

2014

## Exploiting behavioral biometrics for user security enhancements

Nan Zheng

*College of William & Mary - Arts & Sciences*

Follow this and additional works at: <https://scholarworks.wm.edu/etd>



Part of the [Computer Sciences Commons](#)

---

### Recommended Citation

Zheng, Nan, "Exploiting behavioral biometrics for user security enhancements" (2014). *Dissertations, Theses, and Masters Projects*. William & Mary. Paper 1539623640.  
<https://dx.doi.org/doi:10.21220/s2-3jya-ts34>

This Dissertation is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact [scholarworks@wm.edu](mailto:scholarworks@wm.edu).

**EXPLOITING BEHAVIORAL BIOMETRICS FOR USER SECURITY  
ENHANCEMENTS**

**Nan Zheng**

**Xianyang, Shaanxi, China**

**Master of Science, University of Tennessee Knoxville, 2009**

**Bachelor of Science, University of Science and Technology of China, 2006**

**A Dissertation presented to the Graduate Faculty  
of the College of William and Mary in Candidacy for the Degree of  
Doctor of Philosophy**

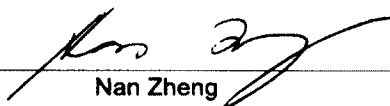
**Department of Computer Science**

**The College of William and Mary  
May, 2014**

## APPROVAL PAGE

This Dissertation is submitted in partial fulfillment of  
the requirements for the degree of

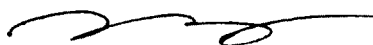
Doctor of Philosophy



---

Nan Zheng

Approved by the Committee, May, 2014



---

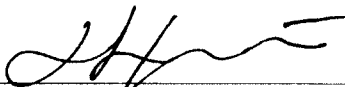
Committee Chair

Associate Professor Haining Wang, Computer Science  
The College of William & Mary



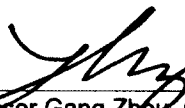
---

Professor Weizhen Mao, Computer Science  
The College of William & Mary




---

Professor Evgenia Smirni, Computer Science  
The College of William & Mary



---

Associate Professor Gang Zhou, Computer Science  
The College of William & Mary



---

Dr. Kun Bai

IBM T.J. Watson Research Center

## COMPLIANCE PAGE

Research approved by

Protection of Human Subject Committee

Protocol number(s): PHSC-2010-05-19-6734-hxwan3

Date(s) of approval: 05/21/2010

## ABSTRACT

As online business has been very popular in the past decade, the tasks of providing user authentication and verification have become more important than before to protect user sensitive information from malicious hands. The most common approach to user authentication and verification is the use of password. However, the dilemma users facing in traditional passwords becomes more and more evident: users tend to choose easy-to-remember passwords, which are often weak passwords that are easy to crack. Meanwhile, behavioral biometrics have promising potentials in meeting both security and usability demands, since they authenticate users by "who you are", instead of "what you have". In this dissertation, we first develop two such user verification applications based on behavioral biometrics: the first one is via mouse movements, and the second via tapping behaviors on smartphones; then we focus on modeling user web browsing behaviors by Fitts' Law.

Specifically, we develop a user verification system by exploiting the uniqueness of people's mouse movements. The key feature of our system lies in using much more fine-grained (point-by-point) angle-based metrics of mouse movements for user verification. These new metrics are relatively unique from person to person and independent of the computing platform. We conduct a series of experiments to show that the proposed system can verify a user in an accurate and timely manner, and induced system overhead is minor. Similar to mouse movements, the tapping behaviors of smartphone users on touchscreen also vary from person to person. We propose a non-intrusive user verification mechanism to substantiate whether an authenticating user is the true owner of the smartphone or an impostor who happens to know the passcode. The effectiveness of the proposed approach is validated through real experiments. To further understand user pointing behaviors, we attempt to stress-test Fitts' law in the "wild", namely, under natural web browsing environments, instead of restricted laboratory settings in previous studies. Our analysis shows that, while the averaged pointing times follow Fitts' law very well, there is considerable deviations from Fitts' law. We observe that, in natural browsing, a fast movement has a different error model from the other two movements. Therefore, a complete profiling on user pointing performance should be done in more details, for example, constructing different error models for slow and fast movements. As future works, we plan to exploit multiple-finger tapplings for smartphone user verification, and evaluate user privacy issues in Amazon wish list.

## TABLE OF CONTENTS

Acknowledgements	ii
Dedications	iii
List of Tables	iv
List of Figures	v
Chapter 1. Introduction	1
Chapter 2. An Efficient User Verification System via Mouse Movements	10
Chapter 3. User Verification on Smartphones via Tapping Behaviors	42
Chapter 4. Exploring Fitts' Law in Web Browsing	72
Chapter 5. Conclusion and Future Work	99
Bibliography	103

## ACKNOWLEDGEMENTS

My first and foremost appreciation goes to my Ph.D. advisor, Dr. Haining Wang, for his excellent guidance and great inspirations. Dr. Wang's research insights and unique approach of solving research problems will definitely make a far-reaching impact to my career. I especially thank Dr. Wang for his patience and confidence in me.

In addition, I would like to thank my fellow group members and colleagues at the College of William & Mary, including but not limited to Yubao Zhang, Dachuan Liu, Haitao Xu, Zhang Xu, Aaron Paloski, Jing Jin, Steven Gianvecchio, Xin Ruan, Aaron Koehl, Bo Dong, Lingfei Wu, Xin Qi and David T. Nguyen for their great support, advices, and discussions during the course of my doctoral pursuit.

I would also like to thank my collaborators Dr. Kun Bai and Dr. Hai Huang at IBM T.J. Watson Research Center for their indispensable efforts and influences on my career path. Also thanks to Dr. Ting He, Dr. Kanat Tangwongsan and Dr. Kang-Won Lee at IBM T.J. Watson Research Center who have patiently mentored and inspired me during my internship.

In addition, I want to thank my doctoral committee members for their valuable directions.

My gratitude also extends to all the other faculty members, staff, fellow graduate students, who helped make my five years at the College of William & Mary a great time.

Finally, my deepest thanks go to my parents, whose unconditional love and support inspired me to complete my journey of doctoral study even during the most difficult periods.

To my parents



## LIST OF TABLES

2.1	Setting Details	20
2.2	Variation of FRR and FAR with Different Number of Clicks in One Block	29
2.3	Comparison with Existing Works	37
3.1	Collected Data	45
3.2	Features of Touchscreen Tapping Behaviors	48
3.3	User Verification Accuracies	61
4.1	Fitts' Law Related Parameters for Users of Mouse	92
4.2	Fitts' Law Related Parameters for Users of Both Mouse and Touchpad	94

## LIST OF FIGURES

2.1	Illustration of angle based metrics	14
2.2	Direction Angle metric plotted for two different users on two different machines each	16
2.3	Angle of Curvature metric plotted for two different users on two different machines each	16
2.4	Curvature Distance metric plotted for two different users on two different machines each	17
2.5	Speed metric plotted for two different users under two different environments each	18
2.6	Pause-and-click metric plotted for two different users undertow different environments each	18
2.7	Distances from one user's curvature angle distribution to those of others, as well as to itself in different settings	19
2.8	System Architecture	21
2.9	SVM hyperplane in two dimensional feature space [36]	23
2.10	Variation of FRR and FAR with the number of clicks. Error bars indicate standard deviation.	27
2.11	Variation of FRR and FAR with threshold	28
2.12	Variation of FRR and FAR with majority votes	29
2.13	FRR and FAR for one user profiled on one platform and tested on another platform	30
2.14	ROC curves with and without partial movements	32
2.15	The entire angle range $360^\circ$ is divided into eight sections. The eight divisions of angle, each with a resolution of $45^\circ$ , are labeled with numbers 1 to 8. For each section the	

	average values of metrics are calculated as features, in order to differentiate between mouse and touchpad data.	35
2.16	The classification accuracy in differentiating mouse and touchpad data, as a function of increasing number of mouse events per block.	37
3.1	Screen layout of our data collection application, and the two-hand typing action	47
3.2	An illustration of two-feature space of a target user and many others. $X_1$ and $X_2$ are the two features. The dashed lines define the boundary of the target user's behavior. Because the target user's behavior is limited to a concentrated area, the boundary blocks the majority of potential impostors.	49
3.3	Timing of tapping on the smartphone from three different users, shown in three vertical panels. Each user typed 20 times of the number string "3244". The solid dots represent key-press time, and the open dots are key-release time. Different colors represent the timestamps of different digits.	50
3.4	User's tapping pressure on smartphone touchscreen, while entering an 8-digit PIN 1-2-5-9-7-3-8-4. Each figure shows pressure readings on a 3×3 smartphone number pad. Darker color indicates a larger tapping pressure. Note that number "6" has no pressure because its is not in the PIN. Figures in the same row are from a same user while typing the PIN the three times.	52
3.5	Distribution of dissimilarity score of typing 3-2-4-4 from a target user's template, to both the target user itself and other 52 users.	53
3.6	Distribution of dissimilarity score of typing the 8-digit PIN 1-2-5-9-7-3-8-4 from a target user's template, to both the target user itself and other users.	53

3.7	Distribution of dissimilarity score of typing another 8-digit PIN 1-2-5-9-8-4-1-6 from a target user's template, to both the target user itself and other users.	54
3.8	System Architecture	55
3.9	An illustration of tradeoff between FRR and FAR by tuning the threshold [55]	59
3.10	Variation of average EER with number of user actions in one-class training for five PIN combinations.	60
3.11	Variation of FRR and FAR with the value of threshold for 3-2-4-4 and the two 8-digit PINs for one target user	60
3.12	Comparison of performance from each feature set in one-class learning, as well as when they are combined together	61
3.13	Variation of Accuracies with number of users in training and testing	62
3.14	Trade-off between FRR and FAR in typing three 4-digit PINs: 1-1-1-1, 3-2-4-4, and 5-5-5-5	63
3.15	Comparison of accuracies between one- and two-class learning. Lengths of error bars show the standard deviation of EER over users.	63
3.16	Dissimilarity scores from one tapping position (i.e., one-handed while sitting) to the other two tapping positions of a same user. Upper and lower panels are for the two different users, respectively.	65
3.17	Effect of mimic attack shown in degree of dissimilarity from the target user. Subfigures, from left to right, correspond to all features considered, and each individual feature set (acceleration, pressure, size, and time). There are two mimic "imposters", and 10 trials before and after their observations on the target user.	67

4.1	Step-wise movement towards target [35]	74
4.2	Layout of the webpage for data collection. Text links are highlighted by red boxes.	78
4.3	ID vs. Mean Time plotted for the forum users' data set. The linear correlation is over 98%. Every data point in the figure is averaged over 180 raw data points.	81
4.4	Probability distribution of $E_{\%}$ , the relative deviation from Fitts' prediction, from raw times of natural web browsing. Histogram bars denote the portion of $E_{\%}$ with values in the corresponding block. Below the histogram is the scattering of raw data, where the location of dense area (between - 50% and +50%) indicates the most probable values of $E_{\%}$ .	82
4.5	Mean absolute percentage error (MAPE) drops exponentially with the increase of cluster size. A cluster size $S$ means every $S$ raw data are clustered and averaged. Note that above 40 actions per cluster, MAPE stabilizes at around 10%.	84
4.6	CDF for the effective width $W_e$ for all movements and for the fast movements. Note that both distributions are not Gaussian, as were assumed in prior works.	85
4.7	CDF for the effective width $W_e$ for the medium and slow movements.	87
4.8	ID vs. Mean Time plotted for all collected physical mouse traces and all touchpad traces. Each data point in the figure is a result of averaging 180 raw data records. Note that the linear correlation is similar for both devices (over 98% in both cases).	87
4.9	ID vs. Mean Time plotted for User 1's physical mouse trace and touchpad mouse trace. Each data point in the figure is a result of averaging 10 raw data records.	88

4.10	ID vs. Mean Time plotted for User 2's physical mouse trace and touchpad mouse trace. Each data point in the figure is a result of averaging 10 raw data records.	89
4.11	ID vs. Mean Time plotted for User 3's physical mouse trace and touchpad trace #1. Each data point in the figure is a result of averaging 10 raw data records.	90
4.12	ID vs. Mean Time plotted for User 3's physical mouse trace and touchpad trace #2. Each data point in the figure is a result of averaging 10 raw data records.	91
4.13	ID vs. Mean Time plotted for User 3's touchpad traces #1 and #2. Each data point in the figure is a result of averaging 10 raw data records.	91
4.14	ID vs. Standard Deviation of Fitts' law calculation for the forum users' data set. Note the linear relationship.	92
4.15	Click inaccuracy	95

# 1 Introduction

User security, implied by its name, refers to security that is both usable and trustworthy to end-users. Traditionally, security experts are primarily concerned with security, i.e., preventing malicious parties from damaging or peeking the user sensitive information. However, as the weakest link in the security chain, end-users tend to favor usability over security in using security-related applications. It is up to end-users, who are not so tech-savvy, to decide whether to install/uninstall a security-related software and the choice of passwords or passcodes. More often, there is a discrepancy between traditional security goals and end-user interests. Taking access control as an example, security experts are more concerned with blocking malicious hackers; while end-users concern more on easy to remember/use and not locking themselves out.

Researchers have recently come to an agreement that, in making a security system usable, one needs to put usability first in the design cycle. In fact, CRA (Computing Research Association) lists one of the grand challenges as to “*give end-users security controls they can understand and privacy they can control for the dynamic, pervasive computing environments of the future*” [27]. In this dissertation, we present our efforts in designing two user verification systems, with the goal of enhancing user security, and in exploring Fitts’ law—one of the fundamental law in HCI (Human-Computer Interaction)—to model user behaviors of web browsing. For our future work, we plan to explore two directions: using multi-finger tapplings for increased security in smartphone user verification, and

evaluating privacy risks in publicly accessible Amazon wish list.

## **1.1 Mouse Dynamics**

In today's Internet-centered world, the tasks of user authentication and verification have become more important than ever before [3, 21, 76, 88]. For highly sensitive systems such as online banking, it is crucial to secure users' accounts and protect their assets from malicious hands. Even in less critical systems such as desktop machines in a computing laboratory, online forums, or social networks, a hijacked session can still be misused to spread viruses or post spam, possibly damaging a user's reputation and other systems. The most common approach to securing access to systems is the use of a password [23, 41]. Unfortunately, passwords suffer from two serious problems: password cracking and password theft [90, 115]. Once a password is compromised, an adversary can easily abuse a victim's account. Thus, there is a great demand to quickly and accurately verify that the person controlling a given user's account is who the user claims to be, termed re-authentication [88].

Existing user verification and re-authentication methods require human involvement, such as providing secret answers to agreed-upon questions. Unfortunately, they only provide one-time verification, and the verified users are still vulnerable to both session hijacking and the divulging of the secret information. To achieve a timely response to an account breach, more frequent user verification is needed. However, frequent verification must be passive and transparent to users, as continually requiring a user's involvement for re-authentication is too obtrusive and inconvenient to be acceptable.

In the first project, we propose a biometric-based approach to verifying users based on passively observable mouse movement behaviors. In general, in order for a re-authentication system to be practical, it must have the following features:



- Accuracy. Not only must the system accurately identify an impostor, it must also have the probability of rejecting a true user close to zero, to avoid inconvenience to true users.
- Quick response. The system should make a quick verification decision. In other words, it should be able to distinguish a user in a timely manner.
- Difficult to forge. Even if a user's profile template is known by an impostor, it will be very hard to mimic the normal biometric behaviors in a consistent manner and then evade the verification system.

Our new approach meets all of these challenges, delivering an accurate and quick verification based on biometrics which are difficult to forge. The basic working mechanism of our approach is to passively monitor the mouse movements of a user, extract angle-based metrics, and then use Support Vector Machines (SVMs) for accurate user verification. The key feature of our approach is to exploit the point-by-point angle-based metrics of mouse movements, which are relatively unique from person to person and independent of the computing platform, for user verification.

Current biometric approaches are limited in applicability: physiological biometrics, such as fingerprints and retinal scans, provide accurate one-time authentication but require specialized hardware which may be expensive or unavailable on all users' machines. On the other hand, behavioral biometrics such as keystroke and mouse dynamics hold promise, because they can be obtained from common user interface (UI) devices that nearly every user can be assumed to own. Compared with keystroke dynamics [64, 77, 84], mouse dynamics has its own advantage for two reasons. First, keystroke monitoring can record sensitive user credentials like usernames and passwords, raising much more serious privacy concerns than mouse movement monitoring. Second, keyboard is much more complex than mouse in structure, and thus keystroke dynamics are more easily

affected by different kinds of keyboards in terms of shape, size, and layout.

However, to date the existing mouse-based user verification approaches have either resulted in unacceptably low accuracy or have required an unacceptably long amount of time to reach a decision, making them unsuitable for online re-authentication. In contrast to previous research, our approach introduces a novel way—point-by-point angle-based metrics—to characterize users' mouse movements, which significantly reduces verification time while keeping high accuracy.

We perform a measurement-based study, derived from 30 *controllable* users and a corpus of more than 1,000 real users in the field. Based on these two sets of mouse movement data, we evaluate the effectiveness of the proposed system through a series of experiments, using the set of angle-based metrics specifically chosen for being both platform-independent and widely variant from user to user.

In summary, the major contributions of this work include:

- We model behavioral biometrics using mouse dynamics, and develop an efficient user verification system. It achieves high accuracy and significantly outperforms existing systems in terms of verification time.
- We propose a novel measurement strategy involving a carefully chosen set of angle-based metrics, which is relatively independent of the operating environment and capable of uniquely identifying individual users.
- We conduct an experiment involving sessions from over 1,000 unique users, which is able to re-authenticate a user within just a few clicks with a high accuracy. This promising result could lead to a practical user verification system, suitable for online deployment in the future.

## **1.2 Smartphone User Verification via Tapping Behaviors**

Smartphones have become ubiquitous computing platforms allowing users any-time access to the Internet and many online services. On one hand, as a personal device, a smartphone contains important private information, such as text messages, always-logged-in emails, and contact list. On the other hand, as a portable device, a smartphone is much easier to get lost or stolen than conventional computing platforms. Therefore, to prevent the private information stored in smartphones from falling into the hands of adversaries, user authentication mechanisms have been integrated into mobile OSes like Android and iOS.

Due to having a much smaller screen and keyboard on a smartphone than the traditional user input/output devices, PIN-based and pattern-based passcode systems have been widely used in smartphones for user authentication. However, many people tend to choose weak passcodes for ease of memorization. A 2011 survey on iPhone 4-digit passcode reveals that the ten most popular passcodes represent 15% of all 204,508 passcodes and the top three are 1234, 0000, and 2580 [5]. Moreover, recent studies show that an attacker can detect the location of screen taps on smartphones based on accelerometer and gyroscope readings and then derive the letters or numbers on the screen [14, 75, 82, 109]. An attacker could even exploit the oily residues left on the screen of a smartphone to derive the passcode [7]. Therefore, it is highly desirable to enhance the smartphone's user authentication with a non-intrusive user verification mechanism, which is user-transparent and is able to further verify if the successfully logged-in user is the true owner of a smartphone.

In the second project, we explore the feasibility of utilizing user tapping behaviors for user verification in a passcode-enabled smartphone. The rationale

behind our work is that individual human users have their own unique behavioral patterns while tapping on the touch screen of a smartphone. In other words, you are how you touch on the screen, just like you are how you walk on the street. The rich variety of sensors equipped with a smartphone including accelerometer, gyroscope, and touch screen sensors, make it possible to accurately characterize an individual user's tapping behaviors in a fine-grained fashion. With over 80 smartphone users participated in our study, we quantify the user tapping behaviors in four different aspects: acceleration, pressure, size, and time. Based on the behavioral metrics extracted from these four features, we apply the one-class learning technique for building an accurate classifier, which is the core of our user verification system.

We evaluate the effectiveness of our system through a series of experiments using the empirical data of both 4-digit and 8-digit PINs. In terms of accuracy, our approach is able to classify the legitimate user and impostors with averaged equal error rates of down to 3.65% for 4-digit PINs. For 8-digit PINs, we achieve even lower equal error rates of down to 3.21%. Overall, our verification system can significantly enhance the security of a smartphone by accurately identifying impostors. Especially for practical use, our tapping-behavior-based approach is user-transparent and the usability of traditional passcodes on a smartphone remains intact. As our approach is non-intrusive and does not need additional hardware support from smartphones, it can be seamlessly integrated with the existing passcode-based authentication systems.

### **1.3 Exploring Fitts' Law in Web Browsing**

Web browsing is mainly driven by *target acquisition* – the movement of a pointing device, such as a mouse, touchpad, and stylus, to an on-screen target. The ability to quantify the way these pointing actions are performed has been studied heavily

before. The previous research works have helped us better understand the effect human motor control has on pointing actions, with applications ranging from the design of more efficient graphical user interfaces (GUIs) [12], to the creation of accurate pointing devices [42], and human biometrics security [88, 116].

Fitts' law [39] is a classic and well-studied law, which quantifies pointing actions in terms of the size, distance, and time to reach the target. The most common formulation of Fitts' law comes from [70] and is referred to as the "Shannon formulation":

$$MT = a + b \cdot \log_2 \left( \frac{A}{W} + 1 \right). \quad (1.1)$$

In Eq. 1.1, the log term is known as the index of difficulty ( $ID$ ). Fitts' law describes a linear relationship between the mean time to complete the pointing action ( $MT$ ) and the index of difficulty ( $ID$ ) of the pointing task, which includes the distance to the target ( $A$ ) and the width of the target in the direction of movement ( $W$ ). The parameters  $a$  and  $b$  are environment- and user-specific.

This law has been one of the most widely used and most well-respected formulas in human-computer interaction, but it represents an idealized view of pointing actions. Nearly all of the experimental results currently in the literature regarding Fitts' law have been performed in extremely clinical settings – on a blank background, a single target square or circle appears, and a user must move the cursor from a starting position to the target as quickly as possible. This style of pointing, however, is not typical of real-world GUI applications. Typically, there are many potential targets on the screen at once. The environment is full of distractions, instead of a blank white screen, and the user spends time considering his or her next move, instead of pointing as quickly as possible. All of these environmental factors might affect the amount of time it takes for a user to complete a pointing action. There is very little research detailing how well Fitts' law applies, if at all, in such scenarios.

Moreover, Fitts' law is not merely a single equation in a vacuum. The law de-

scribes a linear model of human pointing actions, and a model cannot be defined by its mean alone. Other side metrics such as the standard deviation or variance of the model should also be considered when discussing Fitts' law, but the majority of the current literature seems to focus exclusively on the mean as presented by the common formulation.

This work attempts to answer the question: how well does Fitts' law truly model real human pointing tasks in web browsing? We examine Fitts' law in a natural web browsing environment to determine its validity outside of a structured experimental setting. This is accomplished through a data set collected from 1,047 users' natural mouse traces on a real-world website. The major contributions of this work are summarized as follows:

- an application of the Fitts' law formula to pointing actions in a natural web browsing environment, involving a large-scale data collection from 1,047 real-world users on an Internet forum (Section 4.2.1), to assess Fitts' law's applicability to typical GUIs outside of an experimental setting (Section 4.3.1);
- an observation that in web browsing, *fast* movements have a different error model from *slow* movements, which deviates from previous laboratory studies. We speculate that this is partially due to the *open-loop* nature of fast movements (Section 4.3.2);
- a comparison of Fitts' law results for natural browsing using two different pointing devices – physical mouse and laptop touchpad – to determine whether the choice of pointing device has an effect on the linear relationship described by Fitts' law (Section 4.3.3);
- an analysis of the standard deviation of Fitts' law calculations of mean pointing time, to better understand the variance present in the Fitts model (Section 4.3.4).

## **1.4 Future Works**

We leave two open topics for our future work. First, we are going to extend the user verification on smartphones to multiple-finger tapplings, which promisingly provides higher level of robustness and security. Second, we plan to investigate various privacy issues from publicly accessible Amazon wish lists. The detailed research plans are described in Chapter 5.

## **1.5 Organization**

The remainder of this dissertation is structured as follows. Chapter 2 presents our first work in developing an efficient user verification system via mouse movements, including the detailed data collection, measurement and evaluation process. Chapter 3 presents our second work in verifying smartphone users via their tapping behaviors, where a series of experiments are conducted to validate its efficacy in reality. Chapter 4 describes our third project in exploring Fitts' law in web browsing, which is for further understanding people's behaviors when interacting with computers. Chapter 5 concludes and proposes our future research plans on exploring multi-finger tapplings for smartphone user verification and privacy issues in Amazon wish list.

## **2 An Efficient User Verification System via Mouse Movements**

In this chapter, we present our work on verifying users via their mouse movements, which outperforms previous works in terms of both efficiency and accuracy. The chapter is structured as follows. Section 2.1 reviews the background and related work in the area of mouse dynamics. Section 2.2 describes our data collection and measurement, including our choice of angle-based metrics over more traditional metrics. Section 2.3 details the proposed classifier for user verification. Section 2.4 presents our experimental design and results. Section 2.5 discusses issues which arise from the details of our approach, and Section 2.6 concludes.

