImX[k]\sin(2\pi kn/N + 2\pi k\Delta tFs/N))$$

to obtain the shifted sample sequence $x'[1..N]$. Technically, the shift operations are done in the frequency domain. Dolphin first transforms a time domain sequence to the frequency domain, then applies a fractional offset filter [80] to the sequence, and finally transforms them back to the time domain.

To perform SIC, we need to locate the start point of a tone in the received signal. The location process introduces *Symbol synchronization distortion*. Dolphin locates a tone by seeking a sharp power jump in the signal. Therefore, the located point may be a few samples away from the true start point. This error makes the reference symbol not line up with a symbol in the received message signal.

Given a reference symbol ref, a self message m, and a received, interfered sample sequence s, Dolphin performs SIC in the following way. SIC has three parameters w_0, w_{reg}, and β. w_0 and w_{reg} denote the position search range for the first tone and the rest tones, respectively. β is the offset increment. Our implementation used $w_0 = 128$, $w_{\text{reg}} = 40$, and $\beta = 20$. We note that the first message bit $m[0]$ is always set to 1 for synchronization purpose.

1. Dolphin locates the the start point p of the first tone in s and initializes a message pointer $p_m \leftarrow 0$.

2. If $m[p_m] = 0$, go to 6.

3. If $p_m = 0$, set $r \leftarrow w_0$; otherwise, set $r \leftarrow w_{\text{reg}}$.

4. For all $p' \in [p - r \,..\, p + r]$ and all $\Delta t = \frac{\alpha}{\beta \cdot Fs}$, where $\alpha \in [0..\beta]$, Dolphin computes $p_{\min}$ and $\Delta t_{\min}$ such that

$$(p_{\min}, \Delta t_{\min}) = \operatorname*{argmin}_{p', \Delta t} \sum_{k=1}^{L} (s[p' + k] - ref^{\Delta t}[k])^2,$$

   where L is the symbol length and $ref^{\Delta t}$ denotes shifting reference symbol by Δt.

5. Perform SIC: $s[p_{\min} + k] = s[p_{\min} + k] - ref^{\Delta t_{\min}}[k]$, for $1 \leq k \leq L$.

6. If p_m is less than the message length, $p \leftarrow p + L$, $p_m \leftarrow p_m + 1$ and go to 2. Otherwise, SIC terminates.

We remark that, in 4, Dolphin searches $2r$ neighbors around point p to find a best matched position. We search a larger range w_0 at the first tone position than the rest to cancel the impact of the start mismatch.

4.4.6 *Putting The Pieces Together*

Dolphin places the two device speakers as close as possible. From a distant attacker's point of view, the two acoustic signals seem to come from the same point and are difficult to separate. The inseparability protects the confidentiality of the resulting assertions. Figure 4.12 shows microphone and speaker positions on iPhone (left) and Macbook (middle). The rightmost picture shows the placement of the two devices in Dolphin. iPhone and Macbook touch each other at the speakers. iPhone microphone is 1cm away from the touch point while Macbook microphone is 6cm away.

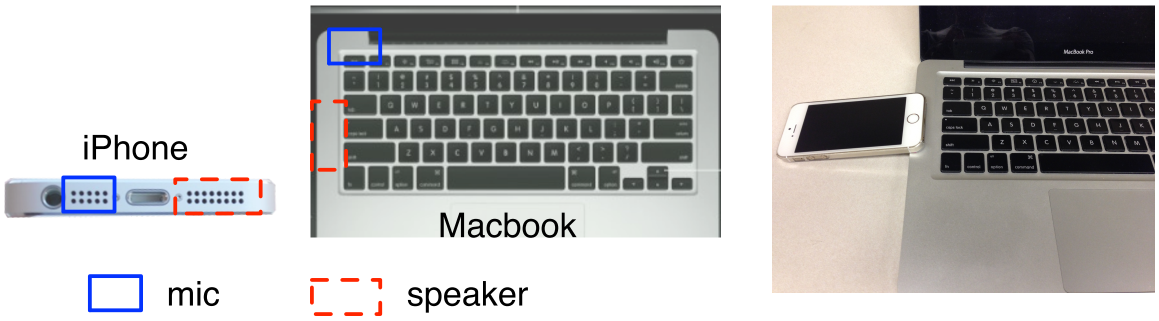


Figure 4.12: Placing Two Device Speakers Closely. Rightmost Is How We Place a Smartphone in Dolphin

First, Dolphin selects a safe distance d. When a smartphone and a terminal are properly placed, Dolphin works as follows. We note that when we say "sending message" we mean sending over the local network.

- The terminal sends a start request to the smartphone. Upon receiving the request, the smartphone measures and estimates the message delivery time $\overline{rtt}/2$ (Section 4.4.4).
- The smartphone sends a "start measurement" message to the terminal. The terminal starts to measure ambient SPL when the message arrives, while the smartphone holds for $\overline{rtt}/2$ seconds to start measurement. Both devices determine the needed signal

amplitude as discussed in Section 4.4.3. In the meantime, the devices respectively choose a 16-bit random number with the first bit fixed to 1. Appending a 4-bit CRC code, each device obtains a 20-bit message to be sent out. Using OOK modulation (Section 4.4.2), each device converts the binary sequence to a 16kHz signal.

- After the above preparation, the smartphone sends a "start transmission" message to the terminal. The terminal emits its signal as soon as it receives the message. The smartphone holds for $\overline{rtt}/2$ to transmit its signal.

- Each device performs SIC over the received signal, i.e., a sequence of samples (Section 4.4.5). The resulting sample sequence is fed to the OOK demodulator to obtain the message from the other party. CRC check is performed over the message. If the check fails, the device reports the failure and requests for another round of protocol. Otherwise, the first 16-bit binary sequence is extracted. Both devices append smartphone's binary sequence to the terminal's. The 32-bit number is the final near field assertion.

- For the "Assertion Only" scenario, each device signs the assertion, encrypts the signature using the other party's public key, and sends it to the other party for validation. For the "Authentication" scenario, the two devices execute a password based authenticated key exchange protocol (PAKE) [15, 24] to derive a shared session key. The 32-bit number serves as the shared password in PAKE.

4.4.7 Security Discussion

Distance restriction is guaranteed by the physical properties of sound. The inverse distance law of acoustics and BER-SNR relation of OOK modulation theoretically guarantee that beyond the safe distance the emitted acoustic signal's power drops to such a low value that it is impossible to correctly demodulate the message.