### **2.1 Background and Related Work**

The underlying principle of biometric-based user authentication is centered on “who you are”. This is very different from conventional user authentication approaches, which are mainly based on either “what you have” or “what you know”. Unfortunately, a physical object such as a key or an ID card can be lost or stolen. Similarly, a memorized password could be forgotten or divulged to malicious users. Conversely, a biometric-based approach relies on inherent and unique characteristics of a human user being authenticated. The biometrics can never



be lost or forgotten, nor can another user easily steal or acquire them. This makes biometrics very attractive for user authentication.

Biometrics are categorized as either physiological or behavioral [110]. Physiological biometrics, like fingerprint and facial recognition, have attracted considerable attention in research [49,72]. The downside of these biometrics is that they need specialized hardware, which can be problematic for wide deployment. For user authentication over the Internet, one cannot always rely on the existence of hardware at the client side. In contrast, behavioral biometrics using human-computer interaction (HCI) can record data from common input devices, such as keyboards and mice, providing user authentication in an accessible and convenient manner.

Behavioral biometrics first gained popularity with keystroke dynamics with Monroe et al.'s work on password hardening in 1999 [76]. Later on, Ahmed and Traore [2] proposed an approach combining keystroke dynamics with mouse dynamics. Mouse dynamics for re-authentication have been previously studied as a standalone biometric by Pusara and Brodley [88]. Unfortunately, their study is inconclusive with only eleven users involved, prompting the authors to conclude that mouse biometrics are insufficient for user re-authentication. Our study relies on an improved verification methodology and far more users, leading us to reverse their hypothesis.

In Ahmed et al.'s work [2,3,78], while achieving very high accuracy, the number of mouse actions needed to verify a user's identity<sup>1</sup> is too high to be practical. Specifically, their experiment requires as many as 2,000 aggregate mouse actions before a user can be recognized, and is not practical for real-time deployment. Conversely, we aim to provide a system suitable for online re-authentication. We first employ a finer-grained data collection methodology, allowing us to collect far more data in less time. We also employ support vector machines (SVMs), which

---

<sup>1</sup>referred to as *session length* in [3, 78].

are considerably faster than the neural networks employed in [3, 78]. Thus, our system can make a decision in just several mouse clicks.

More recently, a survey covering the existing works in mouse dynamics has been conducted with a comparative experiment [58]. It points out that mouse dynamics research should be more aware to reduce verification time and take the effect of environmental variables into account. It can be seen later that, compared to other works, our approach also achieves high accuracy but only requires a small amount of biometric data. Moreover, we explore the effects of environmental factors (different machines, mice, and time) and show that our approach is relatively robust across different operating environments and times.

Graphical passwords [21, 100] are a related form of user authentication, relying on HCI through a pointing device to authenticate a user. Mouse dynamics differ in that they differentiate between users by *how* the users move and click the mouse, rather than *where* the users click. Graphical passwords record where the user clicks on the screen, and subsequently use this sequence as a substitute password. Systems such as these are complementary to our work, and can be deployed together. For instance, one might employ a graphical password system while passively recording a user's mouse dynamics, utilizing the passively recorded measurements as a secondary failsafe to verify the user's identity. This is similar in spirit to using keystroke dynamics with password hardening as in [76].

## **2.2 Measurement and Characterization**

### **2.2.1 Data Collection**

We collect two sets of data. The first data set is from a controllable environment, referred to as the controllable set; while the second data set is from an online forum in the field, referred to as the field set. We have obtained approval from the Institutional Review Board (IRB) of our university, which ensures the appropriate

and ethical use of human input data in our work.

For the controllable set, 30 users are invited personally to participate in the data collection. They are from different ages, educational backgrounds, and occupations. We intentionally set a normal environment for these users and inform them to behave as naturally as possible. Mouse movement data are recorded during their routine computing activities. These activities range among word processing, surfing the Internet, programming, online chatting, and playing games. We make use of a logging tool RUI [67] to record their mouse movement activities.

For the field set, more than 1,000 unique forum users' mouse movements are recorded by JavaScript code, and submitted passively via AJAX requests to the web server. On one hand, these users are anonymous but identifiable through unique login names. However, there is no guarantee on the amount of data collected for a certain user. A forum user could be logged in for a long time with frequent mouse activities, or could perform just one click and then leave. On the other hand, the breadth of this corpus of users is utilized to serve as the base profile for both training and testing purposes.

The raw mouse movements are represented as tuples of timestamp and Cartesian coordinate pairs. Each tuple is in the form of  $\langle \text{ACTION-TYPE}, t, x, y \rangle$ , where ACTION-TYPE is the mouse action type (a *mouse-move* or *mouse-click*),  $t$  is the timestamp of the mouse action,  $x$  is the x-coordinate, and  $y$  is the y-coordinate. Timestamps in our data collection are collected in milliseconds.

## Data Processing

The purpose of preprocessing is to identify every point-and-click action, which is defined as continuous mouse movements followed by a click. *Continuous* mouse movements are series of mouse movements with little or no pause between each adjacent step. Within the  $i$ th point-and-click action for a user  $c$ , we denote the  $j$ th mouse move record as  $\langle \text{MOUSE-MOVE}, t_i, x_i, y_i \rangle_{c,j}$ , where  $t_i$  is the timestamp of

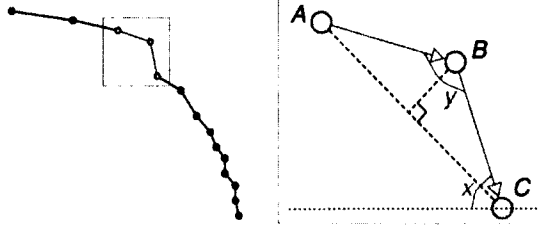


Figure 2.1: Illustration of angle-based metrics.

the  $i$ th mouse movement. Based on the record that belongs to each point-and-click action, we calculate angle-based metrics.

### Metrics

To analyze the mouse movement data, we define three fine-grained angle-based metrics: direction, angle of curvature, and curvature distance. These newly-defined metrics are different from the conventional metrics, such as speed and acceleration, and can accurately characterize a user's unique mouse moving behaviors, independent of its running platform.

- **Direction.** For any two consecutive recorded points B and C, we record the direction traveled along the line  $\overrightarrow{BC}$  from the first point to the second. The direction is defined as the angle between that line  $\overrightarrow{BC}$  and the horizontal (see angle  $x$  in Figure 2.1).
- **Angle of Curvature.** For any three consecutive recorded points A, B, and C, the angle of curvature is angle  $\angle ABC$ ; i.e., the angle between the line from A to B ( $\overrightarrow{AB}$ ) and the line from B to C ( $\overrightarrow{BC}$ ) (see angle  $y$  in Figure 2.1).
- **Curvature Distance.** For any three recorded points A, B, and C, consider the length of the line connecting A to C ( $\overrightarrow{AC}$ ). The curvature distance is the ratio between the perpendicular distance from point B to the line  $\overrightarrow{AC}$  (see the perpendicular lines in Figure 2.1) and the length of  $\overrightarrow{AC}$ . Note that this metric is unitless because it is the ratio of two distances.

As a comparison, we list the definition of two traditional mouse movement metrics, speed and pause-and-click, as follows.

- **Speed.** For each point-and-click action, we calculate the speed as the ratio of the total distance traveled for that action divided by the total time taken to complete the action.
- **Pause-and-Click.** For each point-and-click action, we measure the amount of time between the end of the movement and the click event. In other words, this metric measures the amount of time spent pausing between pointing to an object and actually clicking on it.

## **2.2.2 Mouse Movement Characterization**

### **Dependence on Different Platforms**

One problem we came across in analyzing our data is that it may be difficult or meaningless to compare two users who are using very different machines. The entire user's environment can affect its data: the OS used, screen size and resolution, font size, mouse pointer sensitivity, brand of mouse, and even the amount of space available on the desk near the mousepad. Metrics such as speed and acceleration, then, are poor choices for comparison between users of arbitrary platforms. This is because these two metrics can be skewed by differences in screen resolution and pointer sensitivity. On the other hand, metrics such as pause-and-click are highly dependent on the content a user is reading. For example, a user tends to pause longer before clicking a link on a rich content page such as a wiki article, and hesitates for a much shorter time before clicking a "submit" button.

This makes a good case to use angle-based metrics for arbitrary user comparison instead. Direction and angle of curvature are not based on screen size or any other element of the user's environment, and thus are relatively platform-

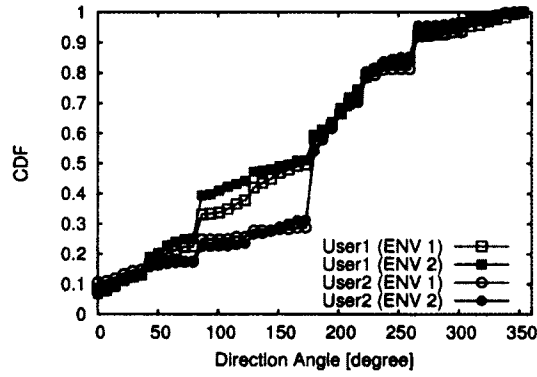


Figure 2.2: Direction Angle metric plotted for two different users on two different machines each.

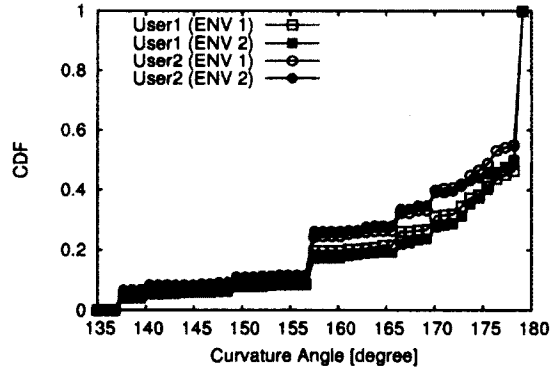


Figure 2.3: Angle of Curvature metric plotted for two different users on two different machines each.

independent. Likewise, curvature distance is a ratio of distances on the screen, and thus self-adjusts for the user's specific environment. A ratio can be compared to another user's ratio across platforms.

Figures 2.2, 2.3, and 2.4 show the comparison of two users with angle-based metrics. We can see that the cumulative distribution function (CDF) curves for the same user's individual data are very similar and well synchronized in shape, even across platforms. This indicates that angle-based metrics are relatively stable on different platforms.

### Uniqueness of Angle-Based Metrics Across Users

The other distinctive feature of angle-based metrics is that they are unique across users. Not only does the same user have very similar angle-based results on

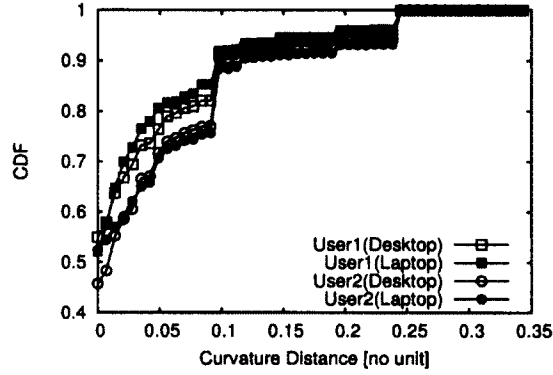


Figure 2.4: Curvature Distance metric plotted for two different users on two different machines each.

different platforms, but different users have clearly different angle-based results, even on similar platforms.

Again, as Figures 2.2, 2.3, and 2.4 show, even though each user's CDF is consistent across different platforms, there is a distinct gap between different users' CDF curves, even on the same platform. As a comparison, Figure 2.5 shows the CDF curves with respect to the speed of the two users, a more commonly-used metric. Figure 2.6 shows the CDF curves with respect to the pause-and-click of the two users. While the different users' CDF curves in both speed and pause-and-click are closely coupled on the same environment, there is a distinct gap between the same user's two curves for different environments. Since the closest matching curve for either user is the curve of the *other* user under the same environment, it can be very hard to uniquely differentiate people using these metrics.

Together with the platform independence discussed above, this makes angle-based metrics superior to speed and pause-and-click for user verification. Note that for easy presentation, we only compare the difference of the mouse dynamics between a pair of users. However, the similar observation holds for the other users.

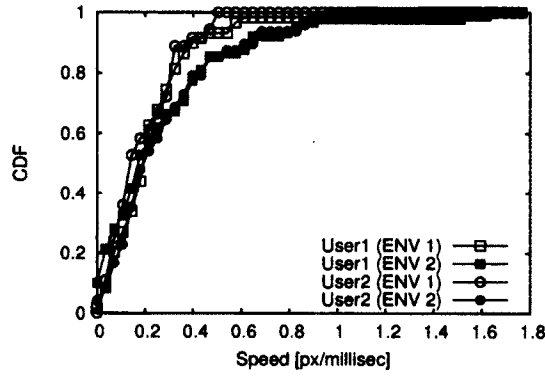


Figure 2.5: Speed metric plotted for two different users under two different environments each.

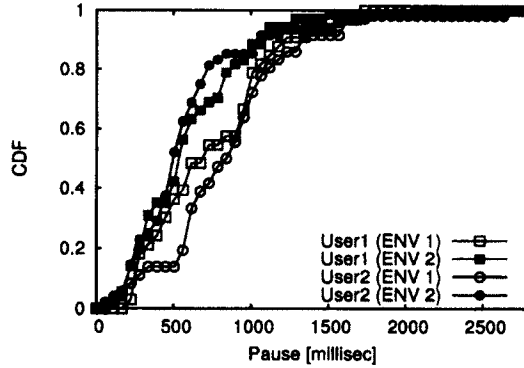


Figure 2.6: Pause-and-click metric plotted for two different users under two different environments each.

## Distance Between Distributions

Using the *distance* between two probability distributions, we further verify if the angle-based features of a user remain relatively stable across different types of mice, platforms, and time, in comparison with those of the other controllable users.

We define the distance between two probability distributions as follows. Since angle-based features are continuous variables, we divide their whole range into discrete intervals, called *bins*, and calculate the probability density functions (PDFs) regarding to each bin. Consider two distributions: the first is expressed as PDF  $\{p_1, p_2, \dots, p_n\}$ , where  $p_i$  represents the probability of falling into the  $i$ th bin; the second distribution is expressed similarly as PDF  $\{q_1, q_2, \dots, q_n\}$ . The *distance* be-



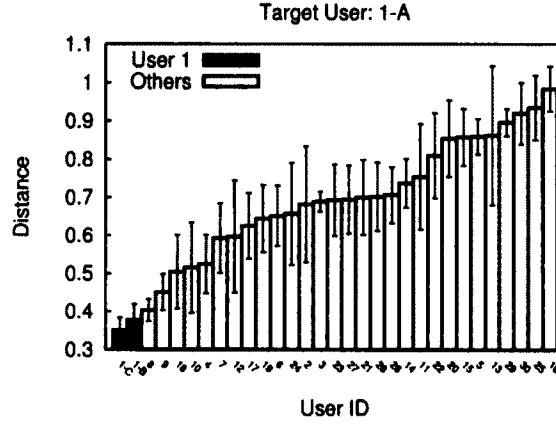


Figure 2.7: Distances from one user's curvature angle distribution to those of others, as well as to itself in different settings.

tween the two distributions is the accumulated deviation from each other over all bins:

$$D(p, q) = \sum_i |p_i - q_i|.$$

Of course, the distance here is dependent on the size of each interval. The smaller we divide the interval, the more subtle differences the distance reflects. However, the bin should not be so small as to enlarge noise.

Figure 3.5 plots the distances of a user from the other users, as well as from itself using a different mouse on a different machine at very different time. Each user's PDF is computed over 1,000 curvature angles randomly selected from its data, and loops for 10 times. The height of each bar is the average distance from the target user (labeled as user 1) in setting A (indexed by 1-A in the figure), and each error bar is standard deviation over the 10 times. Data 1-A, 1-B, and 1-C are all from user 1. The details of these three settings are listed in Table 2.1. Moreover, data 1-B are recorded two and half months later than data 1-A, and data 1-C are recorded two days later than data 1-B.

It is clear that the distances from user 1 to itself using different mice, at different machines, and over different times are the two smallest in the figure, i.e., both are smaller than the distances from user 1 to any other users with the same setting. This implies that the angle-based behavior of a user has its own inherent

Setting	Machine Type	Mouse Type
1-A	Dell Precision T3500	Dell MOC5UO Two-Button Scroll-Wheel
1-B	Apple Macbook MB990LL/A	Apple A1152 One-Button Trackball
1-C	Apple Macbook MB990LL/A	Dell MOC5UO Two-Button Scroll-Wheel

Table 2.1: Setting Details

pattern which is relatively stable across different settings and times; meanwhile, it is also distinguishable from the behaviors of other users. Note that in Figure 3.5, the distance values between user 1-A and some users are very close to each other (e.g. the distance value of user 23 and that of user 27). However, it does not indicate that the behaviors of those users are similar, because the definition of distance here is an accumulation of deviations at different bins. The dynamics of two users' PDFs could be totally different, but at the same time both deviate equally from a third user.

Note that to achieve accurate measurement results, there are two prerequisites in characterizing mouse movement under different environments. First, the polling rates of mouse recorders at different platforms should be configured to the same level. Second, prior to characterizing a user's mouse movement, sufficient mouse events must be collected to create a profile of the user's mouse movement. In particular, we observed that 1,000 mouse actions (which on average can be collected in 2 hours) are large enough to profile a user's mouse behavior well.

### **Number of Mouse Clicks in a Real Session**

For the field set, drawn from 1,074 real users on an online forum over the course of an hour, we recorded an average of 15.14 clicks per user session. Note that because this data was gathered over a one-hour window, this value is a lower

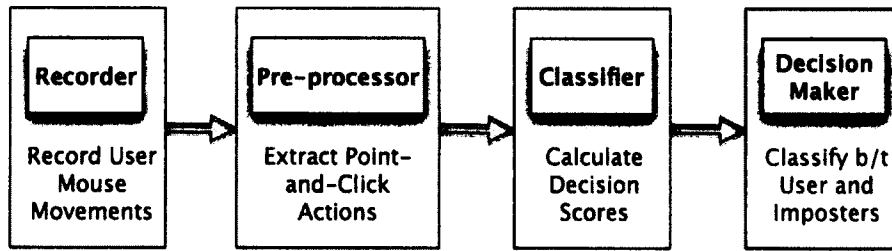


Figure 2.8: System Architecture

bound on the actual number of clicks in an average user session. Any user who was logged in before the window began or stayed logged in after the window ended (including users who stayed logged in longer than an hour) would necessarily have more clicks than recorded. Thus, the actual average is almost certainly higher, and probably much higher.

The number of mouse clicks per user session is closely related to verification time. With the average number of mouse clicks per session being about 15, a verification system based on mouse dynamics must be able to identify a user in fewer than 15 clicks in order to ensure that, on average, a decision is made within one user session. As shown in Section 2.4, our approach can verify a user's identity with high accuracy in only 15 clicks, so our system can reliably make the right decision before the user logs off in a majority of cases.

## 2.3 System Architecture

As shown in Figure 2.8, our proposed user verification system consists of four components—recorder, preprocessor, classifier, and decision maker.

The design of the first two components is straightforward. The main task of the recorder is to record users' mouse movements, while the preprocessor computes the angle-based metrics based on the recorded raw data. The focus of this section is on the design of the classifier and that of the decision maker.

### 2.3.1 SVM Classifier

We choose Support Vector Machines (SVMs) as our classifier to differentiate users based on their mouse movement dynamics. SVMs have been successfully used in resolving real-life classification problems, including handwritten digit recognition [104], object recognition [83], text classification [57], and image retrieval [102]. In general, SVMs are able to achieve comparable or even higher accuracy with a simpler and thus faster scheme than neural networks.

In the two-class formulation, the basic idea of SVMs is to map feature vectors to a high dimensional space and compute a hyperplane, which separates the training vectors from different classes and further maximizes this separation by making the margin as large as possible. SVMs classify data by determining a set of support vectors, which are members of the set of training inputs outlining a hyper plane in feature space [105].

For a binary classification problem, given  $l$  training samples  $\{\mathbf{x}_i, y_i\}, i = 1, \dots, l$ , each sample has  $d$  features, expressed as a  $d$ -dimensional vector  $\mathbf{x}_i$  ( $\mathbf{x}_i \in \mathbb{R}^d$ ), and a class label  $y_i$  with one of two values ( $y_i \in \{-1, 1\}$ ). A hyperplane in  $d$ -dimensional space can be expressed as  $\mathbf{w} \cdot \mathbf{x} + b = 0$ , where  $\mathbf{w}$  is a constant vector in  $d$  dimensions, and  $b$  is a scalar constant. We aim to find a hyperplane that not only separates the data points but also *maximizes* the separation. As Figure 2.9 shows, the distance between the dashed lines is called the *margin*. The vectors (points) that constrain the width of the margin are the *support vectors*. The formulation of our binary class SVM problem is to:

$$\begin{aligned} \text{minimize:} \quad & W(\alpha) = -\alpha^T \mathbf{1} + \frac{1}{2} \alpha^T H \alpha, \\ \text{such that:} \quad & \alpha^T \mathbf{y} = 0, \quad 0 \leq \alpha \leq C \mathbf{1}, \end{aligned}$$

where matrix  $(H)_{ij} = y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$ ,  $\alpha$  is the vector of  $l$  non-negative Lagrangian

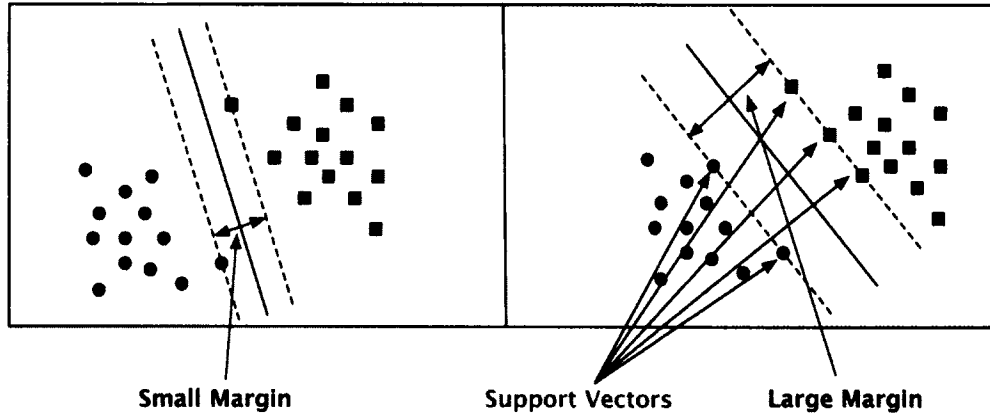


Figure 2.9: SVM hyperplane in two dimensional feature space [36].

multipliers to be determined, and  $C$  is a constant. This minimization problem is known as a *Quadratic Programming Problem (QP)*, which is well studied with many proposed efficient algorithms.

In reality, not all data points can be linearly separated as we assumed. To handle this issue, SVMs use a “kernel trick”. The data are pre-processed in such a way that the problem is transformed into a higher dimension, where they are linearly separable in the new feature space. Given a mapping  $z = \phi(\mathbf{x})$ , and defining the *kernel function* as  $K(\mathbf{x}_a, \mathbf{x}_b) = \phi(\mathbf{x}_a) \cdot \phi(\mathbf{x}_b)$ , our classifier would be

$$f(\mathbf{x}) = \text{sign} \left( \sum_i \alpha_i y_i (K(\mathbf{x}_i, \mathbf{x})) + b \right).$$

A popular choice of kernel function is the Gaussian Radial Basis Function (RBF)  $K(\mathbf{x}_a, \mathbf{x}_b) = \exp(-\gamma \|\mathbf{x}_a - \mathbf{x}_b\|^2)$ , where  $\gamma > 0$ , and is a tunable parameter. In practice, RBF is a reasonable first choice among other kernels, due to its generality and computational efficiency [18].

Thus, the procedure to resolve a classification problem using SVMs is: (1) choosing a kernel function, (2) setting the penalty parameter  $C$  and kernel parameters as well, if any, (3) resolving the quadratic programming problem, and (4) constructing the discriminant function from the support vectors. In particular, we view the user verification problem as a two-class classification problem, and

the learning task is to build a classifier based on the user mouse movements.

In our proof-of-concept implementation, we used the open source SVM package LIBSVM 3.0 [18] for building the prototype. LIBSVM is an integrated tool for support vector classification. We used the default RBF kernel and the cross-validation to find the best parameter  $C$  and  $\gamma$ . In our study, all impostors are classified as +1, and normal data are classified as -1. The detailed experiment setups will be discussed in Section 2.4.

### 2.3.2 Decision Making

In the design of the decision maker, we use two mechanisms, threshold and majority vote, to further improve verification accuracy.

#### Threshold

The threshold determines how SVMs' output is interpreted: a value over the threshold indicates an impostor, while a value under the threshold indicates a true user. To make a user verification system deployable in practice, minimizing the probability of rejecting a true user is sometimes even more important than lowering the probability of accepting an impostor.

By default, in a binary classification problem with labels in  $\{+1, -1\}$ , LIBSVM outputs a score called a *decision value* for each testing sample. If the decision value is greater than 0, the sample is classified as +1, otherwise it is classified as -1.

#### Majority Votes

To build the profile for an authorized user, in training, we randomly pick  $m/2$  samples that belong to the user, labeled as negative (non-impostor), and another random  $m/2$  from the field set, labeled as positive (impostor). We employ a simple *majority vote* decision making scheme in order to improve and stabilize the

verification accuracy. Specifically, before verifying if a sample belongs to the target user, we train the user's profile  $2n + 1$  times. Each time the training samples are different since they are randomly selected. In this way, there will be  $2n + 1$  votes about the predicted label for each testing sample. The label that is voted by the majority, i.e., with greater than  $n$  votes, will be the final predicted label. With majority votes, the decision maker can significantly reduce the randomness of the results and improve verification accuracy.

## 2.4 Experimental Evaluation

In this section, we evaluate the effectiveness of our mouse movement based verification system through a series of experiments, in terms of verification accuracy, verification time, and system overhead. The verification accuracy of our system is measured using (1) the false reject rate (FRR), which is the probability that a user is wrongly identified as an impostor, and (2) the false accept rate (FAR), which is the probability that an impostor is incorrectly identified as a user. Here we define a block as follows:

**Block** (or detecting block) A *block* is composed of mouse movements in a group of point-and-click actions. Statistical features are calculated based on all mouse movements in one *block*.

Note that choosing different sizes (that is, choosing different number of point-and-click actions contained) of a *block* greatly affects verification accuracy, in terms of FAR and FRR.

The verification time is the mean time needed to detect an identity mismatch. This corresponds to the sum of the time for the user to generate mouse actions needed to make a decision, the time to extract the features for this session, and the time to classify the identity. In our approach, the number of mouse actions needed to make a decision equals to the number of clicks contained in one *block*.

As described before, a *block* corresponds to one sample in either training or testing. Verification time is determined by the total number of actions needed to make decisions and the average time cost per action. In general, the larger the number of actions required for decisions and the higher the average time cost per action, the longer the verification time becomes.

### 2.4.1 Experimental Setup

As described in Section 2.2.1, our experiments are based on two sets of data. The first data set is collected in controllable environments. A total of 81,218 point-and-click actions are captured, with an average 5,801 point-and-click actions per user. Overall, 150 hours of raw mouse data are collected. The second data set is recorded from 1,074 anonymous users in an online forum for one hour.

These two data sets serve different purposes. A *target user* is selected from the first data set as the user to be verified, while forum users from the second data set are used as the background. Whereas we can identify a forum user based on its unique login name, the lack of guarantee on its collected data makes it unsuitable to be the verified target. The preprocessor extracts each user's point-and-click actions and computes the angle-based metrics corresponding to each point-and-click. Each of those generated files containing point-and-click actions is divided into two halves. The training data will be extracted from the first half, while the testing data will be from the second half. Therefore, there is no overlap between the training data and the testing data.

### 2.4.2 Verification Results

We construct our classification model based on self and non-self discrimination. That is, for a given user, its profile is learned from a certain number of its own mouse movement samples and an equal number of others' mouse movement samples. Therefore, the training data is composed of positive samples and an



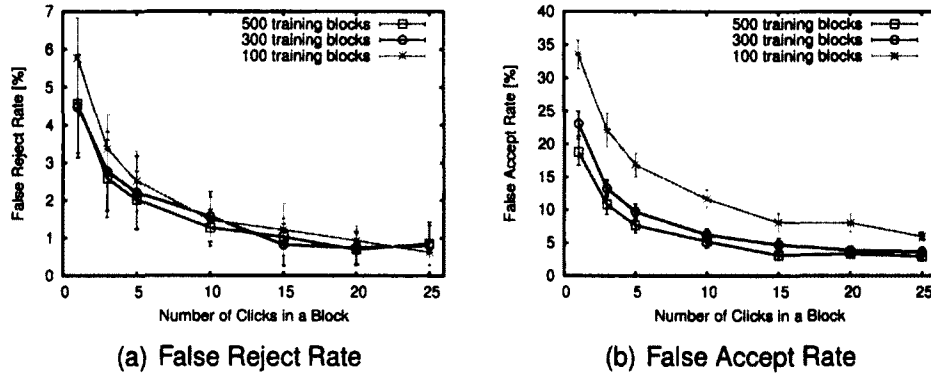


Figure 2.10: Variation of FRR and FAR with the number of clicks. Error bars indicate standard deviation.

equal number of negative samples. We train a separate model for each user in the controllable set. In the training file for a given user, a negative case is a block of point-and-click actions that belongs to itself, while a positive case is a block of clicks that belongs to *others*. Here, *others*' mouse movement blocks are randomly chosen from the forum set, due to its large user population.

There are four configurable parameters in our system: the size of a detecting block, the size of the training data, the threshold, and the number of votes. The first two are associated with the SVM training process. Increasing the threshold value directly lowers false reject rate (FRR), but at the cost of raising false accept rate (FAR). Increasing the number of votes improves verification accuracy in terms of both FRR and FAR, but increases the verification time.

To fully evaluate verification accuracy, we conduct two sets of experiments. In the first set of experiments, we test our classification model trained in the same environment. In the second set of experiments, we test the classification model trained in a different environment.

### Self and Non-Self Discrimination in Same Environment

We first configure the number of mouse clicks per block and the size of the training data. The FRR and FAR with different sizes of detecting blocks and training data

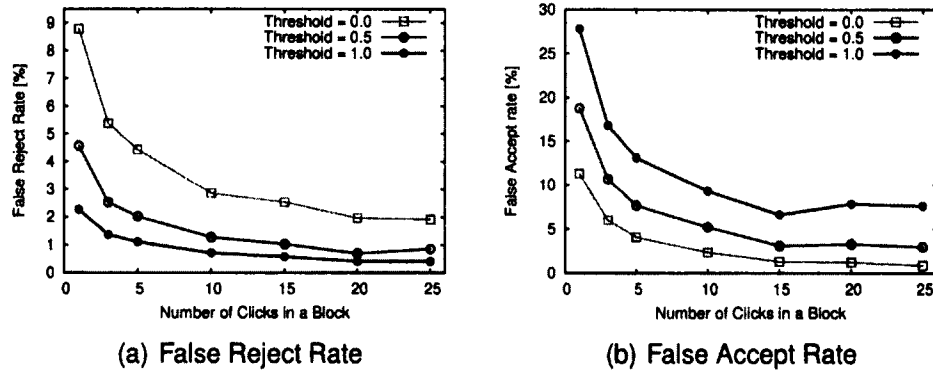


Figure 2.11: Variation of FRR and FAR with threshold.

are shown in Figures 2.10(a) and 2.10(b), respectively. These tests are performed with the default threshold of 0.5 and 5 out of 9 (5/9) majority votes. As larger detecting block size and training data are provided, the SVM classifier becomes more accurate, but we see diminishing returns in accuracy as the number of actions increases, e.g., going from 10 clicks to 25 clicks requires 150% more input but only provides a relatively small increase in verification accuracy; also going from 100 training samples to 300 training samples requires 200% more data, but only returns a relatively small increase in verification accuracy.

With the SVM classifier configured, the last two parameters, the threshold and the number of votes, determine the overall performance of the system.

The threshold can be increased or decreased from the default value of 0.0 to bias the classifier towards authentic users or impostors, lowering the FRR or FAR, respectively. As mentioned in Section 2.3.2, this is a tradeoff between user inconvenience level and system security level. After multiple tests, we observe that setting the threshold value to 0.5 yields a false reject rate 1% on average. Therefore, throughout this chapter, we only show results with a threshold value of 0.5. Setting the threshold value affects both FRR and FAR. Figure 2.11 shows that increasing the threshold value greatly lowers FRR at different sizes of a detecting block.

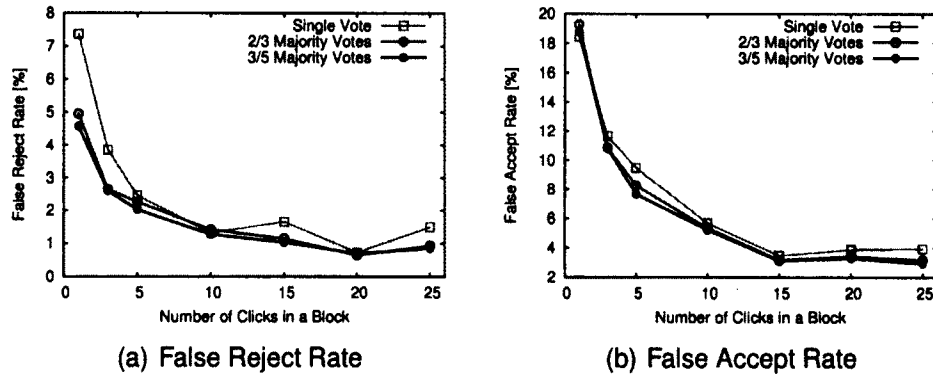


Figure 2.12: Variation of FRR and FAR with majority votes.

The use of majority votes increases the verification accuracy of the system, in terms of both FRR and FAR. Figure 2.12 shows the improvement of FRR by majority votes. However, increasing the number of votes means longer verification time, since for  $n$  votes the classifier needs to be run  $n$  times. Figure 2.12 indicates that the improvement by 2/3 majority votes is comparable to that of 3/5 majority votes.

With a fully configured system (500 training blocks, a threshold of 0.5, and 3/5 majority votes), Table 2.2 lists FRR and FAR averaged over 30 users in the controllable set. It can be seen that, if there are 25 clicks in one *block*, the average false reject rate is 0.86%, so there is only a little chance that an authenticated user is misclassified as an impostor. Meanwhile, we achieve an average false accept rate of 2.96%.

Number of Clicks	FRR	FAR
1	4.57%	18.79%
3	2.59%	10.81%
5	2.02%	7.67%
10	1.27%	5.23%
15	1.03%	3.13%
20	0.70%	3.32%
25	0.86%	2.96%

Table 2.2: Variation of FRR and FAR with Different Number of Clicks in One Block

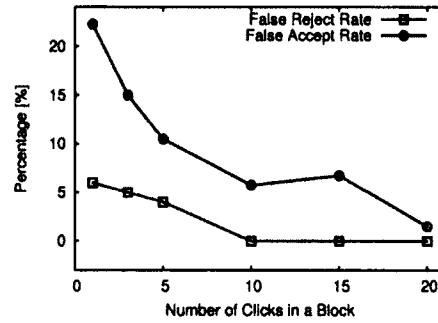


Figure 2.13: FRR and FAR for one user profiled on one platform and tested on another platform.

## Self and Non-Self Discrimination in Different Environment

To verify that our approach also works in different environments with different machines, we conduct another self vs. non-self discrimination experiment on two different machines. More specifically, the user's profile is trained from its mouse movements in a work environment on a desktop, while its mouse movements in a home environment on a laptop are tested. The testing user base includes 5 users. The corresponding FRR and FAR are shown in Figure 2.13, each represented by a single curve, respectively. Note that a user is profiled on one platform but tested on a different one, and hence a single plot shows the result of the test.

It can be seen that our approach works well across different environments and platforms. It further confirms that our classification model indeed captures those features that are intrinsic to a user and not affected by environmental factors.

## Partial Movements

*Partial movements* are a series of continuous mouse movements *without* ending in a click. On one hand, unlike point-and-click actions that have a certain object to reach as a target, some partial movements could be aimless. For example, a user may move its mouse just to stop the screen saver when watching a video. Thus, we observe that the movements of point-and-clicks demonstrate a more consistent pattern than those of partial movements. However, most partial movements are intentionally performed. For example, a user may often move its mouse to

aid reading. In addition, some partial movements are just as well-motivated as point-and-clicks. A user could start moving the mouse to a link, but then decide not to click on it.

Moreover, in a real user session, partial movements occur much more frequently than point-and-clicks. From the forum data we collected, there are only 0.53 mouse clicks per minute on average, but 6.58 partial mouse movements per minute. Figure 2.14 shows the comparison of ROC (Receiver Operating Characteristic) curves with and without partial movements for a randomly selected user (other users' ROC curves are similar).

Suppose we choose 20 mouse clicks in a detecting block. On one hand, without partial movements – that is, when only point-and-clicks are included – the EER (equal error rate) is 1.3%; with partial movements, the EER increases to 1.9%. On the other hand, using partial movements can lower the average verification time by one order of magnitude (about 12 times) in our experiments. Therefore, using partial movements will significantly reduce verification time, but at the cost of accuracy degradation.

### **Subtleties on Verification Time**

The verification time is the time required by a verification system to collect sufficient behavioral data and then make a classification decision. The value of the verification time heavily depends on two factors: (1) the number of required mouse clicks (or mouse movements if partial movements are included) in a detecting block, and (2) how frequently a user generates mouse actions. If the number of mouse actions in a detecting block is already configured, the verification time will be mainly determined by the latter.

There are two verification scenarios, static and continuous, when estimating the number of mouse actions a user generates per unit time. In the scenario of *static* verification, a user is required to perform a series of mouse movements

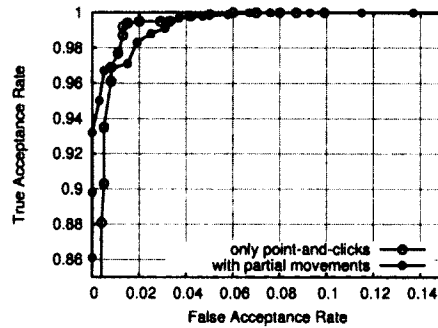


Figure 2.14: ROC curves with and without partial movements.

and its mouse data is verified within a certain amount of time (e.g., login time). A good example of this scenario is a click-based graphical password for user login, where five clicks are estimated to be made in no more than 25 seconds [10, 22]. This implies that the verification time will be less than 100 seconds if 20 mouse clicks are needed in a testing block. By contrast, in the scenario of *continuous* verification, a user's mouse data is continuously collected and verified throughout the entire session. This is non-intrusive to users and meets the goal of passive monitoring. However, the frequency of user mouse actions varies significantly in different sessions. In general, the average frequency of user mouse actions will be much lower than that of the static scenario. The reason is that often there is a period of silence between a user's previous and next mouse actions while a user is reading or typing. An observation from the forum data we collected indicates, on average, it takes 1.89 minutes for one mouse click to happen. If we choose 20 mouse clicks in a detecting block, the verification time could be as long as 37.73 minutes; however, the verification time will be reduced to 3.03 minutes if partial movements are used.

### System Overhead

The verification system can be deployed in two different scenarios. In the first scenario, for example, it can be used for access control in a computer lab or on a personal computer. In this case, it will be installed at the client side by the system

administrator. In this standalone scenario, normally only a single user is present for verification at any one time, and the end host has a plenty of resources to fulfill the verification tasks. Thus, the performance impact caused by the verification on the host is minor.

In the second scenario, for an online application such as banking account verification, the system will be placed at the server side, with JavaScript embedded inside users' account home page. Deployed at the server side, our system needs to be able verify hundreds or even thousands of users simultaneously in real-time. Thus, system overhead becomes an issue, and the system must be efficient in terms of memory and CPU costs.

We first estimate the memory overhead of the verification process. We profile the verification process using the Linux tool `valgrind`, and find out that it only consumes 3.915 KBytes of memory per testing block during the operation. The primary memory cost is to accommodate the accumulated user-input actions and SVM outputs for each online user. A single user-input action consumes 12 bytes, 4 bytes each for the 3 angle-based metrics. A detecting block of 10 user-input actions consumes 120 bytes, and this is the per-user memory requirement. If 120 bytes is scaled to 1,000 online users, it is only 117.19KBytes in total, which is negligible considering that online websites currently store the user name, password, IP address, security questions and literally dozens of other attributes for each user.

The computational overhead is the sum of CPU costs in pre-processing and detecting (including classifying and decision making). The pre-processing on 15 minutes' user inputs from more than 1,000 users in a typical website, including 5,270 point-and-click actions, takes only 20.937 seconds. The CPU cost is measured on a Pentium 4 Xeon 3.0Ghz, using the Linux command `time`. Note that the forum trace is collected during 15 minutes from about 1,000 online users, implying that it takes about 23.3 milliseconds to process data generated in one second.

At the same time, the verification takes only 229 milliseconds over 5,801 point-and-click actions. In comparison to pre-processing, this is negligible. Therefore, the induced computational overhead is minor on the server.

In terms of disk space for storing user profiles, the signature of a single user profile generated by the training process consumes 203.32KBytes. If it is scaled to 100,000 users, that is 19.4GBytes, which is very affordable at a personal computer, let alone a high end server.

### 2.4.3 Classification of Pointing Devices

In reality, it is common that a same user uses different kinds of pointing devices from time to time. For example, a laptop user may use the on-board touchpad for some time, but switch to a USB-connected mouse when a larger desk space is available. Therefore, it is desirable to detect the type of pointing device being used in a non-obtrusive way. In other words, the question we attempt to answer is, given a series of cursor movements, is it possible to tell if they are from a touchpad or a computer mouse? Intuitively, there exist some biomechanical differences between a touchpad and a mouse because their driven forces are different [32, 85]. While a touchpad involves more finger movements, a mouse involves more wrist movements. In addition, when people use a mouse, they are more likely to move their whole arm. Fitts' law analysis shows [70] that the index of performance for the touchpad is between 1.6 to 2.3, whereas the computer mouse has the values ranging from 2.6 to 10.4. Thus, generally it takes longer time to move the cursor with a touchpad than with a computer mouse. All these factors contribute to a detectable difference between the moving behavioral patterns using a touchpad and a mouse.

Here we introduce an additional angle-related metrics, *moving orientation*, in which a movement direction falls into one of the eight sections shown in Figure 2.15. We further define a set of new features by associating moving orien-



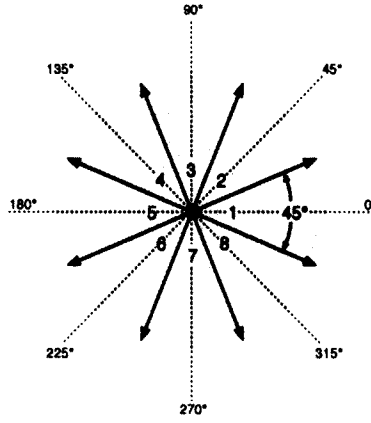


Figure 2.15: The entire angle range  $360^\circ$  is divided into eight sections. The eight divisions of angle, each with a resolution of  $45^\circ$ , are labeled with numbers 1 to 8. For each section, the average values of metrics are calculated as features, in order to differentiate between mouse and touchpad data.

tation with speed and curvature angles. In particular, we average all the corresponding metric values like speed per orientation section. As a result, for a given metric, there are eight average values with respect to the eight different moving orientations in total. These additional features are described as follows.

- **Normalized Step-wise Speed per Moving Orientation.** In Figure 2.1, a step-wise speed is calculated as the length of vector  $\overrightarrow{AB}$  over the time difference between  $A$  and  $B$ . It is further normalized with the averaged step-wise speed over the whole continuous movement. Our purpose is to quantify the distribution of moving speed with respect to different moving orientations.
- **Averaged Curvature Angle per Moving Orientation.** The choice of this feature is due to the underlying differences in using touchpad and mouse — they involve different bodily parts for movements: the former is mostly with finger, while the latter is with both wrist and finger. These two different moving mechanisms make their kinetic traits deviate from one orientation to another. For this very reason, exploring these angle-related dynamics suits our need in differentiating mouse and touchpad data.
- **Change of Adjacent Curvature Angles per Moving Orientation.** The variation of curvature angle is closely related to angular acceleration. From phys-

ical science, angular acceleration is proportional to *moment of force* (also known as *torque*,  $\tau = r \times F$ ). And again, touchpad and mouse usages involve different bodily parts, which affects the magnitude and direction of forces exerted on the pointing devices, and in turn the angular acceleration is affected. Overall, this feature depicts one more aspect of differences in touchpad and mouse behaviors.

We differentiate movement behaviors between a traditional mouse and a touchpad by employing these new metrics and features, in addition to the three metrics introduced in Section 2.2.1. In accuracy evaluation, we collected a data set from 21 users using either mouse or touchpad. Figure 2.16 plots the results of classification accuracy based on the above feature set. The curve shows the accuracy as the function of the number of mouse events per block. On average, there are about 10 mouse events in one mouse click action. Thus as shown in Figure 2.16, we can achieve more than 90% accuracy in differentiating mouse and touchpad after 400 events or about 40 point-and-clicks. Generally the more mouse events are available in calculation of features, the higher accuracy we achieve. Some small fluctuations present in the curve, and it is not always monotonically increasing with more mouse events per block. This is because we use statistics for computing the features. There exists random noise, and hence more mouse events does not always make it closer to the true distribution.

#### **2.4.4 Classification within Controlled Dataset**

In order to cross validate our approach, we conduct an additional set of experiments only using the controlled dataset. Within the controlled dataset, half of the users act as the target users, and the other half serve as impostors only. We run one round of classification for each target user, where the rest of users in the controlled dataset (i.e., non-target-users) are treated as the background. With a block size of 20 point-and-clicks and the training size of 200 blocks, the average

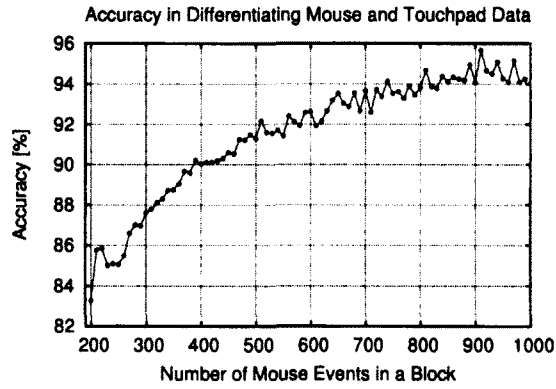


Figure 2.16: The classification accuracy in differentiating mouse and touchpad data, as a function of the increasing number of mouse events per block.

Source	FRR	FAR	Data required	Settings	Notes
[3]	2.4649%	2.4614%	2000 mouse actions	Continuous	Free mouse movements
[78]	0%	0.36%	2000 mouse actions	Continuous	Free mouse movements
[45]	2%	2%	50 mouse strokes	Static	Mouse movements from a memory game
[88]	1.75%	0.43%	Not specified	Continuous	Applies to a certain application
[93]	11.2%	11.2%	3600 mouse curves	Continuous	Free mouse movements
Ours	1.3%	1.3%	20 mouse clicks	Continuous	Free mouse movements

Table 2.3: Comparison with Existing Works

accuracy turns out to be 92.12%. The accuracy slightly declines in comparison to that with the field dataset. This is because the field dataset contains a much larger pool of mouse actions from non-target-users than the controlled dataset. As a result, with the field dataset, the impostor model learned by the classifier captures a richer profile of non-target-user behaviors. And in the context of two-class classification, the boundary between a target user and impostors in the feature space can be more accurately recognized, which yields higher accuracy when utilizing the field dataset rather than the controlled dataset.

## 2.4.5 Comparison with Existing Works

We compare our evaluation results with those of existing works in terms of verification accuracy and time, which are listed in Table 2.3. As described in Section 2.4.2, the verification time is highly dependent on the number of mouse events

needed to make a decision, the type of mouse events used (mouse click, mouse move, or drag-and-drop), as well as how fast a user generates mouse events. Even for the same user at different times, the number of mouse events per unit time varies a lot. However, to the best of our knowledge, our work is the first to achieve high accuracy with a reasonably small number of mouse events.

## **2.5 Discussion**

Since our verification system records users' mouse movements and clicks, privacy concerns may arise. However, compared to keystroke dynamics, the amount of personal information included in mouse dynamics is minimal. In the process of recording keystrokes, the system would record the user's passwords, user names, and other sensitive textual information. By contrast, recording mouse dynamics only reveals the physical movements of a mouse and its clicks within a certain period of time, giving away little to no information about user credentials. Even with the perfect knowledge of a user's mouse movements, the only things an adversary can figure out are when the user clicked and on which position of the screen. Thus, we believe that our verification system will not cause any privacy violations.

In general, mouse-dynamics-based re-authentication techniques are robust against online forgery. A person's unique mouse dynamics are similar to its signature, and like a signature, it is difficult to mimic even with the complete knowledge of the original. In fact, a user's mouse dynamics is a continuous process, making it much harder to forge than a signature. Unlike forging a signature, which only has to be accomplished once, the adversary of our verification system would need to mimic the true user's mouse patterns continuously for the entire length of the session. It is extremely difficult for one user to force itself to consistently move the mouse in such a mechanical way that it matches specific angles, even

if those metrics are known ahead of time. Thus, mouse dynamics generally and our fine-grained angle-based approach in particular, are very robust against on-line forgery. However, how vulnerable mouse-dynamics-based approaches are to offline attacks, especially generative attacks which create concatenative synthetic forgeries [8], is still an open question and will be investigated in our future work.

The retraining of our classifier is necessary to deal with sudden, perhaps temporary, changes in a user's mouse profile. If the user's behavior suddenly changes, due to an unexpected complication such as a sprained wrist, the difference in mouse usage could be large enough for the user to be unrecognizable by the verification system. The system would classify the user as an impostor and prevent that user from accessing its own account. While these sorts of occurrences are relatively rare, to avoid the possible rejection, the user can easily appeal to a system administrator to retrain the classifier with the user's new movement patterns. Once the user's behavior returns to normal (for example, the user's wrist heals), we can either retrain the system again, or simply reuse the previous classifier if a backup is available.