Confidentiality is achieved by the interference of the two signals. An attacker cannot separate or recover the two signals if he is oblivious to both. Ideally, two symbols interfere in three cases "1-1", "1-0", and "0-0". The "0-0" case is easy to detect. For the "1-0" case, an attacker has to determine which party emits bit 1. Addressing this problem requires the knowledge of at least one message. For the "1-1" case, it is difficult for an attacker to determine whether it is a single 1-bit contaminated by noise or the combination of two 1-bits. The real cases are more complicated. For example, determining whether it is a "1-1" case is not easy because signals may cancel each other (see Appendix 4.7).

In Section 4.4.5, we mentioned that during SIC, range w_0 is set to be sufficiently large to tolerate start mismatch. Particularly, choice of w_0 depends on the security level of the implementation. A larger w_0 tolerates a larger mismatch but makes the two signals more vulnerable to separation and puts the confidentiality of the assertions at higher risk. If an implementation does not require confidentiality, it can use a large w_0 to make the system more robust. More attacks are discussed below.

Relay Attacks. An honest party only accepts an arriving signal that starts the first tone at most w_0 samples later than itself does, leaving an extremely small time window for relaying messages. Our implementation uses a time window of $128/44.1 = 2.9$ms. The time window can be squeezed or relaxed by changing w_0. This means that our relay resistance is **adjustable** and thus flexible to be adopted by different applications.

Source Separation Attacks. An attacker may try to separate the two signals by using blind audio source separation (BASS) that has been studied in acoustics [102]. However, BASS algorithms separate audio sources based on the the sound's acoustic pattern, such as frequency, sound pressure level, etc. An example is to separate human speech from background music. However, given a combined message signal in Dolphin, it is difficult to distinguish between a symbol sent by the smartphone and a symbol sent by the terminal. This makes BASS impossible to separate the two signals. We will verify this property in Section 4.5.2.

Multi-reception Attacks. Attackers place multiple microphones near the two communicating devices trying to separate the two signals as in a multi-input multi-output (MIMO) system. However, MIMO needs senders and receivers collaborate to frequently measure the channel information between them because the channels are time-varying. Our threat models assume that an attacker does not control any device and thus cannot carry out such attacks.

Delaying Attacks. Since an attacker controls the local network connections, he can delay the smartphone's probe messages to increase its measured message delivery time but delivers the "start transmission" message as normal. In this way, the smartphone's signal transmission is delayed and the two signals are easy to separate. However, any delay larger than the time window bounded by w_0 causes demodulation failure and voids the resulting assertions.

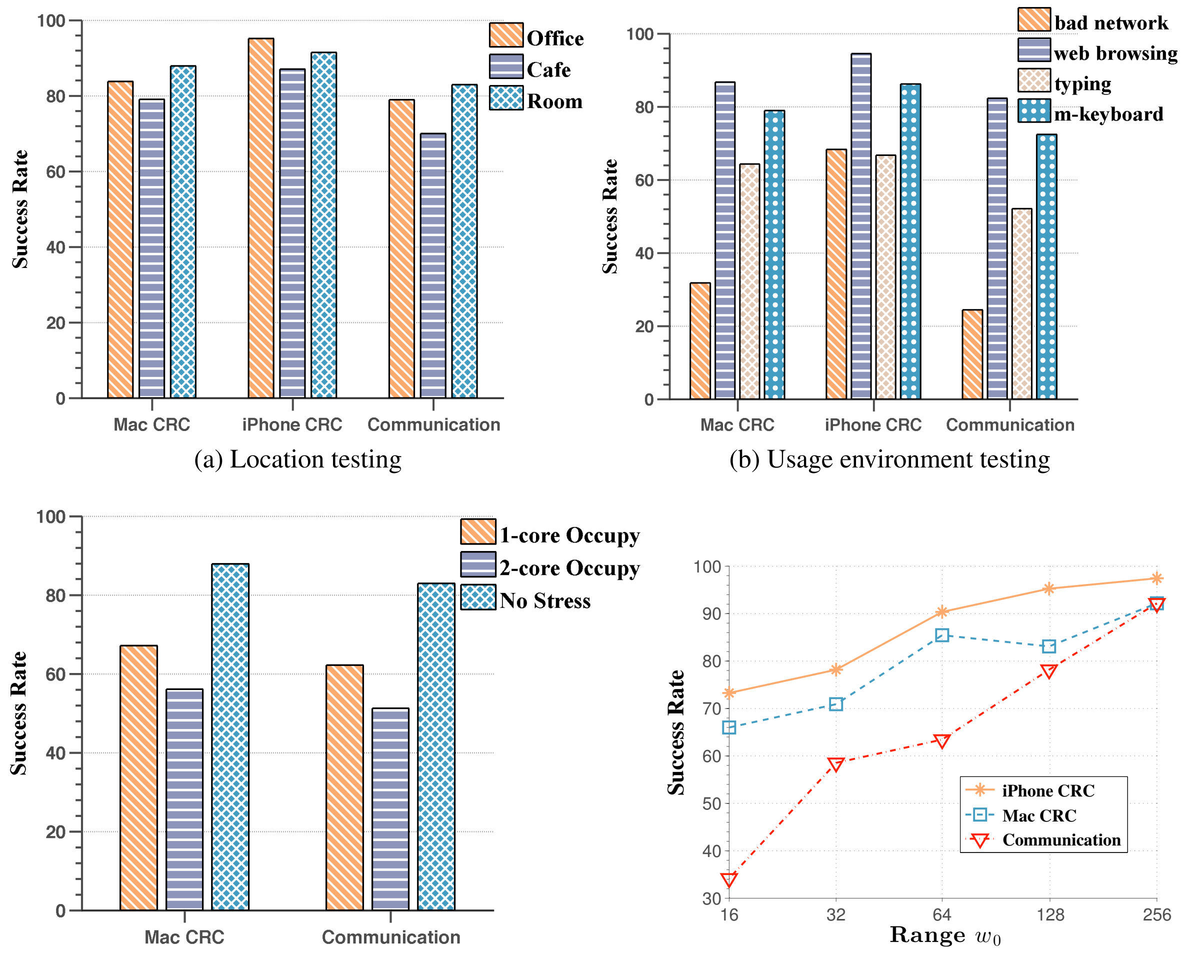
Placement Attacks. Attackers try to find a vantage position where the arriving time gap between the two signals is stretched to a sufficiently large value to separate them. By triangle inequality, we know that the arriving time gap is no more than d_e/v_s, where d_e is the distance between the two speakers and v_s is the speed of sound, i.e., 340m/s. Dolphin requires the two speakers to stay close and d_e is smaller than 0.1m, resulting in a mismatch of 13 ($0.1 \times 44100/340$) samples.