Although our approach is relatively independent of the running environments, it is sensitive to the polling rate of mouse movement recording. In the mouse data collection, a continuous mouse movement is discretized to a set of mouse coordinates, which are sampled at a certain rate. Thus, the measured resolution of mouse movements is dependent on the polling rate of a recorder. The faster the polling rate is, the more fine-grained movements we capture. For example, given a mouse movement curve, a high polling rate can render a smooth accurate shape, but a low polling rate more likely profiles it as a zigzag path. For this reason, in our data collection on different environments, the polling rates of the recorders are configured to the same value. In fact, it is not difficult to maintain a given polling rate under different running environments. Through I/O methods provided in most common programming languages, we are able to set timers and

capture mouse cursor position at a fixed interval.

It is true that with an increase in user population, there is a higher chance that two users share the similar mouse movements. In fact, known as “the scalability problem”, this is a common problem for almost all biometrics approaches. In face recognition, if more people are tested, it is more likely that two users’ faces are similar and could make the classifier fail. The same thing happens in keystroke dynamics, and it has been determined that the accuracy of keystroke dynamics decreases with the increase in sample size [13, 84].

Though promising, our accuracy is unable to reach the European Standard for Access Control Systems, which requires a false acceptance rate (FAR) of under 0.001% and a false rejection rate (FRR) of under 1%. Therefore, we believe that our scheme is more suitable to work together with other authentication methods for user verification, instead of working as a stand-alone authentication system.

## **2.6 Conclusion**

In this chapter, we present a new approach to user re-authentication using the behavioral biometrics provided by mouse dynamics. Our approach focuses on fine-grained angle-based metrics, which have two advantages over previously studied metrics. First, angle-based metrics can distinguish a user accurately with very few mouse clicks. Second, angle-based metrics are relatively independent of the operating environment of a user, making them suitable for online re-authentication.

Our system mainly consists of a recorder, which gathers a user’s mouse dynamics, and a support vector machine (SVM) classifier, which seeks to verify a user as either an impostor or an authenticated party. We gathered two sets of data: one set of 30 users under controlled circumstances, and another set of over 1,000 users on a forum website. We evaluated the system performance in

terms of verification accuracy and time, resulting in a equal error rate (EER) of 1.3% with just 20 mouse clicks. We also showed that, for a system deployed at server side, the overhead required for online verification is negligible.

# **3 User Verification on Smartphones via Tapping Behaviors**

This chapter is on our second work in developing another behavior-based user verification system on smartphones. Exploiting a variety of on-board sensors (accelerometer, gyroscope, and touchscreen sensors) readily available on smartphones, we establish user-specific tapping patterns that can distinguish him/her from others. This chapter is structured as follows. Section 3.1 and 3.2 reviews the background and related work in the area of smartphone user authentication. Section 3.3 describes our data collection and measurement, including our choice of metrics. Section 3.4 details the proposed classifier for user verification. Section 3.5 presents our experimental design and results. Section 3.6 discusses issues which arise from the details of our approach, and Section 3.7 concludes.

## **3.1 Background**

The tapping behaviors of individual users on touchscreen vary from person to person due to differences in hand geometry and finger agility. Each user has a unique personal tapping pattern, reflected on the different rhythm, strength, and angle preferences of the applied force. As our tapping-behavior-based approach verifies the owner of a smartphone based on “who you are” – your physical and behavioral traits, instead of “what you know”, it belongs to biometrics-based



user authentication. In general, a biometrics authentication system authenticates users either by their physiological traits like faces and voices [11,72] or behavioral patterns like finger typing and hand movements [77, 116].

While physiological traits can achieve high accuracy in the process of user authentication, they have not been widely used in mobile devices. Recent studies have also shown that the physiology-based mechanisms deployed in mobile devices are sensitive to certain environmental factors, which could significantly diminish their accuracy and reliability. For example, face recognition may fail due to a different viewing angle and poor illumination [86], and voice recognition degrades due to background noise [11]. However, given the same mobile device, behavioral biometrics tend to be less sensitive to the surrounding environmental factors like darkness or noise.

Exploiting the behavioral information captured by multiple sensors on a smartphone, we can exclusively create a detailed user profile for verifying the owner of the smartphone. Since our approach works seamlessly with the existing passcode-based user authentication mechanisms in mobile devices, it plays a role of *implicit authentication*. In other words, our approach can act as a second factor authentication method and supplement the passcode systems for stronger authentication in a cost-effective and user-transparent manner. More recently, seminal works have been proposed to explore the feasibility of user verification employing the behaviors of pattern-based passwords [30]. However, the false reject rate (FRR) of their work is rather high, which means there is a high chance that the owner of a mobile device would be mistakenly regarded as an impostor and be blocked from accessing the device.

## 3.2 Related Work

### A) Keystroke Dynamics and Graphical Passwords.

Keystroke dynamics has been extensively studied in distinguishing users by the way they type their personal identification number (PIN) based passwords [65]. Research done on the analysis of keystroke dynamics for identifying users as they type on a mobile phone can be found in [9, 25, 26, 60, 79, 112]. Clarke *et al.* [26] considered the dynamics of typing 4-digit PIN codes, in which the researchers achieve an average Equal Error Rate (ERR) of 8.5% on physical keyboard on a Nokia 5110 handset. Zahid *et al.* [112] examined this approach on touchscreen keyboards and achieve, in one best scenario, a low Equal Error Rate of approximately 2% with training set required a minimum of 250 keystrokes.

Researchers have also suggested the use of graphical passwords as an easier alternative to text-based passwords [19, 56], based on the idea that people have a better ability to recall images than texts. A good overview of popular graphical password schemes has been reported in [10]. Chang *et al.* [19] proposed a graphics-based password KDA system for touchscreen handheld mobile devices. The experiment results show that EER is 12.2% in the graphics-based password KDA proposed system, and EER is reduced to 6.9% when the pressure feature is used in the proposed system. Different usability studies have outlined the advantages of graphical passwords, such as their reasonable login and creation times, acceptable error rates, good general perception and reduced interference compared to text passwords, but also their vulnerabilities [103].

### B) Inferring Tapped Information from On-board Motion Sensors.

Several independent researches have found that simply by using data acquired by smartphone motion sensors, it is sufficient to infer which part of the screen users tap on [14, 75, 82, 109]. The first effort was done by Cai *et al.* in 2011 [14].

PIN	Users	Actions	Average Actions Per User	Filtered-Out
3-2-4-4	53	1,751	33	0.80%
1-1-1-1	41	2,577	63	2.64%
5-5-5-5	42	2,756	66	3.70%
1-2-5-9-7-3-8-4	27	1,939	72	7.37%
1-2-5-9-8-4-1-6	25	2,039	82	4.76%

Table 3.1: Collected Data

They utilized features from device orientation data on an HTC Evo 4G smartphone, and correctly inferred more than 70% of the keys typed on a number-only soft keyboard. Very soon, Xu *et al.* further exploited more sensor capabilities on smartphones, including accelerometer, gyroscope, and orientation sensors [109]. Evaluation shows higher accuracies of greater than 90% for inferring an 8-digit password within 3 trials. Miluzzo *et al.* demonstrated another key inference method on soft keyboard of both smartphones and tablets [75]. 90% or higher accuracy is shown in identifying English letters on smartphones, and 80% on tablets. Owusu *et al.* [82] infers taps of keys and areas arranged in a 60-region grid, solely based on accelerometer readings on smartphones. Result showed that they are able to extract 6-character passwords in as few as 4.5 trials.

### **C) User Authentication by Their Behavior on Touch Screens.**

Research has been done in exploring different biometric approaches for providing an extra level of security for authenticating users into their mobile devices. Guerra-Casanova *et al.* [48] proposed a biometric technique based on the idea of authenticating a person on a mobile device by gesture recognition, and achieve Equal Error Rate (EER) between 2.01% and 4.82% on a 100-users base. Unobtrusive methods for authentication on mobile smart phones have emerged as an alternative to typed passwords, such as gait biometrics (achieving an EER of 20.1%) [31, 80], or the unique movement users perform when answering or

placing a phone call (EER being between 4.5% and 9.5%) [28].

Very recently De Luca *et al.* [30] introduced an implicit authentication approach that enhances password patterns on android phones, with an additional security layer, which is transparent to user. The application recorded all data available from the touchscreen: pressure (how hard the finger presses), size (area of the finger touching the screen), x and y coordinates, and time. Evaluation is based on 26 participants, with an average accuracy of 77%.

A latest work conducted by Sae-Bae *et al.* [91] makes use of multi-touch screen sensor on iPad (not phone) to capture the palm movement. They achieved a classification accuracy of over 90%. However, palm movements is not suitable for smartphone screens, since the screen is typically too small for palm movements. Citty *et al.* [24] presented an alternative approach to inputting PINs on small touchscreen devices. It uses a sequence of 4 partitions of a selection of 16 images, instead of 4-digits PINs, to increase the possible combination of authentication sequences. However, inputting the sequence needs extra efforts in memorizing the images sequences. Kim *et al.* [66] introduced and evaluated a number of novel tabletop authentication schemes that exploit the features of multi-touch interaction.

### **3.3 Measurement and Characterization**

Over 80 participants are involved in our data collection. Five different PINs are tested, in which three of them are 4-digit, and two are 8-digit. Here we choose PINs 3-2-4-4, 1-2-5-9-7-3-8-4, and 1-2-5-9-8-4-1-6 to represent these normal cases, but PINs 1-1-1-1 and 5-5-5-5 to represent the two extreme cases, one at the corner and the other at the center, respectively. Each participant is asked to enter an error-free PIN for at least 25 times and we collect a total of 11,062 error-free actions. The user's timing and motion data are recorded during the

process. In this chapter, we refer to an *action* (or user input action) as the process of tapping one PIN, instead of individual digits. The detailed information of the collected data is listed in Table 3.1.

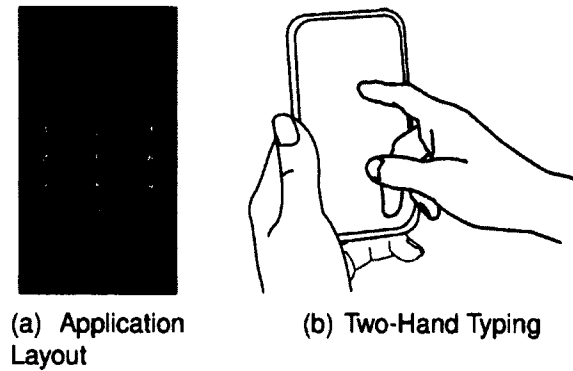


Figure 3.1: Screen layout of our data collection application, and the two-hand typing action.

The timing information is in resolution of milliseconds. Occasionally, some participants fail to make a smooth tapping intentionally or unintentionally. Therefore, we employ a simple outlier removal process to all the collected raw data. An outlier tapping action is often signaled by a markedly longer-than-usual time interval, especially for a user who is very familiar with its own PIN. In our data set, a *smooth* PIN tapping action takes at most 600 milliseconds between subsequent keys for all participants. As a result, an inter-key time of greater than one second always signals such an outlier behavior. By this standard, a small amount of raw data is filtered out, as listed in the right-most column of Table 3.1.

All the data are collected on a Samsung Galaxy Nexus. Its fastest sampling rate on motion sensor readings is about 100Hz. Figure 3.1(a) shows the layout of our Android application for the data collection. In the experiments, all the participants are asked to hold the phone with their left hands, and tap with their right hand index fingers, as shown in Figure 3.1(b).

We make use of the Android APIs to detect the touch event, including both key-press and key-release. Between each key-press and key-release, we record raw data of timestamps, acceleration, angular acceleration, touched-size, and

Feature Set	Description	# of Dimensions	
		4-digit	8-digit
<b>Acceleration</b> (linear & angular)	Acceleration at TouchDown	8	16
	Acceleration at TouchUp	8	16
	Minimum during key-hold time	8	16
	Maximum during key-hold time	8	16
	Average during key-hold time	8	16
<b>Pressure</b>	Pressure at TouchDown	4	8
	Pressure at TouchUp	4	8
<b>Size</b>	Touched size at TouchDown	4	8
	Touched size at TouchUp	4	8
<b>Time</b>	Key hold time	4	8
	Inter-key time	3	7
<b>Total</b>	All features	63	127

Table 3.2: Features of Touchscreen Tapping Behaviors

pressure. Acceleration and angular acceleration are from API `SensorEvent`, while touched-size and pressure are from API `MotionEvent`.

### 3.3.1 Feature Extraction

Based on the raw data, we compute four sets of features for each PIN typing action: acceleration, pressure, size, and time. We describe each of them in the following:

- **Acceleration:** For each digit  $d$  in a PIN action, we calculate the five acceleration values:
  - $A_{d,1}$ : the magnitude of acceleration when the digit  $d$  is pressed down;
  - $A_{d,2}$ : the magnitude of acceleration when the digit  $d$  is released;
  - $A_{d,3}$ : the maximum value of magnitude of acceleration during digit  $d$  key-press to key-release;

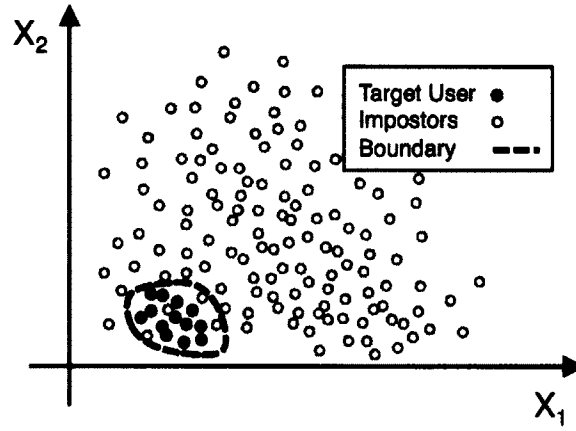


Figure 3.2: An illustration of two-feature space of a target user and many others.  $X_1$  and  $X_2$  are the two features. The dashed lines define the boundary of the target user's behavior. Because the target user's behavior is limited to a concentrated area, the boundary blocks the majority of potential impostors.

- $A_{d,4}$ : the minimum value of magnitude of acceleration during digit  $d$  key-press to key-release;
- $A_{d,5}$ : the average value of magnitude of acceleration during digit  $d$  key-press to key-release.

All above values are the magnitude of acceleration  $\|\vec{a}\| = \sqrt{a_x^2 + a_y^2 + a_z^2}$ .

We choose not to use individual components, because the phone coordinate system is sensitive to location change. A similar procedure is applied to calculate the features from angular accelerations. Combining both acceleration- and angular-acceleration-related features, there are total of 40 in a 4-digit PIN action and 80 in an 8-digit PIN action.

- **Pressure:** We obtain the pressure readings through Android API `MotionEvent.getPressure`.

The returned pressure measurements are of an abstract unit, ranging from 0 (no pressure at all) to 1 (normal pressure), however the values higher than 1 could occur depending on the calibration of the input device (according to Android API documents). In the feature set, we include pressure readings at both key-press and key-release. There are 8 pressure-related features for a 4-digit PIN, and 16 for an 8-digit PIN.

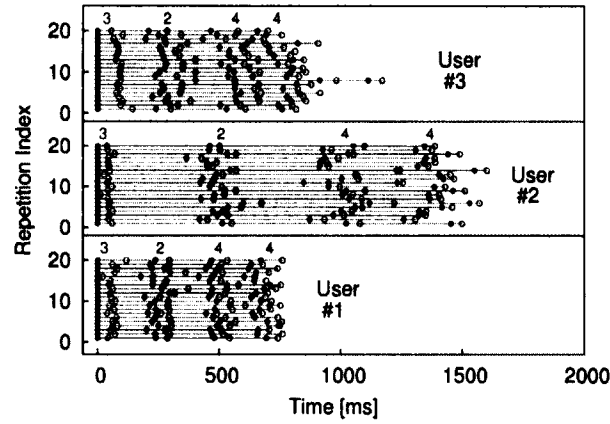


Figure 3.3: Timing of tapping on the smartphone from three different users, shown in three vertical panels. Each user typed 20 times of the number string "3244". The solid dots represent key-press time, and the open dots are key-release time. Different colors represent the timestamps of different digits.

- Size:** Similar to pressure readings, another Android API call `MotionEvent.getSize()` measures the touched size, associated with each touch event. According to Android document, it returns a scaled value of the approximate size for the given pointer index. This represents the approximation of the screen area being pressed. The actual value in pixels corresponding to the touch is normalized with the device's specific range and is scaled to a value between 0 and 1. For each key-press and key-release, we record the size readings and include in the feature set. A 4-digit PIN contains 8 size-related features, and an 8-digit PIN contains 16.
- Time:** key-hold times and inter-key time intervals between two nearby keys. They are measured from the `TouchEvent` timestamps, of both `TouchUps` and `TouchDowns`. Overall, a 4-digit PIN action contains 7 time-related features, while 8-digit PIN contains 15.

For a 4-digit PIN, each action results in a total of 63 features; for an 8-digit PIN, the number of features for one action is 127. Table 3.2 summarizes the description of the above four feature sets.



### **3.3.2 Touchscreen Tapping Characterization**

Our underlying assumption is that a user's feature distribution should be clustered within a reliably small range compared with many others. As a result, those metrics can be exploited to block the majority of impostors, as illustrated in Figure 3.2.

#### **Uniqueness of User Pattern**

As described above, we define four sets of features in order to characterize a user's tapping behaviors on smartphones: acceleration (both linear and angular), pressure, size, and time. All these features can be easily obtained from a smartphone's on-board sensors, and can accurately characterize a user's unique tapping behaviors. Based on the feature data, we observe that each user demonstrates consistent and unique tapping behaviors, which can be utilized for differentiating itself from other users.

Figure 3.3 shows the timestamps of entering the same PIN 3-2-4-4 from three different users, including the moments of each key-press and key-release. Each individual's timing patterns clearly differ, but are very consistent within themselves. This is similar to the observations on a regular computer keyboard [73].

In addition to timing information, motion data such as pressure, touched size, and acceleration also reveal user-specific patterns. Generally speaking, acceleration is proportional to the tapping force applied to the touchscreen, while angular acceleration represents the moment of force. Touched size is related to both user finger size and tapping force. Figure 3.4 shows the tapping pressure from three different users. We can see that three different users' tapping pressure form distinguishable individual patterns, with Subject #1 taps the hardest, Subject #2 taps much more gently, and Subject #3 is gentlest. Meanwhile, the level of tapping pressure is relatively consistent within one subject.

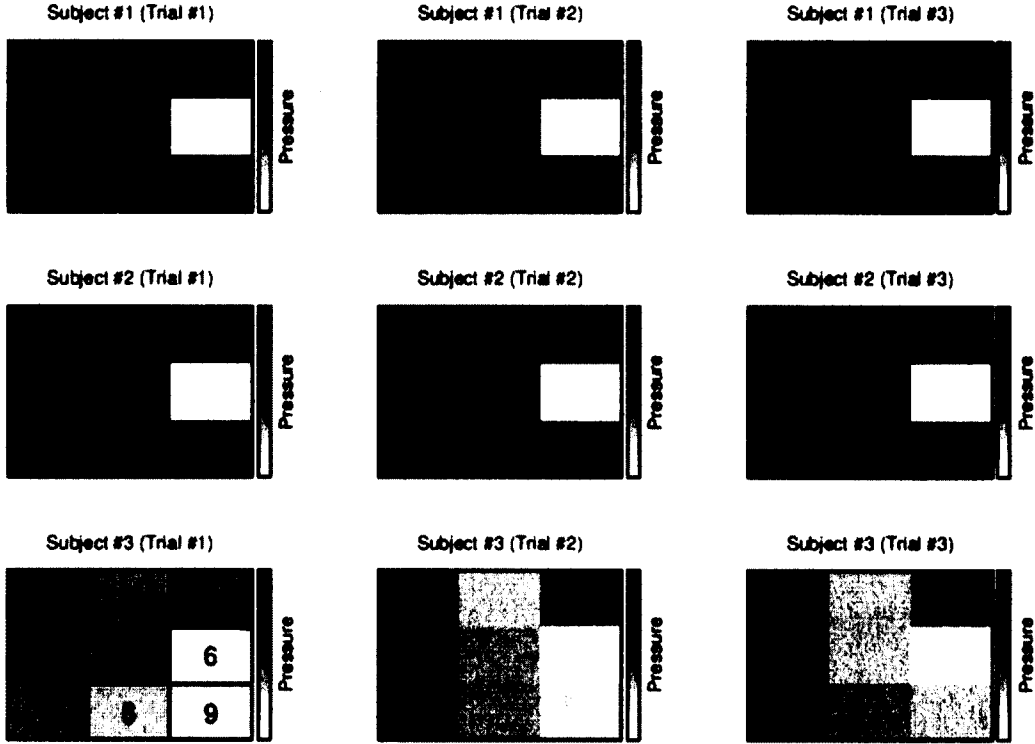


Figure 3.4: Users' tapping pressure on smartphone touchscreen, while entering an 8-digit PIN 1-2-5-9-7-3-8-4. Each figure shows pressure readings on a 3×3 smartphone number pad. Darker color indicates a larger tapping pressure. Note that number "6" has no pressure because it is not in the PIN. Figures in the same row are from a same user while typing the PIN for three times.

## Dissimilarity Measures

We represent each user action as  $n$ -dimensional feature vectors, where  $n$  is the number of feature dimensions. Using the *dissimilarity score* between two feature vectors, we further verify if our extracted features of a user remain relatively stable over multiple repetitions, in comparison with those of the other participants.

As the first step, we compute a target user's *template* as an average feature vector over its  $N$  PIN tapping actions, where  $N = 150$  in our case. At the same time, each feature's standard deviation is computed based on these  $N$  actions.

In our approach, given a new biometric data sample, we evaluate its *dissimilarity score* from the target user's template as follows. Suppose the new data sample's feature vector is  $\mathbf{X} = \{X_1, X_2, \dots, X_i, \dots, X_n\}$ , where  $X_i$  represents the  $i$ th feature dimension; and the target user's template is represented similarly as

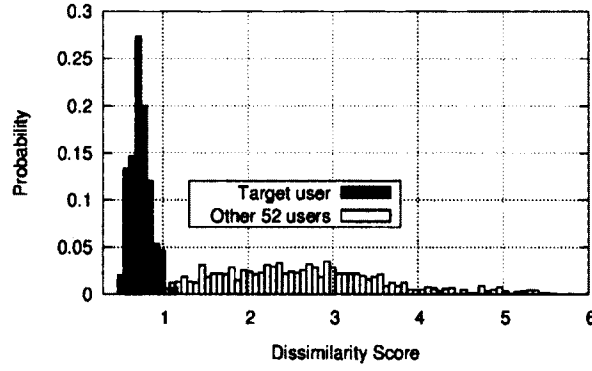


Figure 3.5: Distribution of dissimilarity score of typing 3-2-4-4 from a target user's template, to both the target user itself and other 52 users.

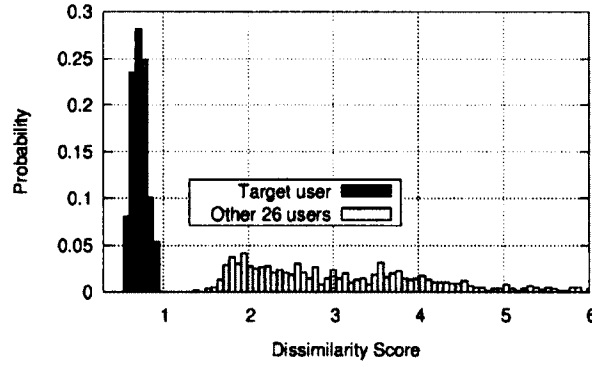


Figure 3.6: Distribution of dissimilarity score of typing the 8-digit PIN 1-2-5-9-7-3-8-4 from a target user's template, to both the target user itself and other users.

$\bar{T} = \{\bar{T}_1, \bar{T}_2, \dots, \bar{T}_n\}$ . The dissimilarity score is the accumulated deviation from the two vectors over all *normalized* features:

$$D(\mathbf{X}, \bar{\mathbf{T}}) = \sum_i \left\| \frac{X_i - \bar{T}_i}{\sigma_i} \right\|, \quad (3.1)$$

where  $\sigma_i$  denotes the standard deviation of the  $i$ th feature over the  $N$  trials in obtaining the target user's template. By dividing  $\sigma_i$ , we give higher weights to those features that have smaller variation within the target user, because they more reliably reflect the target user's specific pattern. This is a standard procedure mostly seen in outlier removal (also known as standard score or z-score in statistics [98]).

Figures 3.5, 3.6, and 3.7 show the distributions of dissimilarity scores, calculated from a target user's template entering three different PINs, respectively, to

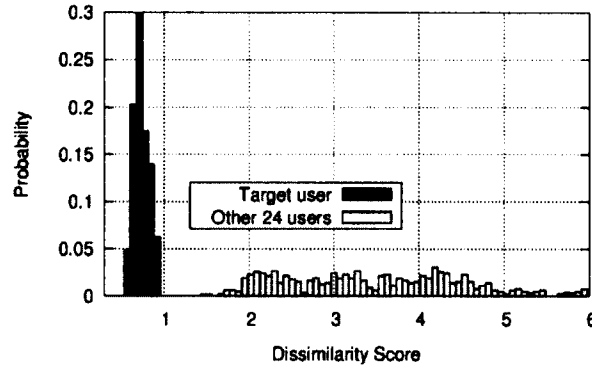


Figure 3.7: Distribution of dissimilarity score of typing another 8-digit PIN 1-2-5-9-8-4-1-6 from a target user's template, to both the target user itself and other users.

both the target user itself and the rest of other users. It is clear that in all three PINs, the dissimilarity scores to the target user itself is highly concentrated on the lower end, indicating a high similarity to its own behavioral template. Meanwhile, the dissimilarity scores of *other* users are dispersed and located on the higher end. For the 4-digit PIN 3-2-4-4 (Figure 3.5), there is a small overlap of the target user itself with others. It implies that only few members among the other 52 users behave similarly to the target user, and may be misclassified. For the two 8-digit PINs (Figures 3.6 and 3.7), the target user's and others' distribution curves are *completely* separated with a clear gap in between. Likely this is because an 8-digit PIN action contains more cognitive information that is user-specific than a 4-digit PIN action.

### 3.4 Classification

The system architecture of our approach consists of a feature module, a classifier module, and a decision maker module as shown in Figure 3.8. Firstly, raw data are recorded during user's tapping actions. Then, four sets of features are calculated and fed into the classifier, which derives a decision score featuring its similarity to the target user's template. The decision score is used by the decision maker to make a final decision, with respect to a predefined threshold value.

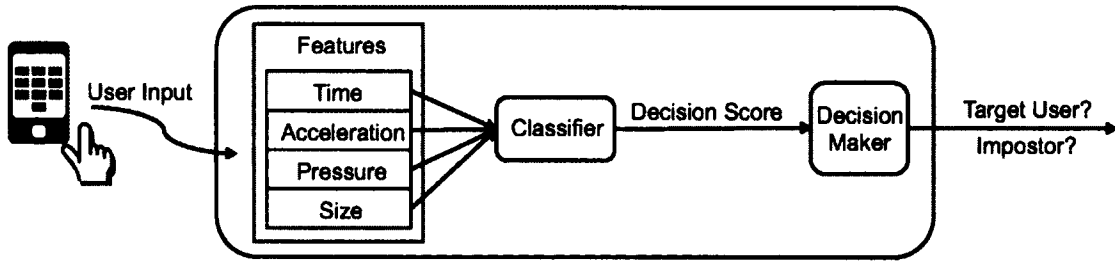


Figure 3.8: System Architecture

The final decision is to label whether an user tapping action is originated from the target user or an impostor.

User behavioral pattern can be derived from either one-class or two-class learning. In this chapter, we study both of the classification schemes. In one-class learning, only the target user's data is needed in training phase; but the learnt model can be applied to classify both the target user or an unknown impostor. Additionally, if other users' data are available, together with the target user's own data, we can conduct a two-class learning. One-class learning is straightforward and more practical because it does not involve other users' data, but with lower verification accuracy. For a two-class classifier, device manufacturers could pre-load some anonymized user data into smartphones before shipping them to their customers. With the pre-load anonymized user data, two-class classification is also feasible to perform in practice and can achieve higher verification accuracy. In the following sections, we describe both one-class and two-class learning process of our approach in detail.

### 3.4.1 One-Class Learning

Our one-class learning process consists of the enrollment and testing phases. In enrollment of a target user  $I$ , taking its  $N$  input actions, we calculate the standard deviations of every feature as  $\sigma_j$  for the  $j$ th feature. In the testing phase, given an unknown sample as  $n$ -dimensional feature vector  $X_Q$ , its distance from each

of the  $N$  feature vectors in the enrollment phase is calculated as:

$$d(X_Q, X_i) = \sum_{j=1}^n \frac{\|X_{Q,j} - X_{i,j}\|}{\sigma_j}, \quad i = 1, \dots, N, \quad (3.2)$$

where  $X_{Q,j}$  is the  $j$ th feature of feature vector  $X_Q$ , and  $X_{i,j}$  is the  $j$ th feature of the  $i$ th feature vector in the enrollment phase. Following this, the distance of  $X_Q$ 's nearest neighbor  $d_{min}(X_Q, I)$  will be chosen as the dissimilarity measurement to the target user's template. The underlying assumption is that if  $X_Q$  belongs to the target user, it should have a short distance to its nearest neighbor in the target user's data. And if  $d_{min}(X_Q, I)$  is below a pre-defined threshold value, it is labeled as from the target user; otherwise, it is labeled as from impostors. Implementation wise, setting a large threshold value means a higher probability of recognizing the target user, but allowing more impostors slip through. A small threshold value strictly blocks out impostors, but may falsely reject the target user.

### 3.4.2 Two-Class SVM for User Verification

Here we adapt user verification as a two-class problem: one class includes the behavioral features of the target user, and the other class denotes the features of *other* users. We choose support vector machines (SVM) as our two-class classifier, due to its good accuracy and efficiency in various applications including face recognition [87], text categorization [57], and image classification [81].

In our implementation, we utilize the open source SVM package LIBSVM 3.12 [18] to perform all the two-class classifications. LIBSVM is an integrated tool for support vector machine learning. The default Radial Basis Function (RBF) kernel is used as the kernel function and cross-validation is applied to find the best parameters  $C$  and  $\gamma$ .

In training, feature vectors from the target user are labeled as negative (-1), and those from others are labeled as positive (+1). In testing, given a feature

vector from an unknown user, the SVM classifier outputs a decision score, which evaluates how probable the unknown features are from the target user. By default, LIBSVM will predict those with a negative decision score as negative cases, i.e., indeed from the target user; otherwise, positive decision scores lead to positive cases, and will be labeled as impostors.

Again, similar to one-class learning, a threshold value can be added to tune the classifier towards either the target user or impostors, based on the decision scores. Setting a positive threshold makes a *strict* classifier, i.e., very sensitive to anomalous behaviors; while setting a negative threshold will tolerate more on the target user's behavioral changes, but less effective in blocking impostors. We will discuss the tradeoff of setting the threshold in the next section.

### 3.5 Experimental Evaluation

Generally, the accuracy of a biometrics-based authentication is evaluated by the following error rates:

- False Reject Rate (FRR) — the probability that a user is wrongly identified as an impostor;
- False Accept Rate (FAR) — the probability that an impostor is incorrectly identified as a legitimate user.

The point at which both FAR and FFR are equal is denoted as the Equal Error Rate (EER). The value of EER can be obtained by tuning a certain threshold until FAR and FAR are equal.

A formal description of a biometric-based verification system is summarized as [55]: given an unknown sample to be verified towards a target user  $I$ , its feature vector  $X_Q$  is compared with the target user's template  $X_I$ . A dissimilarity score  $D(X_Q, X_I)$  is calculated, where  $D$  is a function that evaluates the dissimilarity between two feature vectors. The dissimilarity function  $D$  varies with different

methods of classification. Finally, a threshold value  $t$  is set to determine if  $X_Q$  is from the target user or an impostor:

$$(I, X_Q) \in \begin{cases} \text{target user,} & \text{if } D(X_Q, X_I) \leq t \\ \text{impostor,} & \text{otherwise} \end{cases}$$

This structure applies to both one-class and two-class learning described in Section 3.4. The only difference is their dissimilarity functions  $D$ . Tuning the threshold would give the classifier a preference towards either the target user or the impostors, thus reducing one error rate while increasing the other. An illustration of the tradeoff between FRR and FAR, by tuning the threshold value is presented in Figure 3.9.

We can see in Figure 3.9 that as the threshold value is tuned along the X-axis, the outputs (shadow areas) favor either security or user convenience. For security-critical applications, one might want to have a guaranteed zero percent FAR. It means even at the cost of inconvenience to legitimate users, no impostor is able to get in. This kind of system should tune the threshold at the borderline of the minimum dissimilarity of the impostors' data. On the other hand, for logging into non-security-critical applications, or in a situation that security is less concerned such as at home, usability is more important than perfect impostor rejection. This kind of system should tune the threshold at the borderline of the maximum distance of the target user's data. Because our approach acts as a second factor authentication, which supplements the passcode-based mechanisms for higher assurance authentication in a cost-effective fashion, we focus more on being user-transparent and user-friendly while enhancing the security of PIN-based authentication.

In the following, we present the evaluation results of both one-class and two-class verification systems, along with the effect of threshold and number of ac-



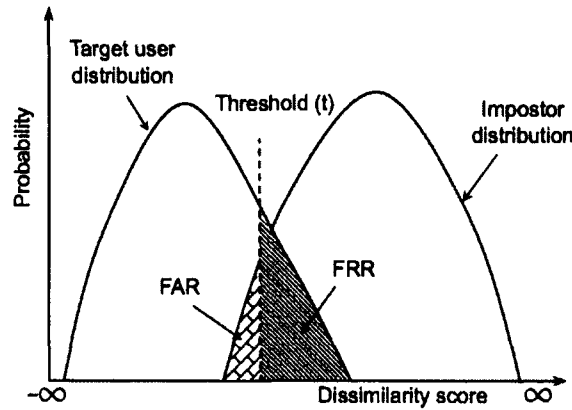


Figure 3.9: An illustration of tradeoff between FRR and FAR by tuning the threshold [55].

tions in training, the comparison with different combination of PINs, and the associated system overhead.

### 3.5.1 One-Class Verification Accuracy

There are two parameters that affect the accuracy in one-class learning: the number of actions in training, and the threshold.

By increasing the number of actions in training, user behavioral patterns become more precise since more actions yield a higher statistical significance. Figure 3.10 shows that the averaged equal error rate (EER) decreases as more user actions are included in training. All five PIN combinations present similarly shaped curves, while the results of 1-1-1-1 and 5-5-5-5 are less accurate than those of 3-2-4-4 and the two 8-digit PINs. From lower accuracy of 1-1-1-1 and 5-5-5-5, it seems that a PIN number with higher repetition of digits reduces the difference in individual users' tapping behaviors, leading to a less accurate verification result. Moreover, for all five PINs, the accuracy remains on a similar level after 20 user actions. This implies, as more user actions are added in training, there is a diminishing gain in accuracy. For example, increasing user actions from 20 to 40 requires twice the time waiting for user input, but only limited accuracy increase is seen.

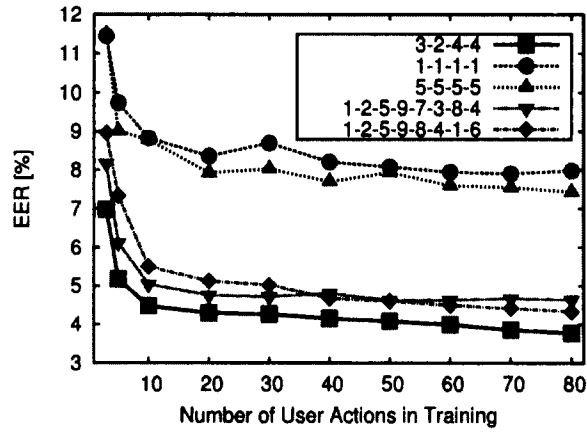


Figure 3.10: Variation of average EER with number of user actions in one-class training for five PIN combinations.

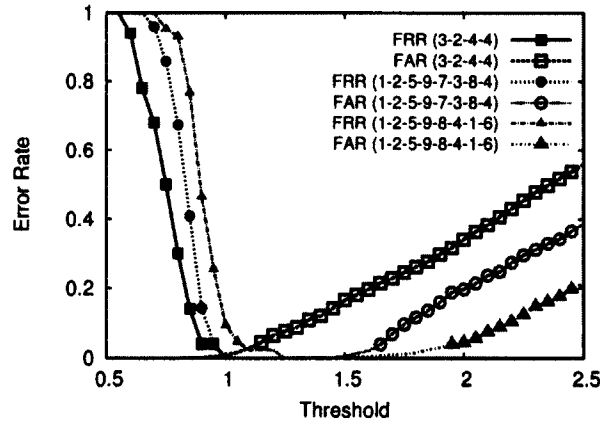


Figure 3.11: Variation of FRR and FAR with the value of threshold for 3-2-4-4 and the two 8-digit PINs for one target user.

The second column of Table 3.3 further lists the exact values of averaged EER, with its standard deviation in parenthesis. In computing EERs, there are 85 user actions included in the training process. As shown in Figure 3.11, in all three PIN combinations, there is a trade-off between FRR and FAR.

As mentioned earlier, four sets of features are included: acceleration, pressure, size, and time. To measure how the four sets of features contribute to the final accuracy, we make four additional rounds of classification, solely based on each feature set. Figure 3.12 shows the accuracy results for the four individual feature sets, as well as those of combining them all together.

It can be seen from Figure 3.12 that, the combination of all four feature sets always outperforms individual feature set, as it is always with the smallest EER in

PIN	EER (One-Class) <sup>a</sup>	EER (Two-Class) <sup>a</sup>
3-2-4-4	3.65% (3.58%)	3.68% (4.38%)
1-1-1-1	6.96% (6.01%)	7.01% (6.45%)
5-5-5-5	7.34% (5.38%)	5.27% (3.69%)
1-2-5-9-7-3-8-4	4.55% (6.23%)	3.21% (4.89%)
1-2-5-9-8-4-1-6	4.45% (4.15%)	4.51% (3.45%)

<sup>a</sup>with standard deviation in parenthesis

Table 3.3: User Verification Accuracies

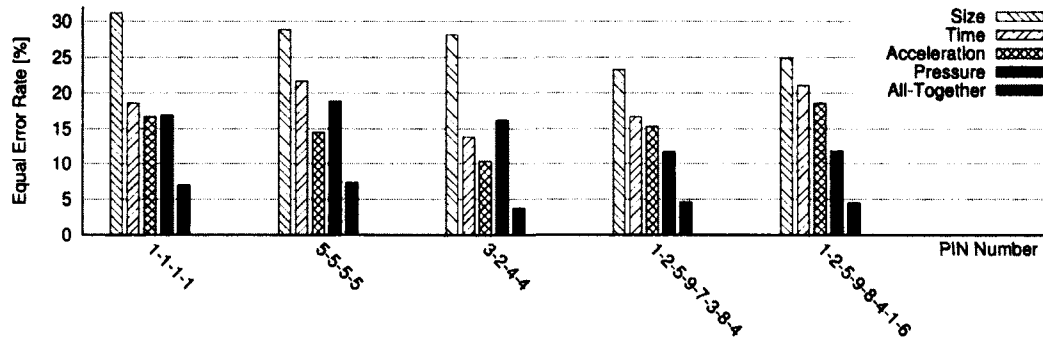


Figure 3.12: Comparison of performance from each feature set in one-class learning, as well as when they are combined together.

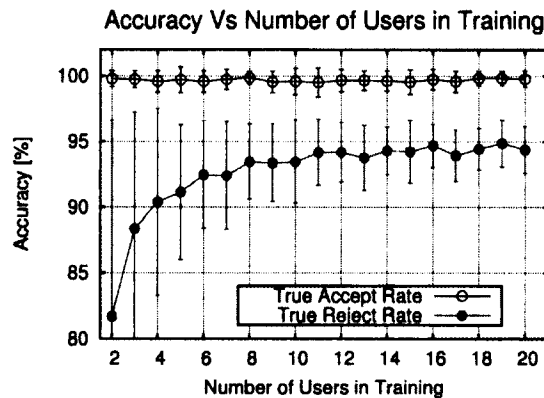
all different scenarios. This is because the four feature sets capture the different aspects of user tapping behaviors, and having them all together should most precisely represent who the target user is. Meanwhile, among the four individual feature sets, acceleration, pressure, and time perform similarly well and achieve more accurate results than size.

### 3.5.2 Two-Class Accuracies

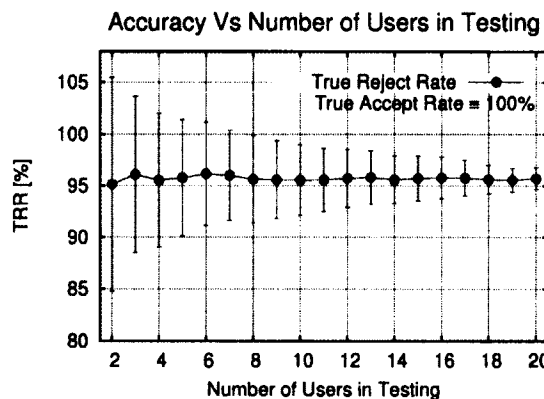
Unlike one-class learning, a two-class classifier is trained on both the target user's and other users' data. Our experimental results show that two-class classifiers usually yield higher accuracy than one-class. Thus, if others' data are available to the target user, the two-class classifier can be an optional implementation to achieve higher accuracy. In fact, this can be done by devices manufactures or mobile OS providers (e.g., Google) to pre-load the two-class classifier with

anonymized user data for training purpose.

In training an SVM classifier, 50 of the target user's actions are input as negative cases, and 50 of *others'* actions (randomly selected from our data set) are input as positive cases. The classifier is tested on the rest of the target user's actions and the other users' data. We make sure that there is no overlap between impostors in training and testing. In this way, the classifier aims to detect *unknown* impostors outside the training set.



(a) Accuracy Vs Number of Users in Training



(b) Accuracy Vs Number of Users in Testing

Figure 3.13: Variation of Accuracies with number of users in training and in testing.

Figure 3.13(a) shows the dynamics of accuracy under the different number of other users (i.e., potential impostors) in training, in terms of true reject rate (TRR) and true accept rate (TAR). For easy presentation, we use "impostor" and "other users" exchangeably in the following part of this section. The error bars show

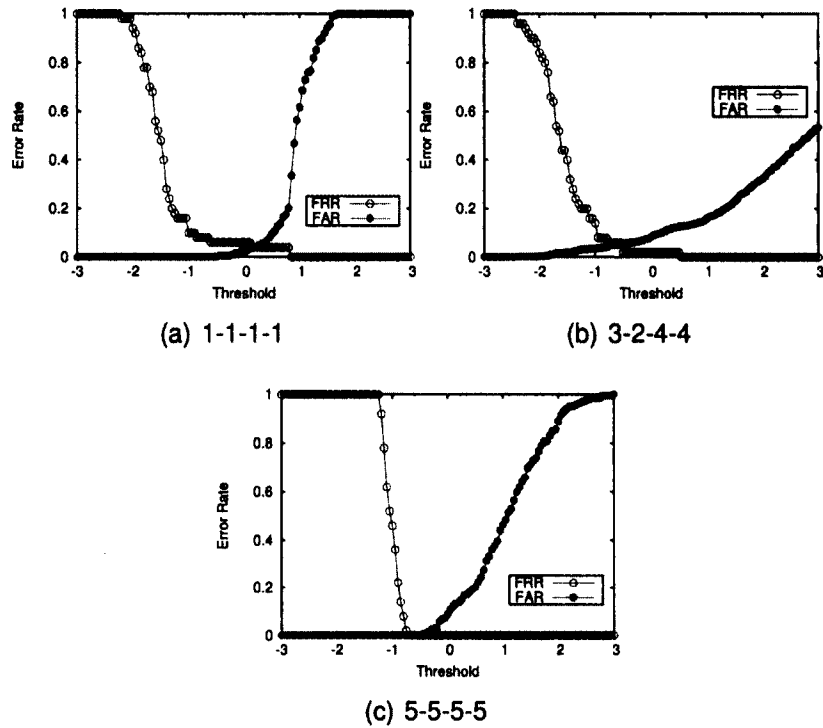


Figure 3.14: Trade-off between FRR and FAR in typing three 4-digit PINs: 1-1-1-1, 3-2-4-4, and 5-5-5-5.

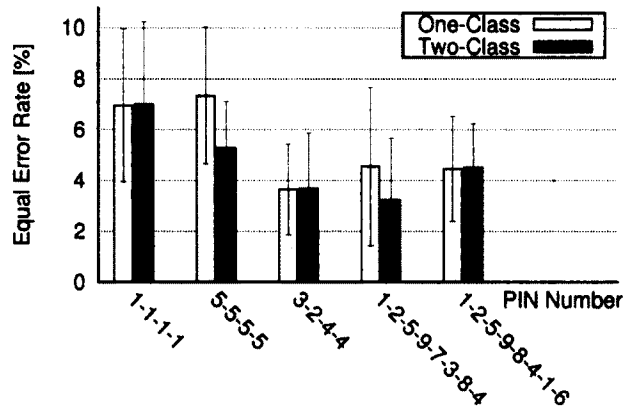


Figure 3.15: Comparison of accuracies between one- and two-class learning. Lengths of error bars show the standard deviation of EER over users.

standard deviations over different combinations of the impostor set. We can see that more other users involved in training leads to a higher and more stable TRR (the rate of detecting an impostor). In the mean time, TAR stays relatively on the same level, regardless how many other users are involved in training.

Besides the number of training impostors (i.e., other users in training), we

also evaluate the verification accuracy with respect to the varying number of testing impostors. Figure 3.13(b) shows the dynamics of accuracy under the different number of impostors in testing, in terms of true reject rate (TRR) and true accept rate (TAR). Again, the error bars show standard deviations over different combinations of the impostor set. With a larger impostor pool in testing, the average value of TRR stays similar, while its fluctuation getting smaller. It indicates that our method is robust in detecting a large pool of potential impostors.

Similar with the one-class classifier, there is also a trade-off between FRR and FAR in the two-class approach. Figures 3.14(a), 3.14(b), and 3.14(c) show the trade-offs between FRR and FAR in entering three different 4-digit PINs by a target user, respectively. We can see that by tuning the threshold value, one of the error rates decreases at the cost of increasing the other one.

The third column of Table 3.3 shows the averaged EERs for the two-class classifier, and Figure 3.15 visually compares the accuracy results with those of the one-class approach. While the two-class classifier achieves the similar accuracy as the one-class classifier in three scenarios, it clearly outperforms the one-class classifier in the other two scenarios.

### 3.5.3 System Overhead

In our implementation, the verification system is entirely built on a smartphone. As a stand-alone system, there is only a single user present for verification at any given time. There is no communication overhead associated with our user verification.

We first estimate the memory overhead of the verification process. The verification process is profiled using the Android SDK tool DDMS, and we find out that it only consumes 11.195 MBytes of heap memory during a one-class testing process. The computational overhead is the sum of CPU costs in raw data processing (calculating features) and detecting (including classifying and deci-

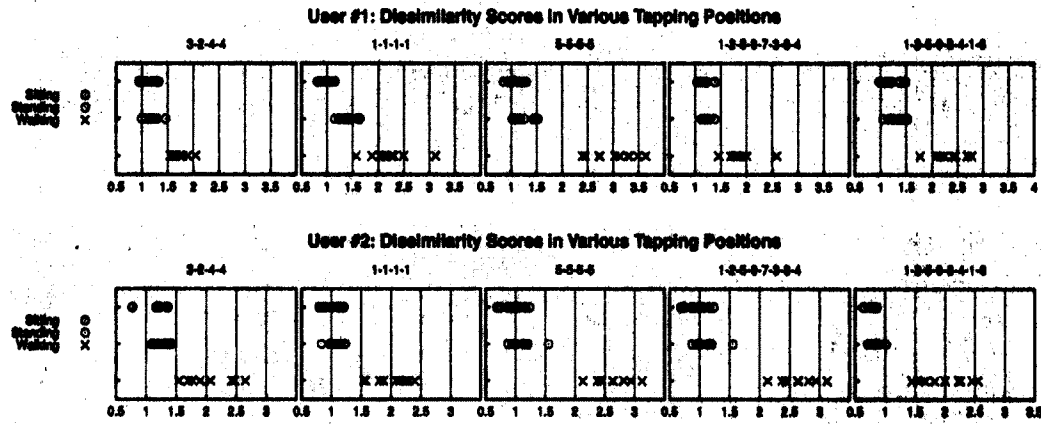


Figure 3.16: Dissimilarity scores from one tapping position (i.e., one-handed while sitting) to the other two tapping positions of a same user. Upper and lower panels are for the two different users, respectively.

sion making). The pre-processing on one user input action of a 4-digit PIN takes only 0.022 seconds. The detecting process takes another 0.474 seconds, where the major part lies in finding the nearest neighbor from all 85 reference feature vectors. The CPU cost is measured on a Samsung Galaxy Nexus, using two `Date.getTime()` utility call at the beginning and end of the running time. Overall, the induced computational overhead is minor on the smartphone. In terms of disk space for storing user template, the signature of a single user profile generated by the training process consumes only 150.67KBytes. It is very affordable on an entry-level smartphone, let alone high-end models.

## 3.6 Additional Issues in Reality

### 3.6.1 Multiple Positions

So far we only measure the user tapping behaviors in a given position. However, it is quite possible that a user types in its passcode under different positions (e.g., single handed using the thumb). To handle different input positions, we can measure and store multiple behavioral patterns for different positions during the training period. The rich sensors equipped with smartphones allow us to easily detect the physical position of the device and choose the appropriate behavioral

pattern for verification. For example, accelerometer readings straightforwardly signal if a user is in a moving or non-moving status. And gyroscope readings can even infer the user's hand position (one-handed vs. two-handed) when tapping on the smartphone, as shown in a recent study [46].