Shouting Attacks. During the course of measuring ambient SPL, an attacker creates a loud sound to make the devices transmit acoustic signals with a higher power than needed. In this way, the attacker attempts to force two distant devices to generate a valid assertion. First, we note that a device speaker cannot sound arbitrarily large and such an attack fails when the two devices are far away from each other. In out implementation, Macbook and iPhone both sound less than 100dBSPL at 1cm which means that the SNR drops to 0 beyond 3.16 meters ($10^{\frac{50}{20}}$cm) in a 50dBSPL environment. In addition, to achieve a wanted distance d', an attacker has to lift the ambient noise SPL by $20\log_{10}\frac{d'}{d}$. For example, $d = 15$cm in Dolphin, an attacker has to increase the noise by 16dBSPL so as to attack two devices that are 1 meter apart. This 16dBSPL increment is so loud ($> 2\times$ loud [99]) to arouse a device owner's suspicion. Since Dolphin is designed as an automated and long-running service, it can measure the ambient SPL at random time points and reject any suspiciously sharp increment.

Distance Fraud. In this attack, one of the two communicating devices is malicious and tries to get valid assertions when it is beyond the safe distance. We remark that this attack is not the main attack Dolphin aims to defend against because our threat model assumes that both devices are honest. We argue that such attacks are difficult to succeed. First, an attacker has to find some way to increase the SNR that drops to a useless level beyond the safe distance. He can either use a directional microphone to increase the arriving signal power or execute an active noise control system to reduce the noise power. However, using a directional microphone in close proximity is very suspicious and an active noise control system is not practical in an enclosed space as large as a room [92].

4.5 Performance Evaluation

We have designed a series of tests to evaluate the performance of Dolphin, including success rate under various conditions, security performance, execution time, and energy usage



(a) Location testing (b) Usage environment testing

(c) Stress testing (d) Testing w_0

Figure 4.13: Success Rate Testings

on smartphones.

4.5.1 Success Rate

Before presenting the results, we remark that Dolphin uses 4-bit CRC code to detect bit errors and repeats the protocol if an error occurs. Hence, in practice, Dolphin guarantees to successfully achieve a valid assertion. Comparing with contextual co-presence approaches, Dolphin enables a communicating party to locally check the validity of an assertion via CRC check. We suppressed the redo part of Dolphin in the evaluation of success rate. In the following experiments, each device performs one signal transmission per execution of Dolphin. We say the communication is successful if the two resulting assertions are identical.

First, we tested our system at three locations, a living room, an office with A/C working, and a coffee shop during busy hours. Table 4.2 lists the average ambient noise SPL and statistics of network ping time at the three locations.

Table 4.2: Statistics of Locations

	Office	**Coffee Shop**	**Room**
Noise avg (dBSPL)	59.4	67.3	42.8
RTT avg (ms)	3.801	9.468	6.055
RTT stddev (ms)	1.922	10.666	2.407

At each location, we let iPhone and Macbook conduct the protocol and do 150 signal transmissions. After each transmission, we record whether the CRC check succeeds on both devices and whether the communication succeeds. The success rates are shown in Figure 4.13a. The communication success rate is the highest in the living room (82.9%) and the lowest in the noisy coffee shop (70%). The success rate in the office is also high, although the ambient noise SPL is not low. The reason is that the noise in the office is mostly caused by A/C (which is stable), while the coffee shop often has a noise burst, e.g., running an espresso machine or blending coffee beans. We also notice that the iPhone has a very high CRC success rate, because its microphone is very close to the two speakers and captures the signal much more clearly than the Macbook does. Inspired by this observation, we ran Dolphin on two iPhones, an iPhone 5 and an iPhone 5s. We put the two iPhones face up, bottom to bottom, and 1cm apart. Finally, we obtained a high communication success rate — 96.3%, 98.2%, and 95.7% at the office, living room, and coffee shop, respectively.

Next, we tested our implementation under a high latency WiFi environment and three human usage environments: web browsing, heavy typing on the build-in keyboard, and heavy typing on an external USB keyboard. Each testing performed 150 signal transmissions. During the testings, we kept a non-stop web browsing or keyboard typing while the system was running. To make a loud noise, we use an old fashioned mechanical keyboard when using an external keyboard in testing. The resulting success rates are shown in Figure 4.13b. Bad WiFi connection suppresses the communication success rate to 21%. The network RTT has a mean of 939.67ms and a standard deviation of 243.488. The low success rate is caused by the high latency variance which makes Dolphin's latency estimation inaccurate. In bad network testing, iPhone has a high CRC success rate because the mismatch is so large that the iPhone only heard its own signal. Typing on the build-in keyboard results in a lower success rate than typing on the external one, because the keys around `CAPS` key on the build-in keyboard are close to the speakers and typing on these keys affects the signals very much. Web browsing does not affect the success rate since it uses less touch to the `CAPS` key area.

The first search range w_0 determines Dolphin's tolerance to the signal mismatch and affects the success rate. We executed Dolphin using different w_0, which was set to 16, 32, 64, 128, and 256 respectively. The resulting success rates are depicted by Figure 4.13d. We observe that the growth of w_0 dramatically increases the success rate.

Dolphin synchronizes two signals by estimating system latency. Multiple tasks contending resources may make the estimation inaccurate and fail the communication. We tested the success rate of Dolphin while letting one or two CPU-intensive processes occupy CPU resources. We ran an infinite AES encryption process to occupy one of the two cores in the CPU and Geekbench 3 to occupy both. The testing is performed on Macbook side because 1) working on computer and putting smartphone aside is a common scenario in smartphone assisted system; 2) the testing is already sufficient to show the success rate change. The resulting success rates are shown in Figure 4.13c. The success rate is reduced to 62% when one core is occupied and 50% when both are occupied. Mac OSX scheduler tries to assign equal resources to long-running jobs [91]. Contention between jobs makes the latency estimation not accurate and thus fails the synchronization and the corresponding SIC phase.

4.5.2 Security Performance

Dolphin utilizes fast decaying acoustic signal to restrict the distance between two devices. We placed iPhone (resp. Macbook) speaker at different distances to Macbook(resp. iPhone) microphone and let the speaker transmit signals. The received SNRs are shown in Figure 4.14a. We can see that doubling distance reduces SNR by nearly 6dB, which verifies the inverse distance law, see Equation (4.1). This makes obtaining valid assertions beyond the safe distance impossible.

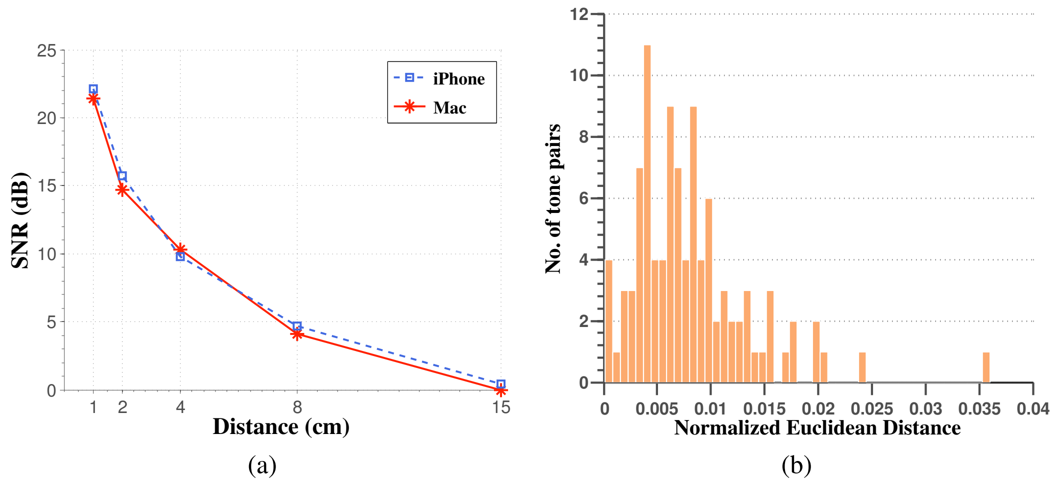
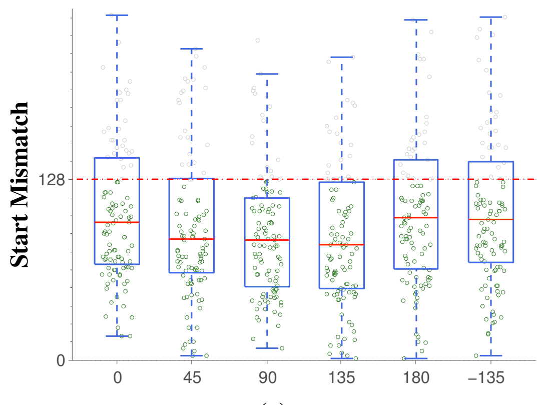


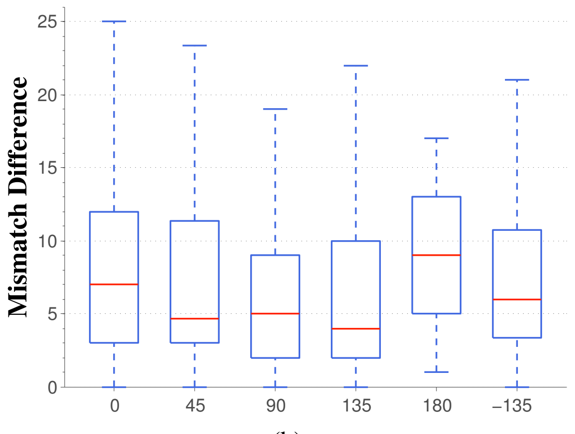
Figure 4.14: (a): SNR vs. Distance (b): Distance of two tones

One may worry about whether the two signals can be separated by their individual patterns by using, for example, machine learning approaches. We argue that the two sine wave tones produced by Macbook and iPhone are almost identical by showing the normalized Euclidean distance between them. The Macbook emits a tone first and the iPhone sounds later. We use a third iPhone to record the two tones, obtaining two sample sequences. The distance is calculated as follows: 1) use cross correlation to find a start point in each sequence such that beginning from the point, the two sequences are optimally matched; 2)obtain two sequences, x and y, truncated from the start points; 3) normalize x, y using min-max normalization and calculate the normalized Euclidean distance between them. We calculated the distance for 100 pairs of tones, and plotted the results in Figure 4.14b. As can be seen from the figure, most distances are less than 0.015, indicating that the two tones in most cases are almost identical.

As discussed previously, an attacker cannot separate two signals by placing a device at a vantage position with respect to the touch point of Macbook and iPhone. This ensures the confidentiality feature of Dolphin. We placed a smartphone ("attacker") to collect two signals at various positions around the touch point. Particularly, the smartphone is placed at 10cm to the touch point but at different angles. In our device placement (Figure 4.12), let the positive direction be the direction from the touch point to the rightmost. If we draw a circle centred at the touch point, a line connecting the centre and a point on the circle forms an angle with the positive direction. An angle is positive if the positive direction turns clockwise to the line. We placed the smartphone at the position of angles of $0°$, $45°$, $90°$, $135°, 180°$, and $-135°$. For each position, the two devices transmitted 100 pairs of signals. The attacker captured a pair and calculated the start mismatch as in Section 4.4.4. Figure 4.15a shows the results in boxplots and the x-axis lists the position angles. We can see that for all testing positions most of the signal pairs are entangled, i.e.,



(a)



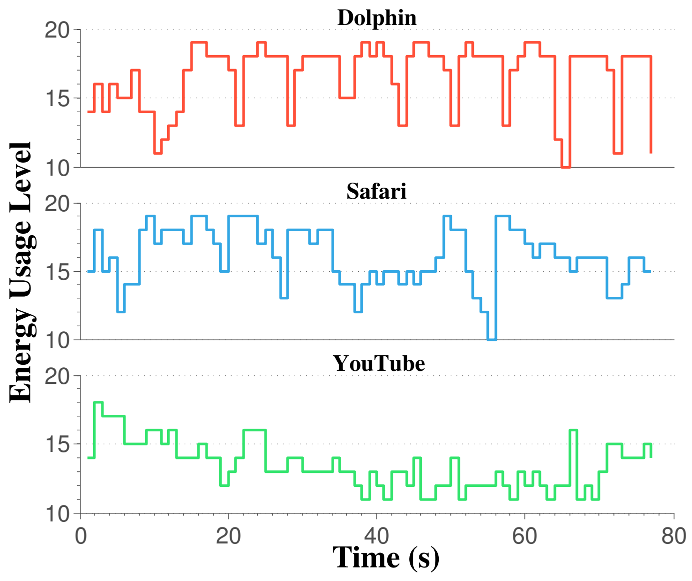
(b)

Figure 4.15: Mismatch of Two Signals Obtained by an Attacker on Various Positions