To explore on multiple tapping positions, we further conduct two more sets of empirical measurements. First, we collect data with only one-handed tapping, which is in parallel with the two-handed case in Section 3.5. In one-handed tapping, the phone is held in one hand, and tapped with the thumb finger of the same hand. Due to the use of a different finger, the one-handed tapping behavior is different from that of two-handed. However, with the one-handed data set as the training data, we perform the same evaluation process as in Section 3.5 and achieve an average EER of 3.37% over all PINs in the on-handed case, indicating the effectiveness of our approach just like in two-handed tapping.

In addition to different hand positions, there are also various body positions a user would switch from time to time. While tapping its passcode, a user can be sitting, standing, or even slow walking. It is desirable to see how different body positions affect a user's tapping behavior. To answer this question, we carry out an additional experiment with two users, who tap in PINs with three body positions: sitting, standing, and walking. Using the trained model with one-handed tapping while sitting as the baseline, Figure 3.16 shows the dissimilarity scores to three different tapping positions, sitting, standing, and walking. Our major observation is that: as long as the user remains static, its tapping behaviors are similar under different body positions. Note that we do *not* intend to cover all possible tapping positions in different environments (which is almost impossible), but instead to draw some insights based on the common scenarios.

Our approach will work well for different input positions: sitting or walking, single-handed or two-handed. The challenge is merely to increase the training period and cover different input positions with more feature sets, which will im-



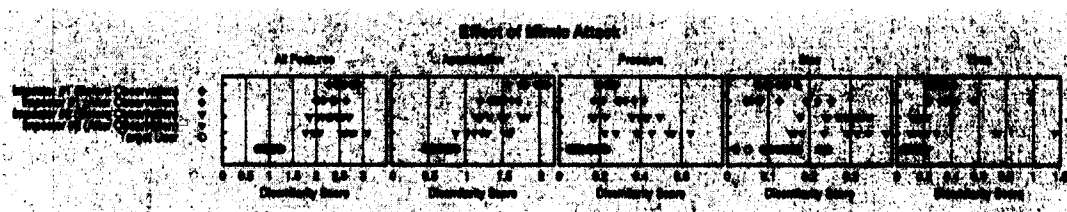


Figure 3.17: Effect of mimic attack shown in degree of dissimilarity from the target user. Subfigures, from left to right, correspond to all features considered, and each individual feature set (acceleration, pressure, size, and time). There are two mimic “impostors”, and 10 trials before and after their observations on the target user.

pose a larger memory and CPU overhead in verification. However, we could further reduce the system overhead by optimizing the classifier implementation from different aspects of mobile devices. Note that the current one-class classifier has not been optimized. We will further explore this direction in our future work.

### 3.6.2 Mimic Attacks

Theoretically, our behavior-based verification system can be bypassed if an impostor can precisely mimic the tapping behaviors of the device’s owner. However, this is extremely difficult if not impossible in practice. Intuitively, even if the impostor has overseen how the device’s owner previously entered the passcode, it might be able to mimic the timing aspect. But the other features, such as pressure, acceleration, and size, are much more difficult to observe and reproduce.

In order to quantitatively measure the effect of mimic by observation, we set up an experiment involving three users. One of them is the target user who is observed closely by the other two “impostors” who try to mimic the target user’s tapping behaviors. Impostor #1 has the same gender and similar hand/finger size as the target user, while impostor #2 has different gender and larger hands/fingers than the target user. The goal is to see how physiological differences can impact the outcome of mimic attacks. Before mimicking, the two impostors closely observe (over the shoulder) how the target user tapped in the PIN 3-2-4-4 for 10 times each. They are also guided to especially pay attention to the four features

in our approach, i.e., tapping rhythm, acceleration, pressure, and touched size.

Figure 3.17 plots the dissimilarity scores from the target user's model to the two impostors before and after their mimic trials. The leftmost subfigure corresponds to all features included, and the rest four correspond to individual contribution from the four different features, respectively (acceleration, pressure, size, and time). Our experimental results clearly show that there is no significant improvement in mimicking given the behavior observation. Taking all four features into account, it is evident that a mimic attack is very hard to succeed. For each individual feature (acceleration, pressure, size, and time) shown in Figure 3.17, we can see that only the dissimilarity scores of acceleration are consistently reduced (i.e., its score range shifts towards that of the target user after observation). However, for the other three features (including pressure, size, and time), out of the 10 mimic attempts, just one or two trials may be slightly closer to the target's model, but their score ranges spread even wider. Thus, the behavior mimicking does not increase the chance of evasion with respect to these three features. This somewhat contradicts our intuition that timing would be easier to mimic than the other features.

With the experimental results shown in Figure 3.17, we believe that the robustness of our approach against a mimic attack mainly lies in the following three aspects.

- There are multiple dimensions in the features we used and most of them are independent from each other. Although an impostor may mimic one dimension without much difficulty, mimicking multiple dimensions simultaneously is extremely difficult as small physical movements like tapping are hard to observe and precisely reproduce. For example, acceleration directly relates to tapping force ( $F = m \cdot a$ ), so if the impostor intentionally manages to tap in a gentler or harder fashion, its behavior can get closer to that of the target user. However, pressure is harder to mimic because it equals

to tapping force divided by touched area. These two independent factors must be adjusted at the same time, which is more challenging. Timing (or tapping rhythm) is also hard to mimic, because timing contains multiple dimensions in our approach: 7 in a 4-digit PIN, and 15 in a 8-digit PIN. Those individual time intervals (especially key-to-key intervals) are relatively independent. An impostor may mimic the target user with a roughly fast or slow rhythm, but it is hard to reproduce the specific key-to-key dynamics.

- The fine-grained measurement resolution makes our features hard to mimic. For example, in our experiment, timing is measured in order of millisecond. This time resolution is much higher than human perception, and hence it is very hard for an impostor to accurately and consistently mimic tapping rhythm at such a low-level resolution.
- The physiological differences from the target user set up another barrier for mimic impostors. In our feature set, the touched size is heavily affected by the finger size, and the tapping rhythms also depend on hand agility and geometric shape. In general, it is very difficult for a person with bigger hand/fingers to mimic someone with smaller hand/fingers, and vice versa.

As more sensors have been available on mobile devices, more features will be included for more accurate user verification, and hence mimic attacks will just become less likely to succeed.

### **3.6.3 User Behavior Changes**

This work builds on the assumption that a user's behavior is consistent and no abrupt change happens over a short period of time, but the assumption might not always be true, e.g., due to a physical injury. In such scenarios, the behavioral-based verification mechanism should stay minimally intrusive to the user. One feasible solution is to contact with the service providers to disable the verification

function remotely and start the re-training. The purpose of our user verification is to provide additional security in common day-to-day usage while still allowing the user to disable it in rare cases. As we have shown previously, the sensitivity to false positives and negatives are controlled by various threshold values. Whether or not exposing the sensitivity control, e.g., setting it to Low, Medium, and High, can improve user experience is debatable. On one hand, it allows users to make a conscience choice to trade off between security and convenience. On the other hand, it is no longer user-transparent.

#### **3.6.4 Passcode Changes**

In our approach, only the tapping features of the currently active passcode are measured and recorded in a user's smartphone. One might ask what happens when the user need to change its passcode? Although people do not frequently change their passcodes, updating passcode in a quarterly or yearly basis is recommended or required by most passcode-based systems. When this happens, our verification system could automatically remain inactive for a while and start another training session to build a new set of tapping features based on the newly created passcode. The characterization of tapping features are conducted in background till a stable pattern has been successfully compiled after multiple trials. Note that the methodology of our scheme is not bounded to certain passcodes. In other words, our approach can be applied to any passcode a user chosen in practice.

### **3.7 Conclusion**

As mobile devices are getting widely adopted, ensuring their physical and data security has become a major challenge. A simple peek over the shoulders of the device owner while the passcode is being entered and a few minutes of hiatus

would allow an attacker to access sensitive information stored on the device. Using more complex passcodes and/or secondary passcodes can reduce the chance of such attacks, but it brings significant inconvenience to the users. We found that a user's tapping signatures if used in conjunction with the passcode itself can also achieve the same goal, and moreover, the added security can be obtained in a completely user-transparent fashion.

Previous works have shown the feasibility of this approach, but their high error rate makes these mechanisms impractical to use as too many false positives will defeat the purpose of being user-transparent. Having collected data of over 80 different users, explored both one-class and two-class machine learning techniques, and utilized additional motion sensors on newest generation of mobile devices, we are able to demonstrate accuracies with equal error rates of down to 3.65% for 4-digit PINs, and 3.21% for 8-digit PINs.

## **4 Exploring Fitts' Law in Web Browsing**

In this chapter, we describe our work on applying Fitts' Law to model natural web browsing behaviors from end users. The chapter is organized as follows. Section 4.1 describes the background of Fitts' law and web browsing behavior, and surveys related work. Section 4.2 details our data collection and processing. Section 4.3 evaluates the validity of Fitts law under natural web browsing, along with its proposed error model. Section 4.4 discusses several Fitts' law related issues. Section 4.5 concludes the chapter.

### **4.1 Background and Related Work**

A large number of research works have been conducted to learn web browsing behaviors, with the goal of measuring user interests [17, 68, 94], web page quality [97], search quality [1, 52], predicting user demographic [51], and providing personalization [101]. The existing studies are heavily based on information of pageview activities, including pageview paths, time spent on webpages, frequencies of webpage visiting, etc. By contrast, in this chapter, we explore human browsing behavior from a different perspective: the kinetics of user point-and-click actions in web browsing. Our work is useful in complementing previous works to better model user browsing behavior in a more comprehensive manner.

We focus on studying Fitts' law, one of the most influential laws in human-computer interaction research for decades. Essentially, Fitts' law reveals the length of time it takes to perform a task with a pointing device such as a mouse. For instance, how long does it take to move the mouse cursor to a particular position on the screen. It is expressed as in Eq. 1.1. The significance of Fitts' law is that it provides quantitative information regarding the accumulated time of multiple perceptual-motor feedback cycles for users to interact with a system using a pointing device.

Fitts' law is closely related with several theories on submovement analysis, and the major ones, in chronological order, consist of, *the iterative corrections model* [29,61], *the impulse variability model* [92] (also known as the Schmidt's law), and *the optimized initial impulse model* [74] (also known as the Meyer's law). A comprehensive review of the three models can be found in [89]. The first two models emphasize on either solely feedback control or solely initial impulse, while the third model (the Meyer's law) combines these two views, and gave a satisfactory explanation supported by empirical evidence. Therefore, the underlying message from Fitts' law is an optimal planning of human motor-control bounded by speed-accuracy tradeoff<sup>1</sup>. In another word, even in a task as simple as reaching for a target, human motor skill automatically balances the speed and accuracy in an optimal way, with an outcome of target-reaching both accurately and rapidly.

Note that Schmidt's and Meyer's laws can serve as independent models for pointing actions, and they are closely related to Fitts' law. The former is for rapid pointing, and the latter constitutes a more generalized law (combining both Fitts' and Schmidt's laws). However, in terms of application, Schmidt's and Meyer's laws require submovement analysis at low level, while Fitts' law only involves measurements (i.e., *total* movement time, *total* distance, and target size) without

---

<sup>1</sup>In presence of speed-accuracy tradeoff, one cannot accurately aim for a target with no error while moving extremely fast.

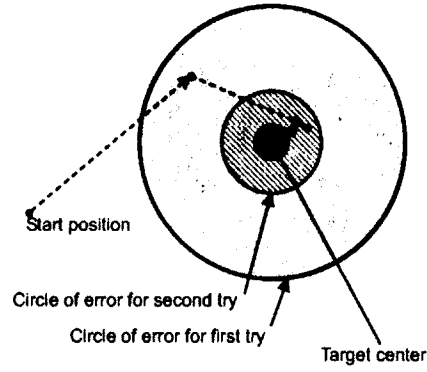


Figure 4.1: Step-wise movement towards target [35]

any submovement-level variables. Therefore, due to its simplicity and ease-of-use, along with its success in many aiming-related experiments [6, 54, 62, 63, 69, 89], we have chosen Fitts' law as the focus of this study, instead of the other laws. And our objective is to specifically verify if Fitts' law alone is applicable for modeling daily pointing actions with computer mice in a natural web browsing environment.

Here for simplicity, we present the first model, i.e., the iterative corrections model, proposed by Crossman and Goodeve [29] in 1963. This is not a perfect model in itself, but it especially reveals how the logarithmic term in Fitts' law came from perceptual-motor feedback loops. Figure 4.1 shows how the movements in each step get gradually smaller as the target gets closer, involving discrete cycles of sensing and movement [35].

It is assumed that each of submovement reduces the distance to the target geometrically, that is, it moves a constant fraction  $1 - r$  of the remaining distance. Each of them takes the same time  $t$ . When remaining distance is such that the error circle of the remaining movement is less than the size of the target then we get inside the target. When the target has been reached:

$$r^N A = \frac{W}{2}.$$



Solving for  $N$ :

$$N = \frac{1}{\log_2 1/r} \log_2 \frac{2A}{W}.$$

The time required for completing all submovements are:

$$T = Nt = \frac{t}{\log_2 1/r} + \frac{t}{\log_2 1/r} \log_2 \frac{A}{W}.$$

This especially explains the logarithmic term in Fitts' law.

A large body of prior works have been dedicated to Fitts' law, since it was first proposed in 1954 [39]. Seminal works in the HCI (human-computer interaction) community include [15], [96], [108], [113], etc. However, only a few of existing literatures concern with real-life pointing behaviors, based on unobtrusively collected data.

Chapuis *et al.* [20] are among the first to notice the need to stress-test Fitts' law in natural GUI settings. They questioned if one can apply the Fitts' law obtained from controlled laboratory experiments to characterize the pointing activities "in the wild". Their underlying motivation is the same as ours, pointing "in the wild" involves far more cognitive processes than in a controlled laboratory setting, such as deciding what is the target, coping with possible interference from the field environment or planning for higher-level tasks. In their field study of 24 users, the results indeed deviate from those in controlled laboratory environments.

Slijper *et al.* [95] applied Bayesian statistics to model hand movements, drawn from a large-scale collection of users' daily mouse movements. Human arm movements are found to be strongly correlated to prior experience, making them predictable via Bayesian statistics analysis. Thus, Slijper *et al.* achieved their primary goal, which is to predict hand moving directions by utilizing the directional distribution.

Hust *et al.* [53] conducted another field study on the evaluation of real-life pointing performance for motor-impaired people. High variance is found within

each participant, implying that it is insufficient to measure performance based on a single laboratory session. However, as target width and distance are not captured in the data collection, the theme of their study is not focused on the evaluation of Fitts' law. Moreover, since the experimented subjects are with motor impairments, it is unclear if a healthy person will still display considerable variance.

More recently, the differences between the natural and laboratory controlled mouse movements are further acknowledged in Gajos *et al.*'s work [44]. They found that, many of those mouse pointing movements "in the wild" are affected by extraneous factors, which include, for example, deciding what task to perform next and searching for the right user interface element. Motivated by this observation, the authors unobtrusively collected mouse pointer trajectories from 18 participants. A classifier is then trained to discriminate between deliberate, targeted mouse pointer movements and those movements that are distracted.

Meanwhile, Evans and Wobbrock [38] developed the *Input Observer*, a novel tool to passively collect user input data, from which they measured both text entry and mouse pointing performance "in the wild". With regard to pointing performance, the authors carefully measured target size and pointing errors by utilizing crowdsourcing. In the process of raw pointing data, a novel segmentation technique is employed to identify each trial, and further, outlier removal is used to damp the noise. As a result, the pointing performances "in the wild" are measured to be very close to that from laboratory studies, in terms of average pointing error, movement time, and throughput. However, in their work, only a small subset of collected data (~11% of pointing data) are considered in order to obtain laboratory-quality results; by contrast, in this chapter, as our goal is to see if Fitts' law works well in regular web browsing activities, there is minimal data filtering done.

Overall, our work significantly differs from these previous works in terms of

scale. Whereas the largest data set of the previous studies includes 24 volunteer users, our study involves two orders of magnitude more human users. There are more than 1,000 participants who are real-world Internet users, and their pointing actions are recorded while using a web browser. Moreover, we solely focus on pointing behaviors in web browsing, while the previous works study pointing data from many different applications at the client side.

## **4.2 Data Collection**

This work includes two different data sets, obtained under two different circumstances, for the purposes of two different types of measurements. The data sets are described below, followed by an explanation of how the data is processed into a meaningful form.

### **4.2.1 Data Sets**

In order to assess Fitts' law in a real-world setting, we first collect data outside of controlled laboratory conditions. More than 1,000 unique Internet users' mouse movements are recorded by JavaScript code embedded on a web forum<sup>2</sup>, and submitted passively via AJAX requests to the web server. Figure 4.2 shows the layout of the website homepage. Based on the vBulletin template, the webpage has a simple outlook comprised of text links stacked vertically. Icons and images are present too, but not visually prominent as text links. It is impossible to know about online users' biographical information (gender, age, education background), and there is no guarantee on the amount of data collected for a certain user (a forum user could be logged in for a long time with frequent mouse activities, or could perform just one click and then leave). On the other hand, the

---

<sup>2</sup> The website is generated from vBulletin forum software, and has a similar layout as <https://www.vbulletin.com/forum/forum.php>. In addition, we do not argue that this is a representative website, but it suffices as a case study for exploring the Fitts' law in the wild.

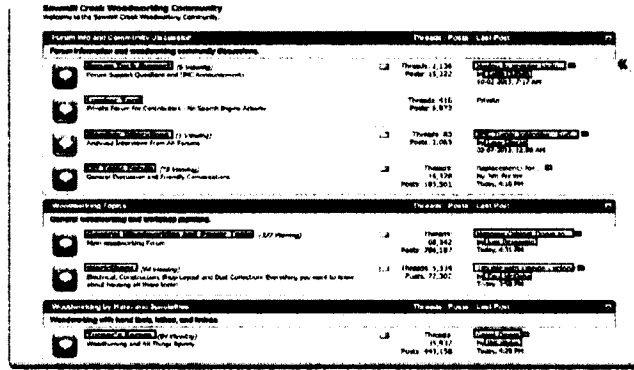


Figure 4.2: Layout of the webpage for data collection. Text links are highlighted by red boxes.

breadth of this corpus represents a large sample of real Internet users browsing naturally, as they were free to browse webpages without being interrupted. Thus, it is ideal for studying how well Fitts' law models natural pointing behavior when using a web browsing application.

The second data set is collected to study the effect on Fitts' law of pointing actions using different pointing devices. This data collection is conducted in a controllable environment. Ten people are invited personally to participate in the second set of data collection. They use two different pointing devices, mouse and touchpad, to interact with GUIs. During the data collection, the users' activities are performed naturally without any interference. Their pointing actions are recorded using the RUI tool [67].

## 4.2.2 Data Processing

The raw mouse movements are represented as the tuples of timestamp and Cartesian coordinate pairs. Each tuple is in the form of  $\langle \text{ACTION-TYPE}, t, x, y, \text{TARGET-TYPE} \rangle$ , where ACTION-TYPE is the mouse action type (a *mouse-move* or *mouse-click*),  $t$  is the timestamp of the mouse action,  $x$  is the x-coordinate,  $y$  is the y-coordinate, and TARGET-TYPE is the type of clicked target (a text link, text, image, or form element). All timestamps are collected in milliseconds, and coordinates in pixels.

These raw data points then are further processed to extract the pointing ac-

tions. Here we choose to consider only *point-and-click* actions with a text link target, because the overwhelmingly dominant majority of pointing targets in the forum website are text links. A point-and-click is defined as a continuous movement followed by a click. A continuous movement is defined as a sequence of movement with little to no pause between the beginning and end of the movement. Here “little to no pause” means that the time lapse is less than 100 milliseconds between any two adjacent mouse records. Here 100-millisecond is an empirical threshold, which is roughly the shortest time scale of human perception [16]. The time for a point-and-click action is measured from the first mouse-move event to the last mouse-move event. In our data collection algorithm, raw mouse records are only generated with either mouse-move or mouse-click events. Therefore, with respect to a pointing action with multiple pauses, as long as the period of zero-velocity is less than 100ms (which is true in most cases), multiple submovements are still preserved as one point-and-click. We only choose point-and-click actions because a movement that ends in a pause (rather than a click) could just be the user idly “shaking” the mouse cursor or moving it out of the way on the screen. These types of actions do not have a definite target, so they are not covered by Fitts’ law. Conversely, if a user clicks at the end of the movement within a web browser, highly likely it is clicking on a text link at the forum website, and thus we can assume that the pointing action has an aiming target—the text link. Note that our collected pointing data are associated with clicking *on* text-links, as missing the target would fail to send a server request. The only exception here is when a user clicks on a text-link accidentally while aiming for a different GUI target, but this is very unusual. Therefore, we assure that each collected point-and-click action within a web browser has a definite target and is error-free.

For each of these point-and-click actions, we calculate the time to complete the pointing action as the difference in timestamp from the first mouse-move event to the last one before the click. The *ID* is then calculated from the distance,

the difference in Cartesian coordinates between the beginning of the movement and the click, and the target size. It is important to measure the target size as accurately as possible. Several models have been proposed to extend Fitts' law to two dimensional targets. It has been evaluated by [70, 111] that one of the best model is this formula:

$$T = a + b \cdot \log_2\left(\frac{D}{\min(W, H)} + 1\right),$$

where  $\min(W, H)$  is the the smaller dimension of the height ( $H$ ) and width ( $W$ ) of a rectangle target. This model reflects the wide founded intuition that the smaller of  $H$  or  $W$  should dominant overall performance. For this experiment, we filter all point-and-clicks so that only those with a text-link as target are considered. Of all clicks, 17 pixels is the font height of the text hyperlink, which is the smaller dimension of the text link as a clicking target. For this very reason, we choose 17 pixels as a universal target width for all clicks on a text link.

To get the mean movement time  $MT$  in Fitts' formula Eq. 1.1, raw data must be averaged. We group the time to point values by ID into buckets of width 0.5, then each bucket is averaged to produce a single mean time (MT) point on the graph.

## 4.3 Evaluation

Fitts' law describes a linear relationship between the index of difficulty ( $ID$ ) of a pointing task and the mean time ( $MT$ ) to complete that task. We may not know the values of parameters  $a$  and  $b$  ahead of time, but we know the fit should be linear. Thus, to evaluate how well Fitts' law applies on a given data set, we plot the MT vs. ID of the pointing actions extracted from the data set. We calculate the correlation coefficient ( $r$ ) for each plot to measure how closely it fits a linear function. An  $r$  value of close to 1.0 means perfectly linear data (100% linear

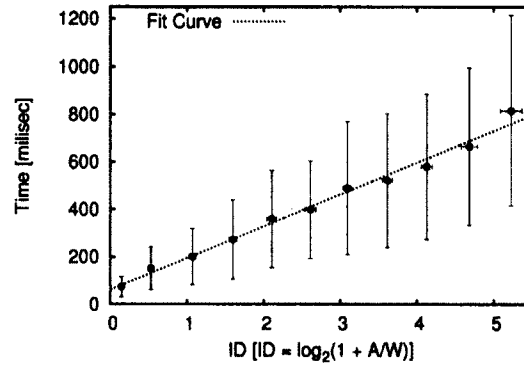


Figure 4.3: ID vs. Mean Time plotted for the forum users' data set. The linear correlation is over 98%. Every data point in the figure is averaged over 180 raw data points.

correlation), while an  $r$  value of close to 0.0 means completely random data (0% linear correlation).<sup>3</sup>

### 4.3.1 Fitts' Law in Natural Browsing

How well does Fitts' law apply in a natural web browsing environment? To answer this question, we evaluate the linearity of the first data set, containing 1,047 real-world forum users' data. Figure 4.3 shows the mean movement time ( $MT$ ) as a function of  $ID$ , where error bars show the standard deviation of movement time. The data fit a linear regression with  $r$  showing a 98.28% correlation, which strongly suggests that the forum data follows Fitts' law. We can safely conclude from this that Fitts' law is robust enough to have real-world applications, not just under contrived laboratory situations. The  $a$  and  $b$  coefficients are 48.00 ms and 145.84 ms/bits, respectively. Here the non-zero value of parameter  $a$  is partially due to fact that Fitts' law does not apply to movements with very low  $ID$ s [29].

However, without averaging, Fitts' law has a poor fit. Given an actual raw movement time  $T_e$ , we define *percentage error* to measure the relative deviation

<sup>3</sup>The correlation coefficient can actually range from -1 to 1; an  $r$  value close to -1.0 would mean the data perfectly fits a line with negative slope (i.e., an inverse relationship). All data discussed in this work has a positive slope (i.e., a direct relationship).