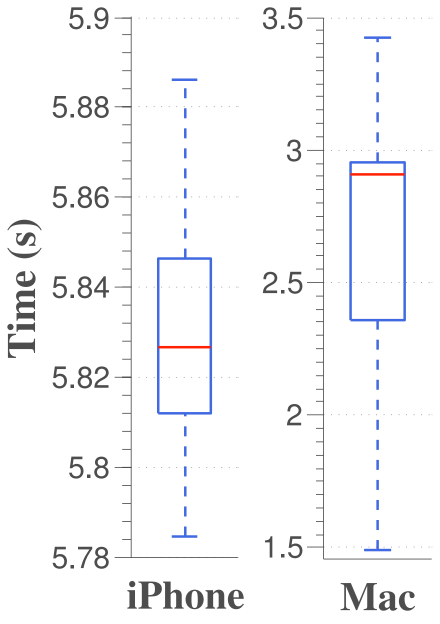
start mismatch (green points) is smaller than 128. An attacker's advantage is the difference between the start mismatch obtained by the attacker and the one obtained by an honest party. In Figure 4.15b, we show the attacker's advantages for all testing pairs. We observe that most (>80%) of the attacker's advantages are below 15, which conforms to our previous analysis, and all of them are smaller than 25. Therefore, it is of small probability that an attacker obtains two separated signals while the two honest parties do not notice it. From the figure, we also notice that an attacker maximizes his advantage by placing devices on positions of 0° and 180°, whose median (red line) is obviously higher than the rest. This also verifies our analysis and suggests that these two positions should be especially cleared during the system execution.

4.5.3 Energy Usage

Dolphin is designed to be running on smartphones, where energy saving is vital. By using iPhone's logging for developers, we tracked the energy consumptions of Dolphin as well as Safari and Youtube for comparison purpose. The test device is an iPhone 5s running iOS 7. The log file is transferred to and visualized in XCode Instrument Tool which uses unit Energy Usage Level on a scale of 0-20 to indicate the current energy usage. The track lasted for 30 minutes and energy usage is logged per second. Figure 4.16a shows a snapshot of the logged results. It is clear that watching online video consumed the least



(a)



(b)

Figure 4.16: (a)Dolphin's Energy Usage Level; (b)Signal Processing Time on iPhone and MacBook

energy and Dolphin consumed nearly the same energy as Safari did. Specifically, the ratio of Dolphin's total energy consumption to Safari's is 1.03. Such an energy consumption achieves our goal of sustainability, since web browsing is not a battery killer task and can last for nearly 10 hours [101].

The execution time of Dolphin consists of two main parts — signal transmission and the following signal processing, including FFT, SIC, etc. The signal transmission costs $700 \times 20/44100 = 0.317$ seconds. To get signal processing time, we ran Dolphin for 100 times and calculated the average time for signal processing. The result is shown in Figure 4.16b. The time values of iPhone squeeze around the mean of 5.83 seconds while the time values of Macbook have a large standard deviation but a smaller mean, 2.68 seconds.

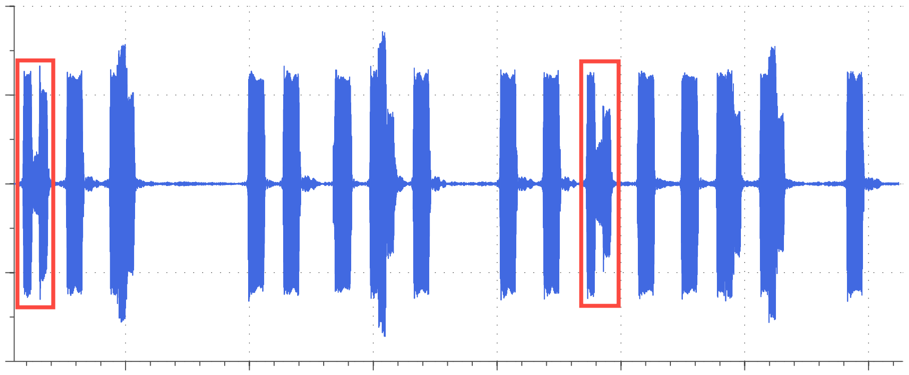
4.6 Conclusions

We have designed and implemented Dolphin — an acoustic near field assertion system. Dolphin utilizes the fast decay property of acoustic signals to restrict distance, and full-duplex communication to defend against relay attacks. It has an adjustable safe distance (for restricting distance) and an adjustable time window (for defending against relay attacks), generates confidential assertions, needs zero user interactions, and is battery friendly. Extensive experiments are carried out to evaluate the performance of Dolphin.

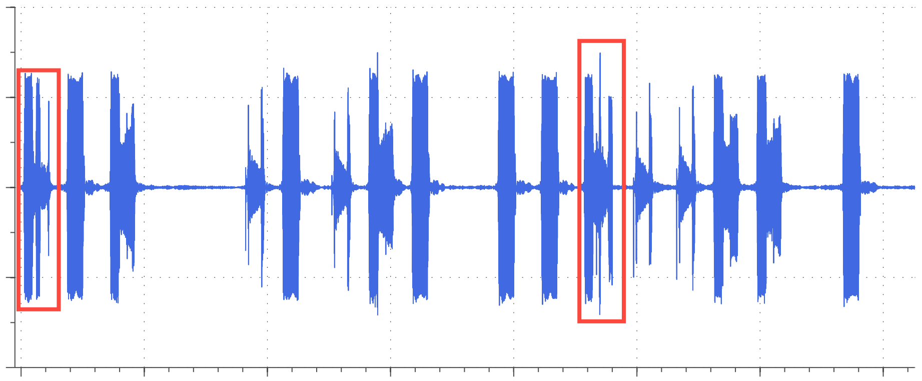
4.7 Illustrating SIC and Demodulation Process by An Example

We use an example to illustrate the SIC and demodulation process of Dolphin. All the plots here use sample index as x-axis and signal amplitude as y-axis.

Macbook and iPhone simultaneously send binary sequences `11100010110111001101` and `10100101100001111100`, respectively. Figure 4.17a shows the signal that was captured by the iPhone and filtered by the HPF (Section 4.4.1).

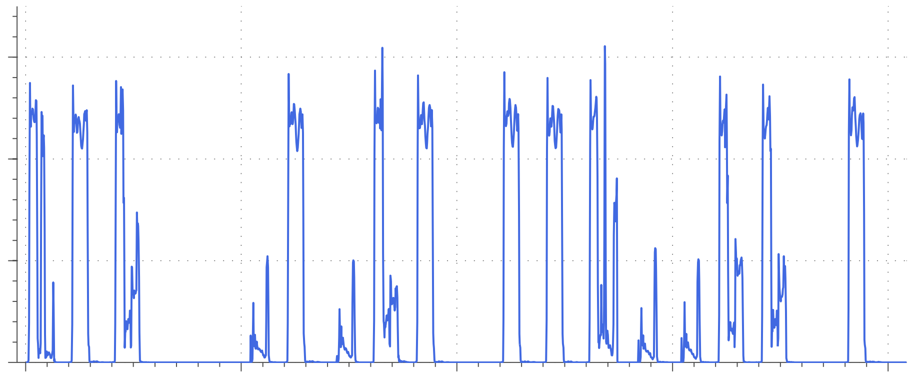


(a) Signal at iPhone: after HPF, before SIC

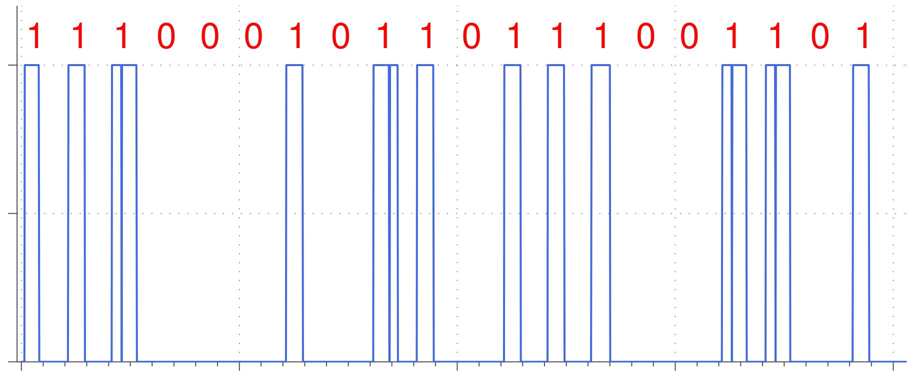


(b) Signal at iPhone: after SIC, before Squaring

An interesting observation is that the two symbols in some "1-1" symbol pairs cancel each other, highlighted by red boxes. The rest of the "1-1" pairs have the two symbols add to each other. This observation shows that detecting the "1-1" case in a real signal transmission is difficult for attackers. The signal obtained after applying SIC is shown in 4.17b.



(c) Signal at iPhone: after Squaring and LPF, before digitalization



(d) Signal at iPhone: decoded message

Figure 4.17: Digital Signal Processing on iPhone

SIC has filled part of the cancellation in the two special “1-1” symbol pairs, which makes the correct decoding possible for the communicating party. Next, the signal is squared and filtered by a low pass filter (LPF), shown in Figure 4.16c. Finally, we digitalize the signal by comparing each sample value against the mean value of the first symbol (Figure 4.16d). We observe that the iPhone successfully extracted the sequence `11100010110111001101` transmitted by the Macbook.

REFERENCES

[1] BUMP TECHNOLOGIES.

[2] International standard: Iso/iec 11770-4:2006(e). http://webstore.iec.ch/preview/info_isoiec11770-4%7Bed1.0%7Den.pdf, 2006.

[3] What is s beam, and how do i use it on my samsung galaxy s iii? http://www.samsung.com/us/support/supportOwnersHowToGuidePopup.do?howto_guide_seq=7042&prd_ia_cd=N0000003&map_seq=48157, 2013.

[4] M. Abdalla, C. Chevalier, and D. Pointcheval. Smooth projective hashing for conditionally extractable commitments. In *Advances in Cryptology-CRYPTO 2009*, pages 671–689. Springer, 2009.

[5] M. Abdalla and D. Pointcheval. Simple password-based encrypted key exchange protocols. In *CT-RSA*, pages 191–208, 2005.

[6] A. Ahmed and I. Traore. Anomaly intrusion detection based on biometrics. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 452–453. IEEE, 2005.

[7] A. Ahmed and I. Traore. A new biometric technology based on mouse dynamics. *IEEE Transactions on Dependable and Secure Computing*, 4(3):165–179, 2007.

[8] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO*, pages 255–270, 2000.

[9] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *NDSS*, 2002.

[10] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *NDSS*, 2002.

[11] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. Nist special publication 800-57. *NIST Special Publication*, 800(57):1–142, 2007.

[12] J. R. Barry, E. A. Lee, and D. G. Messerschmitt. *Digital communications*. Springer, 2004.

[13] L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar. Device-enabled authorization in the Grey system. In *Information Security: 8th International Conference, ISC 2005*, pages 431–445. Springer, Sept. 2005.

[14] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in CryptologyEurocrypt 2000*, pages 139–155. Springer, 2000.

[15] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *IEEE S&P 1992.*, pages 72–84. IEEE, 1992.

[16] F. Bergadano, D. Gunetti, and C. Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):367–397, 2002.

[17] Biometric Signature ID. BSI recommends all multi factor authentication include gesture biometrics to prevent data breaches to existing secure systems for highest cyber security. http://www.biosig-id.com/bsi-recomm ends-all-multi-factor-authentication-include-gesture-biometrics-to-prevent-data-breaches-to-existing-secure-systems-for-highest-cyber-security/, 2012.

[18] S. Bluetooth. Simple pairing whitepaper. Technical report, Technical report, Bluetooth SIG, 2006. http://www. bluetooth. com/Bluetooth/Apply/Technology/Research/Simple_Pairing. htm, 2006.

[19] S. Brands and D. Chaum. Distance-bounding protocols. In *Advances in Cryptology - EUROCRYPT'93*, pages 344–359. Springer, 1994.

[20] E. Bresson, O. Chevassut, and D. Pointcheval. New security results on encrypted key exchange. *Public Key Cryptography–PKC 2004*, page 145.

[21] E. Bresson, O. Chevassut, and D. Pointcheval. Security proofs for an efficient password-based key exchange. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 241–250. ACM, 2003.

[22] C. Busold, A. Taha, C. Wachsmann, A. Dmitrienko, H. Seudié, M. Sobhani, and A.-R. Sadeghi. Smart keys for cyber-cars: secure smartphone-based nfc-enabled car immobilizer. In *ACM CODASPY 2013*, pages 233–242. ACM, 2013.

[23] Canalys Inc. Smart phones overtake client PCs in 2011, http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011, 2012.

[24] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. MacKenzie. Universally composable password-based key exchange. In *Advances in Cryptology - EUROCRYPT 2005*, pages 404–421. Springer, 2005.

[25] C. Castelluccia and P. Mutaf. Shake them up!: a movement-based pairing protocol for cpu-constrained devices. In *MobiSys*, pages 51–64, 2005.