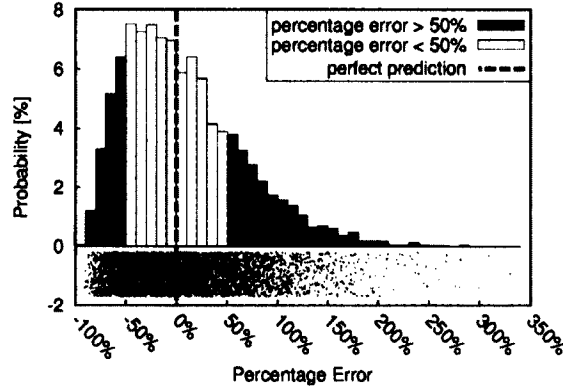


Figure 4.4: Probability distribution of  $E_{\%}$ , the relative deviation from Fitts' prediction, from raw movement times of natural web browsing. Histogram bars denote the portion of  $E_{\%}$  with values in the corresponding block. Below the histogram is the scattering of raw data, where the location of dense area (between -50% and +50%) indicates the most probable values of  $E_{\%}$ .

to the Fitts' prediction  $MT$ :

$$E_{\%} = 100\% \times \frac{T_e - MT}{MT}. \quad (4.1)$$

A perfect prediction from Fitts' law corresponds to  $E_{\%} = 0$ . The larger the absolute value of  $E_{\%}$  is, the more  $T_e$  deviates from Fitts' prediction. Figure 4.4 shows the probability density of percentage error  $E_{\%}$  (defined in Eq. 4.1), with the scattering of raw data in the lower panel. The vertical line at  $E_{\%} = 0\%$  (means zero error) indicates a perfect prediction. The percentage errors from Fitts' law to raw data span from -100% to 350%, and are highly concentrated between -50% and +50%. This wide range of scattering in terms of percentage error demonstrate, when applied to natural browsing, Fitts' prediction is not very accurate and a proper error model should be taken into account.

To measure the average deviation from Fitts' model, we define the *mean absolute percentage error* (MAPE) as:

$$M = \frac{100\%}{n} \sum_{i=1}^n \frac{T_{e,i} - MT_i}{MT_i}, \quad (4.2)$$

where  $n$  is the number of data points,  $T_{e,i}$  is the  $i$ th actual movement time, and  $MT_i$  is the Fitts' prediction corresponding to the  $ID$  of  $i$ th data point. By averaging



all 5,270 raw records, we have an MAPE of 46.40%. This is a significant deviation and cannot be ignored. In other words, when estimating user movement time on webpages using Fitts' model, one should take into account a relative error of around  $\pm 46.40\%$ .

Of course, the above deviation from Fitts' model are all based on *raw* data points, not *averaged* ones as in Fitts' original definition. And from a practical point of view, if one intends to utilize Fitts' law as a model on raw data, it is crucial to ask: how should raw data be clustered and averaged? How would the size of clusters affect the accuracy of Fitts' law?

As in our case  $ID$  is continuous, we have to partition the observations along the  $ID$  axis before averaging raw movement time  $T_e$ s. The raw data (represented as pairs  $\langle ID, T_e \rangle$ ) are clustered and averaged by groups of size  $S$ , ranging from 1 to 50. For instance, a group size of 10 means every 10 adjacent records in the as-sorted data are averaged. (In Evans and Wobbrock's work [38], a more sophisticated clustering technique is employed; by contrast, as our goal is not to produce laboratory-quality results, but to verify if Fitts' law works well in web browsing, we choose a relatively straightforward and simple clustering method.) Intuitively, when more data are included for averaging in each group, the closer the averaged data should be to the Fitts' predicted values, which leads to a smaller mean absolute percentage error (MAPE). Figure 4.5 plots MAPE as a function of cluster size, with its value from 1 to 50. It confirms our expectation, and furthermore, MAPE drops with an exponential rate with the increasing number of actions in a cluster. Note that with a cluster size of above 40, MAPE stabilizes at around 10%.

### 4.3.2 Error model

In fact, the large deviation of raw movement time from Fitts' prediction is mentioned in many previous studies [20,35] under laboratory settings, but few of them elaborate in details on this issue. And it is learned that the deviation from Fitts'

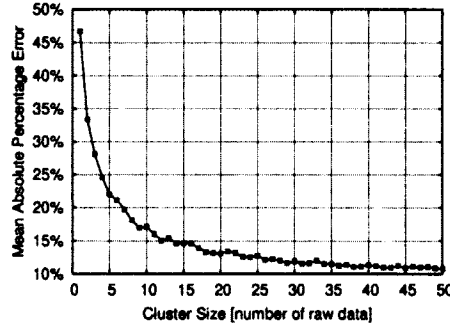


Figure 4.5: Mean absolute percentage error (MAPE) drops exponentially with the increase of cluster size. A cluster size  $S$  means every  $S$  raw data are clustered and averaged. Note that above 40 actions per cluster, MAPE stabilizes at around 10%.

model is due to endpoint variability in human movement. More specifically, the spread of hits in aimed movement forms a Gaussian distribution about the target center [29, 39, 47, 70, 106, 114].

An *error* in aimed pointings occurs when a user misses a target. Errors are the outcome of endpoint variability. An error model aims to predict error rates – the probability of missing a target – given the index of difficulty  $ID$  and the movement time. Intuitively, a higher  $ID$  (meaning a more difficult pointing task) and a shorter movement time (meaning a faster pointing action), lead to a higher error rate.

In particular, Wobbrock *et al.* [107] derived an error model from Fitts' law itself, based on the assumption that distance from endpoint to target center forms a Gaussian distribution. They evaluated their error model through a series of controlled experiments, and it is proved to be a very well fit. By employing the “effective movement time” from [106, 107], Fitts' law (Eq. 1.1) can be rewritten as

$$T_e = a + b \cdot \log_2 \left( \frac{A}{W_e} + 1 \right), \quad (4.3)$$

where  $W_e$ , is the *effective target width* coincident with the *actual* (unaveraged) movement time  $T_e$ . Solving for  $W_e$ , we have

$$W_e = \frac{A}{2^{\frac{T_e - a}{b}} - 1}. \quad (4.4)$$

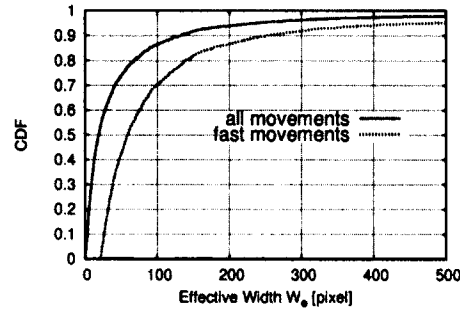


Figure 4.6: CDF for the effective width  $W_e$  for all movements and for the fast movements. Note that both distributions are not Gaussian, as were assumed in prior works.

Introducing  $W_e$  allows for mathematically growing or shrinking the effective target, to correspond to a *actual* movement time. Under the heuristic that  $W_e$  follows a normal distribution about the real target width, Wobbrock *et al.* are able to derive an accurate error model for laboratory controlled movements.

In this chapter, we intentionally follow Wobbrock's error model as a guideline to examine if pointing actions "in the wild" can be interpreted with the same error model in laboratory studies. And we find that, different from laboratory results, in a natural browsing environment, fast movements have a different error model from slower movements. To further understand the high uncertainty of the actual movement time  $T_e$  with respect to Fitts' prediction (shown in Figure 4.4), we define three categories of movements: one is with better pointing performance (i.e., shorter movement time) than Fitts' prediction (*fast movements*), one is with nominal performance comparative to Fitts' prediction (*medium movements*), and the third one with worse performance than Fitts' prediction (*slow movements*).<sup>4</sup> They are defined as follows:

$$\left\{ \begin{array}{ll} T_{\%} < 0.9 & \rightarrow \text{fast movements;} \\ 0.9 \leq T_{\%} \leq 1.1 & \rightarrow \text{medium movements;} \\ T_{\%} > 1.1 & \rightarrow \text{slow movements.} \end{array} \right.$$

<sup>4</sup>This is in the same spirit with Card, Moran & Newell's division of three imaginary humans according to their HCI performance: Fastman, Slowman and Middleman [16].

Our choice of the boundary values above is based on the human users' perception of fast, medium, and slow movements. Given the near-perfect fit on the average movement time (refer to Fig. 4.3), the Fitts' predicted value is set as the nominal case. Thus, fast movement corresponds to moving towards a target faster than Fitts' prediction (better pointing performance); slow movement is with a slower average speed than Fitts' prediction (worse pointing performance); and medium movement is the average case in accordance with Fitts' prediction (typical pointing performance). And our intention of dividing into the three categories is to explore how the range of deviation from Fitts' law affects their error models. Of all 5,084 raw data records, there are 45.44% fast movements, 12.84% medium movements, and 41.72% slow movements. Faster and slower movements have distinct nature, because of the difference in *closed-loop* and *open-loop* movements [47, 70, 107]. An aimed movement by human consists of multiple loops of feedback from neural system (to determine the next submovement) and constant motor fine-tuning. Open-loop movements are without fine-tuning, while close-loop movements are in contrast with careful fine-tuning. Setting Fitts' prediction as a standard bar, we speculate that faster movements are more prone to open-looped.

For each of the categories, we calculate the effective width  $W_e$  by Eq. 4.4 and examine its statistical properties. We apply the values of parameters  $a$  and  $b$  we get in Section 4.3.1. Note that in Eq. 4.4, for those records with  $T_e < a$ , which implies a very quick point-and-click,  $W_e < 0$ . We first filter out those records with a negative  $W_e$ , which represent 3.53% of all data. Figure 4.6 plots the distribution of effective width  $W_e$ . As we can see from the CDF curve for *all* movements, it does *not* readily follow a normal distribution as prior works assumed. The same applies to fast movements as one category. Figure 4.7 shows the distributions for slow and medium movements, which are Gaussian-shaped, with medium movements more Gaussian-like. Overall, it is clear that fast movements follow a completely

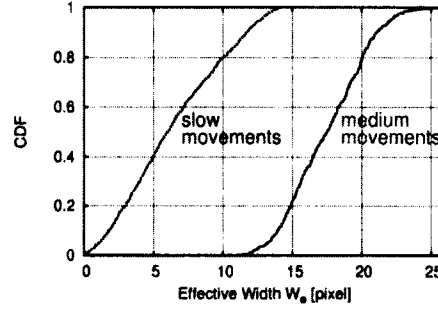


Figure 4.7: CDF for the effective width  $W_e$  for the medium and slow movements.

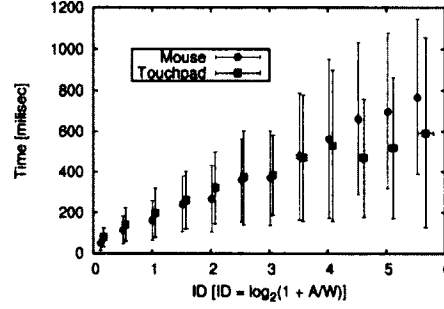


Figure 4.8: ID vs. Mean Time plotted for all collected physical mouse traces and all touchpad traces. Each data point in the figure is a result of averaging 180 raw data records. Note that the linear correlation is similar for both devices (over 98% in both cases).

different statistical model from other two categories, which implies a different error model.

The derivation of the Fitts' law error model from Wobbrock *et al.* [107] then becomes:

$$P(E) = 1 - \int_{-z}^{+z} f(x)dx,$$

where

$$z = 2.066 \frac{W}{A} \left( 2^{\frac{T_E - a}{b}} - 1 \right),$$

and  $f(x)$  is the probability density function of the distance from target center, which is not necessarily a normal distribution.

Regarding the observed distinction between faster and slower movements, it is evident that Fitts' law does not model fast movements well. Rather, faster movements tend to be dominated by initial impulse, which is more fit into Schmidt's law [92].

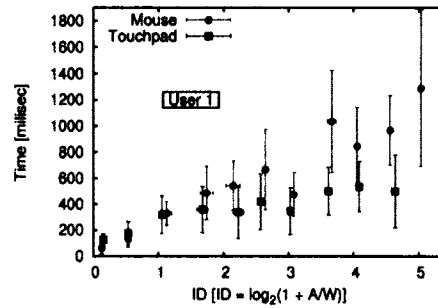


Figure 4.9: ID vs. Mean Time plotted for User 1's physical mouse trace and touchpad mouse trace. Each data point in the figure is a result of averaging 10 raw data records.

### 4.3.3 Effect of Pointing Device

How does the choice of pointing device affect Fitts' law? While many early Fitts' law studies were performed using a stylus [40], recent studies have used many other pointing devices, including physical mice [15]. Does this choice affect the accuracy of the Fitts model? What about the use of a laptop touchpad? To answer this question, we evaluate the linearity of the second data set, containing both physical mouse and touchpad traces from across 10 people, for a total of 8 mouse traces and 7 touchpad traces. Figure 4.8 shows the combined physical mouse data plotted alongside the combined touchpad data. The physical mouse data show a linear correlation of 99.71%, a nearly perfect linear fit. The touchpad data matches the physical mouse data for low ID pointing tasks, but diverges somewhat for high ID pointing tasks. However, its  $r$  value still shows a 98.04% linear correlation, strongly suggesting that Fitts' law accurately models pointing tasks with either type of pointing device. The close similarity between their correlation coefficients indicates that the choice of pointing device does not impact the law's applicability – again, showing that Fitts' law is robust to different real-world environments.

The divergence of touchpad data at high ID values implies that, in reaching for a distant hyperlink (which leads to a high ID), touchpad is different from a traditional mouse. In the case of using touchpad, a user tend to make multiple

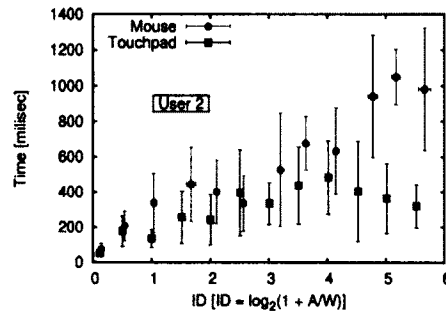


Figure 4.10: ID vs. Mean Time plotted for User 2's physical mouse trace and touchpad trace. Each data point in the figure is a result of averaging 10 raw data records.

swipes before reaching a distant target. While using a traditional mouse would usually be completed in a single swipe. In the touchpad data, what we have recorded as "continuous movement" is actually the last swipe, in which the user is likely to speed up, and that makes MT smaller than Fitts predicted.

One can also compare the traces recorded from one individual on two different pointing devices, for those users who recorded both physical mouse and touchpad data. There are three such users. Figure 4.9 shows the results of plotting User 1's mouse and touchpad traces. The relationship shows a similar trend as in the combined case — the data matches at low ID values but diverges at high ID values. However, the data still appears to be mostly linear.

Figure 4.10 shows the results of plotting User 2's single mouse trace and touchpad trace. Here, the mouse trace is linear as expected, but touchpad trace diverges greatly, even showing a downward nonlinear curve at high ID values. This shows that even for a single user, environmental factors can cause a difference in pointing actions from trace to trace, but when averaged the pointing actions all fit the Fitts model.

Figures 4.11, 4.12 and 4.13 show the results of plotting User 3's single mouse trace and two touchpad traces. All of this user's data follows a tightly correlated strong linear relationship. This shows that it is possible for a user's pointing actions using a touchpad to very closely match his or her pointing actions using a physical mouse. Thus, the use of different pointing devices does not break the

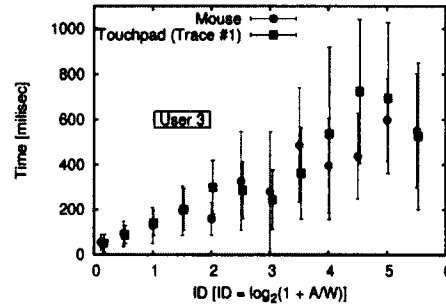


Figure 4.11: ID vs. Mean Time plotted for User 3's physical mouse trace and touchpad trace #1. Each data point in the figure is a result of averaging 10 raw data records.

#### Fitts model.

Note that the two users in our measurement above point faster with touchpad than mouse (refer to Figs. 4.9 and 4.10). This somewhat deviates from the observation of previous works [37, 71], where users point slightly faster with mouse than touchpad. Two reasons might apply here. Firstly, in our measurement, a user's movements are confined in the browser window. As touchpad is a friendly environment for short-range movement (i.e., one finger stroke), it is no surprise to see some faster pointings with touchpad than mouse within the browser window. Secondly, both touchpad and mouse devices have been greatly improved than two decades ago (when previous experiments [37, 71] are done), and thus it is quite possible that the observation made two decades ago does not hold nowadays in some scenarios.

### 4.3.4 Standard Deviation of Movement Time

The Fitts' law formula describes the mean time to complete a pointing action, and thus Fitts' law research in general focuses mainly on metrics involving the mean. However, a model is not defined solely by its mean value; there are also other considerations to take into account. For the purposes of this work, we focus specifically on the standard deviation (and, by extension, the variance) of the time to complete a pointing action given by the Fitts' model under a natural web



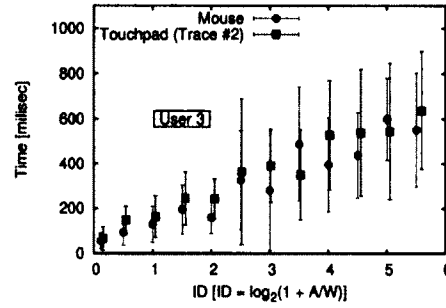


Figure 4.12: ID vs. Mean Time plotted for User 3's physical mouse trace and touchpad trace #2. Each data point in the figure is a result of averaging 10 raw data records.

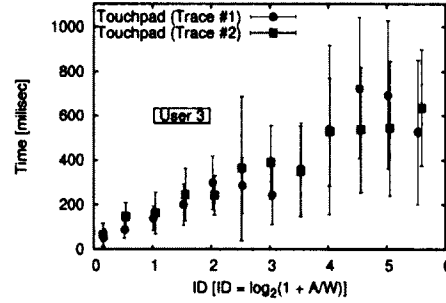


Figure 4.13: ID vs. Mean Time plotted for User 3's touchpad traces #1 and #2. Each data point in the figure is a result of averaging 10 raw data records.

browsing environment.

To analyze the variance present in the Fitts model, we perform the same Fitts' law calculations on the first data set containing 1,047 forum users' traces. This time, however, we plot the *ID* of the pointing task versus the *standard deviation* of the time to complete the pointing action, rather than the *mean* time to point. Figure 4.14 shows the results. Clearly, Fitts' law describes not only a linear relationship between *ID* and the mean pointing time, but also a linear relationship between *ID* and the standard deviation of pointing time. In other words, the higher the *ID* of a pointing task, the more variance there is in the time it takes different users to complete such a task. This can be explained by signal-dependent noise in human neuromotor systems [50], which increases with the growth of control signal strength.

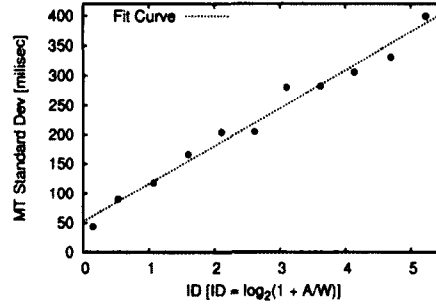


Figure 4.14: ID vs. Standard Deviation of Fitts' law calculations for the forum users' data set. Note the linear relationship.

User	Gender	$a$ [ms]	$b$ [ms/bits]	Correlation		Movement Categories			Device
				$R^2$	MAPE	Slow	Medium	Fast	
# 1	Male	6.28	117.14	0.933	41.73%	41.33%	13.68%	44.99%	Mouse
# 2	Male	-119.08	125.13	0.933	28.15%	40.36%	13.11%	46.53%	Mouse
# 3	Male	-21.68	242.52	0.936	33.24%	43.80%	14.60%	41.61%	Mouse
# 4	Male	17.77	109.73	0.933	42.59%	41.54%	13.03%	45.42%	Mouse
# 5	Female	50.52	118.74	0.963	42.59%	41.54%	13.03%	45.42%	Mouse

Table 4.1: Fitts' Law Related Parameters for Users of Mouse

## 4.4 Discussion

In this section, we discuss four related issues of Fitts' law in the context of web browsing: firstly, we derive an analytical estimate on the inaccuracy in our distance measurement, and further show that the caused inaccuracy is minor overall; secondly, in acknowledge of the difference between web browsing and traditional Fitts' law experiments, we provide a guideline on how to apply Fitts' law to web browsing; thirdly, as touchscreen (on mobile phones and tablets) has become increasingly popular for web browsing, we discuss if it is possible to extend Fitts' law to touchscreens; lastly, we present the Fitts related parameters for certain users, and preliminarily explore the possibility of user profiling.

### 4.4.1 Inaccuracy on Distance Measure

In our experiment, the distance to the target ( $A$  in Eq. 1.1) is measured as the distance from beginning of the continuous movement to the clicked endpoint. However, ideally  $A$  should be the distance from starting position to the *center*

of the target. Due to human movement variance, the clicked endpoint could not be exactly at the target center. In fact, it has been found in previous works [29,39, 47,70,106,114] that distance of the endpoint to the target center forms a normal distribution. Therefore, in our measurement, inaccuracy on the true distance  $A$  occurs especially when clicking on long hyperlinks. Figure 4.15 shows that, when clicking on a very long link, a user could either take path #1 to the left of the target center, thus rendering an underestimated distance; or he/she could take path #2, resulting in an overestimated distance. Overall, we believe that the inaccuracy on distance in our experiment is minor, since a typical webpage should be dominated by short links which only contain one or two words.

In particular, when link length is much less than the true distance to target center, the deviation of our measurement to the true distance is negligible. We denote  $D$  as the horizontal width of the target link,  $A$  as the true distance to target center, they are illustrated in Figure 4.15. For a certain *actual* click action, we denote the distance from the clicked endpoint to target center as  $d$ . Angle  $\theta$  is angle between the line of true distance and the line of actual path taken. The relative error of measured distance  $A'$  to the true distance  $A$  is

$$\delta A = \frac{\|A - A'\|}{A}. \quad (4.5)$$

From the law of cosines, we have

$$A' = \sqrt{A^2 + d^2 - 2A \cdot d \cdot \cos \theta}. \quad (4.6)$$

User	Gender	$a$ [ms]	$b$ [ms/bits]	Correlation	MAPE	Movement Categories			Device
				$R^2$		Slow	Medium	Fast	
# 6	Male	27.70	226.34	0.925	30.80%	32.91%	16.08%	51.01%	Mouse
# 6	Male	121.04	97.32	0.851	41.65%	41.64%	12.79%	45.57%	Touchpad
# 8	Female	61.57	167.51	0.931	39.98%	40.56%	17.46%	41.97%	Mouse
# 8	Female	108.64	65.83	0.678	49.01%	38.59%	10.68%	50.73%	Touchpad

Table 4.2: Fitts' Law Related Parameters for Users of Both Mouse and Touchpad

Therefore,

$$\begin{aligned}
(\delta A)^2 &= \frac{(A - A')^2}{A^2} \\
&= \frac{A^2 + (A^2 + d^2 - 2A \cdot d \cdot \cos \theta)}{A^2} \\
&\quad - \frac{2A\sqrt{A^2 + d^2 - 2A \cdot d \cdot \cos \theta}}{A^2} \\
&= 2 + \left(\frac{d}{A}\right)^2 - 2 \cdot \frac{d}{A} \cdot \cos \theta \\
&\quad - 2\sqrt{1 + \left(\frac{d}{A}\right)^2 - 2 \cdot \frac{d}{A} \cdot \cos \theta}. \tag{4.7}
\end{aligned}$$

In case of clicking on short links with a horizontal length  $D$ , we assume  $D \ll A$ . For a successful click, we have  $d < D/2$ . Thus  $D \ll A$  means  $d \ll A$ , or  $d/A \rightarrow 0$ . Define  $\varepsilon = d/A$ , Eq. 4.7 becomes

$$(\delta A)^2 = 2 - 2\sqrt{1 + \varepsilon^2 - 2\varepsilon \cdot \cos \theta} + \varepsilon^2 - 2\varepsilon \cdot \cos \theta. \tag{4.8}$$

As we can see, since  $\|\cos \theta\| \leq 1$ , when  $\varepsilon \rightarrow 0$ , which corresponds to short links,  $\varepsilon \cdot \cos \theta \rightarrow 0$  as well. Therefore, the relative error of our distance measure  $\delta A \rightarrow 0$ .

#### 4.4.2 Guidelines on Applying Fitts' Model to Web Browsing

From our experiments, we learn that the way one can apply Fitts' law to web browsing is different from what previous works describe for restricted laboratory settings. Therefore, we summarize a suggested guideline on how to apply Fitts' law model in web browsing as follows:

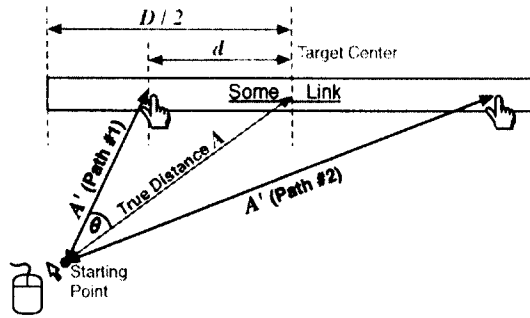


Figure 4.15: Click inaccuracy.