[26] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[27] T. Clancy, N. Kiyavash, and D. Lin. Secure smartcardbased fingerprint authentication. In *Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications*, pages 45–52. ACM, 2003.

[28] W. Claycomb and D. Shin. Extending formal analysis of mobile device authentication. *Journal of Internet Services and Information Security*, 1(1):86–102, 2011.

[29] W. R. Claycomb and D. Shin. Secure device pairing using audio. In *Security Technology, 2009. 43rd Annual 2009 International Carnahan Conference on*, pages 77–84. IEEE, 2009.

[30] M. Conti, I. Zachia-Zlatea, and B. Crispo. Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call. In *Proceedings of ASIACCS2011*, pages 249–259, Hong Kong, China, 2011. ACM.

[31] H. Corrigan-Gibbs and B. Ford. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM conference on Computer and Communications Security*, pages 340–350. ACM, 2010.

[32] A. Czeskis, M. Dietz, T. Kohno, D. S. Wallach, and D. Balfanz. Strengthening user authentication through opportunistic cryptographic identity assertions. In *ACM CCS 2012*, pages 404–414, 2012.

[33] A. De 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 2012 ACM annual conference on Human Factors in Computing Systems*, pages 987–996. ACM, 2012.

[34] R. Dhamija and A. Perrig. Deja vu: A user study using images for authentication. In *USENIX Security 2000*, pages 45–58. Usenix Denver, CO, 2000.

[35] B. Duc, S. Fischer, and J. Bigun. Face authentication with gabor information on deformable graphs. *IEEE Transactions on Image Processing*, 8(4):504–516, 1999.

[36] A. Francillon, B. Danev, and S. Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS 2011*, 2011.

[37] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis. Practical nfc peer-to-peer relay attack using mobile phones. In *Radio Frequency Identification: Security and Privacy Issues*, pages 35–49. Springer, 2010.

[38] L. Francis, G. P. Hancke, K. Mayes, and K. Markantonakis. Practical relay attack on contactless transactions by using nfc mobile phones. *IACR Cryptology ePrint Archive*, 2011:618, 2011.

[39] H. Gamboa and A. Fred. A behavioral biometric system based on human-computer interaction. In *Proceedings of SPIE*, volume 5404, pages 381–392, 2004.

[40] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and clear: Human-verifiable authentication based on audio. In *ICDCS*, page 10, 2006.

[41] Google Inc. Android developer guide. http://developer.android.com/guide/basics/what-is-android.html, Feb 2012.

[42] Google Inc. Android sensor manager, http://developer .android.com/reference/android/hardware/sensorma nager.html, Feb. 2012.

[43] A. Groce and J. Katz. A new framework for efficient password-based authenticated key exchange. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 516–525. ACM, 2010.

[44] T. Halevi, D. Ma, N. Saxena, and T. Xiang. Secure proximity detection for nfc devices based on ambient sensor data. In *ESORICS 2012*, Sep.

[45] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *IEEE Symposium on Security and Privacy*, pages 129–142, 2008.

[46] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *Ubicomp*, pages 116–122, 2001.

[47] D. P. Jablon. Strong password-only authenticated key exchange. *ACM SIGCOMM Computer Communication Review*, 26(5):5–26, 1996.

[48] D. P. Jablon. Extended password key exchange protocols immune to dictionary attack. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 1997. Proceedings., Sixth IEEE Workshops on*, pages 248–255. IEEE, 1997.

[49] Jaden. iOS 5 iphone lockscreen glitch allows you to bypass password protection and explore the phone.app! http://www.ijailbreak.com/iphone/ios-5-iphone-lockscreen-glitch/, Feb. 2012.

[50] M. Jain, J. I. Choi, T. Kim, D. Bharadia, S. Seth, K. Srinivasan, P. Levis, S. Katti, and P. Sinha. Practical, real-time, full duplex wireless. In *ACM MobiCom 2011*, pages 301–312. ACM, 2011.

[51] S. Jarecki and X. Liu. Fast secure computation of set intersection. In *SCN*, pages 418–435, 2010.

[52] Y. Jing, G.-J. Ahn, Z. Zhao, and H. Hu. Riskmon: Continuous and automated risk assessment for mobile applications. In *Proceedings of 4th ACM Conference on Data and Applications Security (CODASPY)*, pages 99–110. ACM, 2014.

[53] John A. How to reset your android lock screen password. http://droidlessons.com/how-to-reset-your-android-lock-screen-password/, Mar. 2012.

[54] Z. Jorgensen and T. Yu. On mouse dynamics as a behavioral biometric for authentication. In *Proceedings of the ASIACCS2011*, pages 476–482, Hong Kong, China, 2011.

[55] A. Juels. Rfid security and privacy: A research survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, 2006.

[56] J. Katz, R. Ostrovsky, and M. Yung. Efficient and secure authenticated key exchange using weak passwords. *Journal of the ACM (JACM)*, 57(1):3, 2009.

[57] J. Katz and V. Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In *Advances in Cryptology–ASIACRYPT 2009*, pages 636–652. Springer, 2009.

[58] J. Krumm and K. Hinckley. The nearme wireless proximity server. In *ACM UbiComp 2004*, pages 283–300. Springer, 2004.

[59] S. Laur and K. Nyberg. Efficient mutual data authentication using manually authenticated strings. In *CANS*, pages 90–107, 2006.

[60] S. Lawson, May 2011.

[61] A. Levi, E. Çetintaş, M. Aydos, Ç. K. Koç, and M. U. Çağlayan. Relay attacks on bluetooth authentication and solutions. In *ISCIS 2004*, pages 278–288. Springer, 2004.

[62] L. Li, X. Zhao, and G. Xue. An identity authentication protocol in online social networks. In *ASIACCS'2012*. ACM, April 2012.

[63] L. Li, X. Zhao, and G. Xue. A lightweight system to authenticate smartphones in the near field without nfc chips. In *Communications (ICC), 2013 IEEE International Conference on*, pages 6391–6395. IEEE, 2013.

[64] L. Li, X. Zhao, and G. Xue. Near field authentication for smart devices. In *INFOCOM, 2013 Proceedings IEEE*, pages 375–379. IEEE, 2013.

[65] L. Li, X. Zhao, and G. Xue. Unobservable re-authentication for smartphones. In *NDSS*, 2013.

[66] C. H. Lim and P. J. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *Advances in CryptologyCRYPTO'97*, pages 249–263. Springer, 1997.

[67] Lookout Inc. Mobile phone lost and found. https://www.lookout.com/resources/reports/mobile-lost-and-found/billion-dollar-phone-bill, 2012.

[68] G. Lynch. *Single Point of Failure: The Ten Essential Laws of Supply Chain Risk Management*. Wiley, 2009.

[69] D. Ma, N. Saxena, T. Xiang, and Y. Zhu. Location-aware and safer cards: Enhancing rfid security and privacy via location sensing. *Dependable and Secure Computing, IEEE Transactions on*, 10(2):57–69, 2013.

[70] P. MacKenzie. On the security of the speke password-authenticated key exchange protocol. *IACR Cryptology ePrint Archive*, 2001:57, 2001.

[71] J. Mantyjarvi, M. Lindholm, E. Vildjiounaite, S. Makela, and H. Ailisto. Identifying users of portable devices from gait pattern with accelerometers. In *Proceedings of ICASSP'05*, volume 2, pages ii–973. IEEE, 2005.

[72] R. Mayrhofer and H. Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *IEEE Trans. Mob. Comput.*, 8(6):792–806, 2009.

[73] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy*, pages 110–124, 2005.

[74] F. Monrose, M. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. *International Journal of Information Security*, 1(2):69–83, 2002.

[75] F. Monrose and A. Rubin. Authentication via keystroke dynamics. In *Proceedings of the 4th ACM conference on Computer and communications security*, pages 48–56. ACM, 1997.

[76] Y. Nakkabi, I. Traoré, and A. Ahmed. Improving mouse dynamics biometric performance using variance reduction via extractors with separate features. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40(6):1345–1353, 2010.

[77] R. Nandakumar, K. K. Chintalapudi, V. Padmanabhan, and R. Venkatesan. Dhwani: secure peer-to-peer acoustic nfc. In *ACM SIGCOMM 2013*, pages 63–74. ACM, 2013.

[78] NIST, CSE. *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program*, Mar. 2011. Available at http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf.

[79] F. Okumura, A. Kubota, Y. Hatori, K. Matsuo, M. Hashimoto, and A. Koike. A study on biometric authentication based on arm sweep action with acceleration sensor. In *Proceedings of ISPACS'06*, pages 219–222. IEEE, 2006.

[80] S.-C. Pei and Y.-C. Lai. Closed form variable fractional time delay using fft. *Signal Processing Letters, IEEE*, 19(5):299–302, 2012.

[81] A. Perrig and D. Song. Hash visualization: A new technique to improve real-world security. In *International Workshop on Cryptographic Techniques and E-Commerce*, pages 131–138, 1999.

[82] M. Pusara and C. Brodley. User re-authentication via mouse movements. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 1–8. ACM, 2004.

[83] M. Qi, Y. Lu, J. Li, X. Li, and J. Kong. User-specific iris authentication based on feature selection. In *Proceedings of ICCSSE 2008*, volume 1, pages 1040–1043. IEEE, 2008.

[84] D. R. Raichel. *The science and applications of acoustics*. Springer, 2006.

[85] Rohde & Schwarz. Near field communication (nfc) technology and measurements white paper.

[86] H. Rydberg. Multi-touch protocol. . http://www.mjmwired.net/kernel/Documentation/input /multi-touch-protocol.txt, 2009.

[87] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan. Secure device pairing based on a visual channel (short paper). In *IEEE Symposium on Security and Privacy*, pages 306–313, 2006.

[88] N. Saxena and J. Watt. Authentication technologies for the blind or visually impaired. In *Proceedings of the 4th USENIX conference on Hot topics in security*, pages 7–7. USENIX Association, 2009.

[89] D. Schurmann and S. Sigg. Secure communication based on ambient audio. *IEEE Transactions on Mobile Computing*, 12(2):358–370, 2013.

[90] B. Shrestha, N. Saxena, H. T. T. Truong, and N. Asokan. Drone to the rescue: Relay-resilient authentication using ambient multi-sensing. In *FC 2014*, 2014.

[91] A. Singh. *Mac OS X Internals: A Systems Approach*. Addison-Wesley Professional, 2006.

[92] S. D. Snyder. *Active noise control primer*. Springer Science & Business, 2012.

[93] F. Stajano and R. Anderson. The resurrecting duckling: security issues for ubiquitous computing. *Computer*, 35(4):22–26, 2002.

[94] F. Stajano and R. J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols Workshop*, pages 172–194, 1999.

[95] D. Stinson. *Cryptography: theory and practice*. CRC press, 2006.

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

[97] G.-S. Tian, Y.-C. Tian, and C. Fidge. High-precision relative clock synchronization using time stamp counters. In *IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2008.*, pages 69–78. IEEE, 2008.

[98] H. T. T. Truong, X. Gao, B. Shrestha, N. Saxena, N. Asokan, and P. Nurmi. Comparing and fusing different sensor modalities for relay attack resistance in zero-interaction authentication. In *IEEE PerCom 2014*, 2014.

[99] G. Vanderheiden. About Decibels (dB). http://trace.wisc.edu/docs/2004-About-dB/, 2004.

[100] S. Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *CRYPTO*, pages 309–326, 2005.

[101] Vince. Apple iPhone 5s battery life test. http://blog.gsmarena.com/apple-iphone-5s-battery-test/, 2013.

[102] E. Vincent, M. G. Jafari, S. A. Abdallah, M. D. Plumbley, and M. E. Davies. Blind audio source separation. *Centre for Digital Music, Queen Mary University of London, Technical Report C4DM-TR-05-01*, 2005.

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

[104] X. Zhao, L. Li, G. Xue, and G. Silva. Efficient anonymous message submission. In *INFOCOM, 2012 Proceedings IEEE*, pages 2228–2236, Orlando, FL, USA, March 2012. IEEE.

[105] Z. Zhao and G.-J. Ahn. Using instruction sequence abstraction for shellcode detection and attribution. In *Proceedings of 2013 IEEE Conference on Communications and Network Security (CNS)*, pages 323–331. IEEE, 2013.

[106] Z. Zhao, G.-J. Ahn, J.-J. Seo, and H. Hu. On the security of picture gesture authentication. In *Proceedings of 22nd USENIX Security Symposium (Security)*. USENIX, 2013.

[107] N. Zheng, A. Paloski, and H. Wang. An efficient user verification system via mouse movements. In *Proceedings of ACM CCS2012*, pages 139–150. ACM, 2011.

[108] N. Zheng, A. Paloski, and H. Wang. An efficient user verification system via mouse movements. In *ACM Conference on Computer and Communications Security*, pages 139–150, 2011.

[109] E. Zwicker, H. Fastl, and H. Frater. *Psychoacoustics, Facts And Models, volume 22 of Springer Series of Information Sciences*. Springer, Berlin, 1999.