1. **Data Collection:** Besides x-y coordinates and timestamps, target types must be recorded as well, as it is needed to measure the target width.
2. **Clustering and Averaging:** Sort all records with increasing  $ID$ s, choose a proper cluster size  $S$  (our results show that  $S > 40$  yields optimal results), then average every  $S$  raw data.
3. **Linear Regression:** Plot the averaged  $ID$  and  $MT$  pairs, fit them in a straight line, and calculate parameters  $a$  and  $b$ . Note that they are user- and environment-specific.

#### 4.4.3 Possible Extension to Touchscreens?

Touch screen in tablets and mobile phones has become an increasingly popular device for web browsing. However, with a touch screen, to reach for a link on a webpage, users usually slide and tap. This is very different from using mice and touchpad, which always involves continuous cursor movements before clicking. Although one can do a deliberately structured Fitts' law test as in [33], we believe that there is no regular pointing behavior under natural browsing in a mobile device or tablet. And even in [33]'s Fitts' law results [34], we can see that the data points with an iPhone are very noisy, and thus lack a linear pattern, especially compared to those with mice and touchpad.

#### 4.4.4 User Profiling by Fitts' Parameters

As we mentioned in Section 1, the two constants  $a$  and  $b$  in Fitts' law formula Eq. 1.1 are affected by both the user and the environment factors. Those two constants, along with other Fitts law related parameters, such as correlation coefficient, mean absolute percentage error (MAPE), percentage of slow, medium, and fast movements, how do they differ from user to user under the same environment? And how are they affected by different environments given a same user? If the parameters are more affected by users than by environmental factors, then it would be possible to predict which user group an online user belongs to. For example, it may be possible to make a good guess about if the user online is male or female, right-handed or left-handed, age ranges, or even education backgrounds. This could be very useful for online authentication because for a certain website, such as online banking or student account, the registered users are relatively stable. Here we present preliminary results from our controlled data set, on how the parameters are affected by different users and environments.

Table 4.1 shows Fitts' law related parameters for all 5 users using mice. There are 3 male users and 2 female users. We can see that, with respect to dividing user groups, male vs. female as an example, there is no strong indication in any of the parameters for gender differences. For example, female user #5 is very similar to male user #1 in all parameters, except for parameter  $a$ ; but  $a$  alone cannot reflect gender differences as female user #4's  $a$  is very close to that of male user #1.

Which of the two factors, user or environment, plays a major role in affecting Fitts' law related parameters? To answer this question, we look at the mouse and touchpad data of the same user. Table 4.2 shows the parameters for 3 users with both mouse and touchpad data. In the table, we can see the diverging effects of changing environments for different users. For example, both user #6 and #8's

parameters are greatly affected by changing from a mouse to touchpad, while user #7' parameters are relatively stable regardless of using mouse or touchpad.

Overall, from the preliminary results, there is no definite line in dividing different user groups; and the changing of environments has different effects on different users. However, the size of our controlled data set, 10, may be too small to draw a definite conclusion. Moreover, the parameters we explored cannot capture all characteristics of human mouse movements in web browsing. It is possible that other kinetic-related metrics are able to differentiate user groups. We leave this for future works.

## 4.5 Conclusion and Future Work

This chapter examined the Fitts' model in the context of natural web browsing. Mouse movement data from over 1,000 real-world Internet users was collected via Javascript embedded on a web forum, and the analysis showed a linear relationship between the  $ID$  and  $MT$  of the task with over 98% correlation, suggesting strong evidence that Fitts' law extends well to web browsing behavior.

In addition, we evaluated the deviation in raw movement time from Fitts' predicted  $MT$ , especially the error model proposed by previous works. From the raw data, there exists a large deviation from Fitts' predicted values, with a 46.40% mean absolute deviation. We further divided all movements into three categories by the Fitts' predicted  $MT$ : slow, medium, and fast movements. And fast movements were shown to have an error model different from the other two categories, which indicates their open-looped nature.

Moreover, this chapter examined the effect of differing pointing devices on the Fitts model. Pointing data was collected from 10 people variously using physical mice and laptop touch pads. The analysis showed that both devices had a strong linear relationship between  $ID$  and  $MT$  (over 98% correlation in both cases), and

that the results were nearly identical at low  $ID$  values, yet diverged slightly at high  $ID$  values.

Finally, this chapter discussed other Fitts' Law considerations, namely the standard deviation in Fitts' Law calculations. The forum data set was analyzed and the standard deviation of  $MT$  plotted against  $ID$ . The result showed that Fitts' Law also describes a linear relationship between  $ID$  and standard deviation, implying that variance in time to point increases as  $ID$  increases.

There are a number of possible directions for our future work. We plan to conduct a large-scale user data collection, with their demographic information known. Then, we plan to further verify the possibility of classifying users into different groups (gender, age, handedness, etc) based on Fitts' law parameters. We also plan to explore the possibility to detect web bots by checking if the mouse movement pattern follows Fitts' law.



## 5 Conclusion and Future Work

The protection of an end user's identity and privacy is always the core mission of security research. In recent years, it has become increasingly critical due to prevalence of online media and mobile applications. In this dissertation, we have focused on user re-authentication by exploiting user behavior patterns and modeling. The first two projects are dedicated to user verification based on mouse movement patterns and touchscreen tapping behaviors, respectively, and the third one is on modeling web browsing behaviors from end users by Fitts' Law.

For future works, we plan to pursue in two directions. Firstly, we intend to exploit multi-finger tapplings (as opposed to single-finger tapplings in Chapter 3) for smartphone user verification. Secondly, we will conduct another research on analyzing privacy issues in Amazon wish list [4]. The rest of this chapter will detail research plans regarding these two topics specifically.

### Multi-Finger Tappings for Smartphone User Verification

As demonstrated in Chapter 3, we have seen how PIN entering process can contribute to user verification by using single-finger tapplings. In this section, we propose to extend the work in Chapter 3 by further exploiting *multi-finger* tapplings. For the purpose of smartphone user verification, enabling multi-finger tapplings has the following benefits.

First, it leads to increased key space<sup>1</sup> for entering a smartphone PIN. Under a single-finger tapping scheme, given a PIN number, a user essentially has only one way of entering his/her PIN (i.e., tapping every digit with one designated finger, e.g., index finger). In contrast, multi-finger tapping adds the possibility of using different fingers for entering the digits within one passcode. Those different finger combinations clearly form distinguishable patterns in terms of various sensor-related features (such as contact size, tapping pressure, and timings). For this very reason, a successful impostor also needs to know the specific finger combination associated with the target user. As an example to illustrate the key space under the multi-finger tapping scheme, the number of ways for entering a 4-digit passcode can be calculated as follows.

- Number of ways if using four different fingers =  $P_5^4 \times (P_4^4 + C_4^2(P_3^3 + P_2^2) + C_4^3 \times P_2^1 + 1) = 9,720$ ;
- Number of ways if using three different fingers =  $C_4^2 \times P_5^3 \times (P_4^4 + P_3^3 + C_2^1 \times C_2^1 \times P_3^3 + C_2^1 \times P_2^2 + C_2^1 \times P_2^2) = 22,320$ ;
- Number of ways if using two different fingers =  $C_4^1 \times P_5^2 \times (P_4^4 + C_3^1 \times P_3^3) + C_4^2 \times P_5^2 \times (P_4^4 + C_2^1 \times C_2^1 \times P_3^3 + C_2^1 \times P_2^2) = 9,600$ ;
- Number of ways if using one finger =  $C_5^1 \times P_4^4 = 120$ .

The final key space associated with a fixed 4-digit passcode, therefore, is  $9,720 + 22,320 + 9,600 + 120 = 41,760$ , which is more than four times larger than that using single-finger tapings.

Second, the multi-finger tapping scheme further increases the security level by exploiting more personal traits than the single-finger scheme. Specifically, it grasps the uniqueness of *all* fingers' size and flexibility from the target user, instead of just one designated finger. For example, it would be much harder for

---

<sup>1</sup>Here key space is defined as the number of distinguishable ways of entering a given PIN.

an impostor to regenerate a user's behavior involving with five different fingers than just one finger. Thus, multi-finger tapplings render a richer personal tapping profile for the target user.

However, there might be hot-spot ways on entering a certain passcode. For example, to enter 1-2-3-4, people would probably prefer to use thumb-index-middle-fourth fingers, rather than out-of-order fingers. We will explore this issue in the future.

## **Privacy Issues in Amazon Wishlist**

As social media has become increasingly more popular in recent days, it is common for users to reveal their personal information and preferences publicly online [43]. One such example is the Amazon wish list [4], which is an online gift registry. It allows users to add wished merchandise and then spread the link via emails or social networks. Because Amazon wish list is by nature to be shared with others, it is in very large extent can be seen by all online visitors<sup>2</sup>. In this project, we plan to evaluate and analyze user privacy leaks from publicly shared Amazon wish lists.

For an initial assessment, we have collected data from 46,000 users' Amazon wish lists using Java with support of jsoup library 1.7.2 [59]. Among all users, our data indicate that 40.4% of the users are publicly sharing birthday information, 33.6% are sharing location information, and 24.7% are sharing both pieces of information. Considering the the total number of active Amazon accounts is currently 237 million [99], we can see a large number of population expose their personal information {name, home town, birthday}, which are associated with most commonly used items in security questions for password recovery. Note that although fake profiles are possible, it is not likely to happen often because:

---

<sup>2</sup>except that users explicitly change its setting to "private".

(1) from a user's standpoint, the inclusion of additional personal information is to have his/her friends accurately search for the wish list in the first place; (2) those additional information are optional to enter. Therefore, we believe the above statistical numbers are of highly reliable.

For the next step, we plan to study the following aspects in data analysis and user preference mining:

- Clustering into user groups based on more specific content in their wish lists like book titles or music categories.
- Monitoring the temporal dynamics of a user's wish list throughout the year during four calendar-based seasons and holidays seasons, including Thanksgiving, Christmas/New Year, Valentine's Day, and Mother's Day, etc.
- To assess the risk of privacy leak from Amazon wish list, we plan to measure the success rate of inferring critical pieces of demographic information from users' wish lists alone. The personal information of interest includes, but not limited to, age range, education background, marital status, and children (their age, gender, etc.), where the ground truth can be obtained from users who reveal that information in "About Me".

# Bibliography

- [1] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, SIGIR '06, pages 3–10, 2006.
- [2] A. A. E. Ahmed and I. Traore. Anomaly intrusion detection based on biometrics. In *Proceedings of the 2005 IEEE Workshop on Information Assurance*, 2005.
- [3] A. A. E. Ahmed and I. Traore. A new biometric technology based on mouse dynamics. *IEEE Transactions on Dependable and Secure Computing*, 4(3):165–179, 2007.
- [4] Amazon Wishlist. <http://www.amazon.com/gp/wishlist>.
- [5] D. Amitay. Most Common iPhone Passcodes. [http://amitay.us/blog/files/most\\_common\\_iphone\\_passcodes.php](http://amitay.us/blog/files/most_common_iphone_passcodes.php). [Accessed: Nov. 2012].
- [6] J. Annet, C. W. Golby, and H. Kay. The measurement of elements in an assembly task: The information output of the human motor system. *Quarterly Journal of Experimental Psychology*, 10:1–11, 1958.
- [7] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith. Smudge attacks on smartphone touch screens. In *Proceedings of the 4th USENIX Conference on Offensive Technologies*, WOOT'10, pages 1–7, 2010.

- [8] L. Ballard, F. Monrose, and D. Lopresi. Biometric authentication revisited: Understanding the impact of wolves in sheep's clothing. In *USENIX Security Symposium*, 2006.
- [9] 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.
- [10] R. Biddle, S. Chiasson, and P. van Oorschot. Graphical passwords: Learning from the first twelve years. *ACM Computing Surveys*, 2011. (to appear).
- [11] F. Bimbot, J.-F. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-García, D. Petrovska-Delacrétaz, and D. A. Reynolds. A tutorial on text-independent speaker verification. *EURASIP J. Appl. Signal Process.*, 2004:430–451, Jan. 2004.
- [12] R. Blanch, Y. Guiard, and M. Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *ACM CHI'04*, pages 519–526. ACM, 2004.
- [13] T. Buch, A. Cotoranu, E. Jeskey, et al. An enhanced keystroke biometric system and associated studies. In *Proceedings of Student-Faculty Research Day, CSIS, Pace University*, pages C4.2–C4.7, 2008.
- [14] L. Cai and H. Chen. Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *USENIX Workshop on Hot Topics in Security (HotSec)*, 2011.
- [15] S. K. Card, W. K. English, and B. J. Burr. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics*, 21(8):601–613, 1978.

- [16] S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Publishers, 1983.
- [17] P. K. Chan. A non-invasive learning approach to building web user profiles. In *ACM SIGKDD International Conference*, 1999.
- [18] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [19] T.-Y. Chang, C.-J. Tsai, and J.-H. Lin. A graphical-based password keystroke dynamic authentication system for touch screen handheld mobile devices. *J. Syst. Softw.*, 85(5):1157–1165, May 2012.
- [20] O. Chapuis, R. Blanch, and M. Beaudouin-Lefon. Fitts' law in the wild: A field study of aimed movements. Technical Report Rapport de Recherche No 1480, LRI, December 2007.
- [21] S. Chiasson, A. Forget, E. Stobert, P. van Oorschot, and R. Biddle. Multiple password interference in text passwords and click-based graphical passwords. In *ACM Conference on Computer and Communications Security (CCS)*, 2009.
- [22] S. Chiasson, P. C. V. Oorschot, and R. Biddle. Graphical password authentication using cued click-points. In *12th European Symposium On Research In Computer Security (ESORICS)*, 2007. Springer-Verlag, 2007.
- [23] S. Chiasson, P. van Oorschot, and R. Biddle. A usability study and critique of two password managers. In *USENIX Security Symposium*, 2006.
- [24] J. Citty and D. R. Hutchings. TAPI: Touch-screen authentication using partitioned images. In *Elon University Technical Report 2010-1*, Tech Report, 2010.

- [25] N. Clarke, S. Karatzouni, and S. Furnell. Flexible and transparent user authentication for mobile devices. *Emerging Challenges for Security, Privacy and Trust*, pages 1–12, 2009.
- [26] N. L. Clarke and S. M. Furnell. Authenticating mobile phone users using keystroke analysis. *Int. J. Inf. Secur.*, 6(1):1–14, Dec. 2006.
- [27] Computing Research Association et al. Four grand challenges in trustworthy computing, 2003.
- [28] 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 the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS'11*, pages 249–259, 2011.
- [29] E. R. F. W. Crossman and P. J. Goodeve. Feedback control of hand-movement and Fitts' law. *Quarterly J. of Exp. Psychology*, 35A(2):251–178, 1963/1983.
- [30] 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 *ACM CHI'12*, pages 987–996, 2012.
- [31] M. Derawi, C. Nickel, P. Bours, and C. Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*, pages 306–311. IEEE, 2010.
- [32] H. Dillen, J. G. Phillips, and J. W. Meehan. Kinematic analysis of cursor trajectories controlled with a touchpad. *International Journal of Human-Computer Interaction*, 19(2):223–239, 2005.



- [33] J. Donovan. Fitts's Law Test. [http://jaredndonovan.com/programming/fitts\\_law/](http://jaredndonovan.com/programming/fitts_law/).
- [34] J. Donovan. Fitts's Law Test Result. [http://jaredndonovan.com/programming/fitts\\_law/graph.png](http://jaredndonovan.com/programming/fitts_law/graph.png).
- [35] H. Drewes. Only one Fitts' law formula please! In *ACM CHI'10*, pages 2813–2822, 2010.
- [36] DTREG. SVM - Support Vector Machines. <http://www.dtreg.com/svm.htm>, Feb 2011.
- [37] B. W. Epps. Comparison of six cursor control devices based on Fitts' law models. In *Proceedings of the Human Factors Society 30th Annual Meeting*, pages 327–331, 1986.
- [38] A. C. Evans and J. O. Wobbrock. Taming wild behavior: the input observer for obtaining text entry and mouse pointing measures from everyday computer use. In *ACM CHI'12*, pages 1947–1956, 2012.
- [39] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954.
- [40] P. M. Fitts and J. R. Peterson. Information capacity of discrete motor responses. *Journal of Experimental Psychology*, 67(2):103–112, 1964.
- [41] D. Florencio and C. Herley. A large scale study of web password habits. In *Proceedings of WWW 2007*, 2007.
- [42] C. Forlines, D. Vogel, and R. Balakrishnan. Hybridpointing: fluid switching between absolute and relative pointing with a direct input device. In *ACM UIST'06*, pages 211–220. ACM, 2006.

- [43] D. Frankowski, D. Cosley, S. Sen, L. Terveen, and J. Riedl. You are what you say: Privacy risks of public mentions. In *SIGIR'06*, pages 565–572, 2006.
- [44] K. Z. Gajos, K. Reinecke, and C. Herrmann. Accurate measurements of pointing performance from in situ observations. In *ACM CHI'12*, 2012.
- [45] H. Gamboa and A. Fred. A behavioral biometric system based on human-computer interaction. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 5404, pages 381–392, Aug. 2004.
- [46] M. Goel, J. Wobbrock, and S. Patel. Gripsense: using built-in sensors to detect hand posture and pressure on commodity mobile phones. In *UIST'12*, pages 545–554, 2012.
- [47] T. Grossman and R. Balakrishnan. A probabilistic approach to modeling two-dimensional pointing. *ACM Transactions on Computer-Human Interaction*, 12(3):435–459, 2005.
- [48] J. Guerra-Casanova, C. Sanchez-Avila, G. Bailador, and A. de Santos Sierra. Authentication in mobile devices through hand gesture recognition. *International Journal of Information Security*, 11(2):65–83, Apr. 2012.
- [49] P. Gupta, S. Ravi, A. Raghunathan, and N. K. Jha. Efficient fingerprint-based user authentication for embedded systems. In *Proceedings of the 42nd annual Design Automation Conference (DAC)*, pages 244–247, 2005.
- [50] C. M. Harris and D. M. Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394:780–784, 1998.
- [51] J. Hu, H.-J. Zeng, H. Li, C. Niu, and Z. Chen. Demographic prediction based on user's browsing behavior. In *Proceedings of the 16th international conference on World Wide Web, WWW '07*, pages 151–160, 2007.

- [52] J. Huang, R. W. White, and S. Dumais. No clicks, no problem: using cursor movements to understand and improve search. In *ACM CHI'11*, pages 1225–1234, 2011.
- [53] A. Hurst, J. Mankoff, and S. E. Hudson. Understanding pointing problems in real world computing environments. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility, Assets '08*, pages 43–50, 2008.
- [54] R. J. Jagacinski, D. W. Repperger, M. S. Moran, S. L. Ward, and B. Glass. Fitts' law and the microstructure of rapid discrete movements. *Journal of Experimental Psychology: Human Perception and Performance*, 6:309–320, 1980.
- [55] A. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):4 – 20, jan. 2004.
- [56] I. Jermyn, A. Mayer, F. Monroe, M. K. Reiter, and A. D. Rubin. The design and analysis of graphical passwords. In *USENIX Security Symposium*, pages 1–14, 1999.
- [57] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of European Conference on Machine Learning*, pages 137–142, 1998.
- [58] Z. Jorgensen and T. Yu. On mouse dynamics as a behavioral biometric for authentication. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS '11*, pages 476–482, 2011.
- [59] jsoup: Java HTML Parser. <http://jsoup.org>.

- [60] S. Karatzouni and N. Clarke. Keystroke analysis for thumb-based keyboards on mobile devices. *New Approaches for Security, Privacy and Trust in Complex Environments*, pages 253–263, 2007.
- [61] S. W. Keele and M. I. Posner. Processing visual feedback in rapid movement. *Journal of Experimental Psychology*, 77:155–158, 1968.
- [62] B. A. Kerr and G. D. Langolf. Speed of aiming movements. *Quarterly Journal of Experimental Psychology*, 29:475–481, 1977.
- [63] R. Kerr. Movement time in an underwater environment. *Journal of Motor Behavior*, 5:175–178, 1973.
- [64] K. Killourhy and R. Maxion. Why did my detector do that?!: predicting keystroke-dynamics error rates. In *Proceedings of RAID'10*, pages 256–276, 2010.
- [65] K. S. Killourhy and R. A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *IEEE DSN'09*, pages 125–134, 2009.
- [66] D. Kim, P. Dunphy, P. Briggs, J. Hook, J. W. Nicholson, J. Nicholson, and P. Olivier. Multi-touch authentication on tabletops. In *ACM CHI'10*, pages 1093–1102, 2010.
- [67] U. Kukreja, W. E. Stevenson, and F. E. Ritter. RUI: Recording user input from interfaces under Windows and Mac OS X. *Behavior Research Methods*, 38(4):656–659, 2006.
- [68] R. Kumar and A. Tomkins. A characterization of online browsing behavior. In *Proceedings of the 19th international conference on World Wide Web, WWW '10*, pages 561–570, 2010.

- [69] G. D. Langolf, D. B. Chaffin, and J. A. Foulke. An investigation of Fitts' law using a wide range of movement amplitudes. *Journal of Motor Behavior*, 8:113–128, 1976.
- [70] I. S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7:91–139, 1992.
- [71] I. S. MacKenzie, A. Sellen, and W. Buxton. A comparison of input devices in elemental pointing and dragging tasks. In *ACM CHI'91*, pages 161–166, 1991.
- [72] C. Mallauran, J.-L. Dugelay, F. Perronnin, and C. Garcia. Online face detection and user authentication. In *Proceedings of the 13th annual ACM international conference on Multimedia (MM)*, pages 219–220, 2005.
- [73] R. A. Maxion and K. S. Killourhy. Keystroke biometrics with number-pad input. In *IEEE DSN'10*, pages 201–210, 2010.
- [74] D. E. Meyer, R. A. Abrams, S. Kornblum, C. E. Wright, and J. E. K. Smith. Optimality and human motor performance: Ideal control of rapid aimed movements. *Psychological Review*, 95:340–370, 1988.
- [75] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury. Tapprints: your finger taps have fingerprints. In *ACM MobiSys'12*, 2012.
- [76] F. Monroe, M. K. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. In *ACM Conference on Computer and Communications Security (CCS)*, pages 73–82, 1999.
- [77] F. Monroe and A. D. Rubin. Authentication via keystroke dynamics. In *ACM Conference on Computer and Communications Security (CCS)*, pages 48–56, 1997.

- [78] Y. Nakkabi, I. Traore, and A. A. E. Ahmed. Improving mouse dynamics biometric performance using variance reduction via extractors with separate features. *IEEE Transactions on Systems, Man, and Cybernetics*, 40(6):1345–1353, 2010.
- [79] M. Nauman and T. Ali. Token: Trustable keystroke-based authentication for web-based applications on smartphones. *Information security and assurance*, pages 286–297, 2010.
- [80] C. Nickel, M. Derawi, P. Bours, and C. Busch. Scenario test for accelerometer-based biometric gait recognition. In *3rd International Workshop on Security and Communication Networks (IWSCN 2011)*, 2011.
- [81] P. H. Oliver Chapelle. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055 – 1064, sep. 1999.
- [82] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. Accessory: password inference using accelerometers on smartphones. In *HotMobile’12*, pages 9:1–9:6, 2012.
- [83] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of the International Conference on Computer Vision*, 1998.
- [84] A. Peacock, X. Ke, and M. Wilkerson. Typing patterns: A key to user identification. *IEEE Security and Privacy*, 2(5):40–47, 2004.
- [85] J. G. Phillips and T. J. Triggs. Characteristics of cursor trajectories controlled by the computer mouse. *Ergonomics*, 44:527–536, 2001.
- [86] P. Phillips, J. Beveridge, B. Draper, G. Givens, A. O’Toole, D. Bolme, J. Dunlop, Y. M. Lui, H. Sahibzada, and S. Weimer. An introduction to the good,

- the bad, & the ugly face recognition challenge problem. In *Automatic Face Gesture Recognition and Workshops (FG 2011)*, 2011 IEEE International Conference on, pages 346–353, march 2011.
- [87] P. J. Phillips. Support vector machines applied to face recognition. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 803–809, 1999.
  - [88] M. Pusara and C. E. 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, 2004.
  - [89] D. A. Rosenbaum. *Human Motor Control*. Academic Press, Inc., 1991.
  - [90] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. Mitchell. Stronger password authentication using browser extensions. In *USENIX Security Symposium*, 2005.
  - [91] N. Sae-Bae, K. Ahmed, K. Isbister, and N. Memon. Biometric-rich gestures: a novel approach to authentication on multi-touch devices. In *ACM CHI'12*, pages 977–986, 2012.
  - [92] R. A. Schmidt, H. N. Zelaznik, B. Hawkins, J. S. Frank, and J. T. J. Quinn. Motor output variability: A theory for the accuracy of rapid motor acts. *Psychological Review*, 86:415–451, 1979.
  - [93] D. Schulz. Mouse curve biometrics. In *Biometric Consortium Conference, 2006 Biometrics Symposium*, pages 1–6. IEEE, 2006.
  - [94] Y. Skadberg and J. R. Kimmel. Visitors' flow experience while browsing a web site: its measurement, contributing factors and consequences. *Computers in Human Behavior*, 20(3):403–422, 2004.

- [95] H. Slijper, J. Richter, E. Over, J. Smeets, and M. Frens. Statistics predict kinematics of hand movements during everyday activity. *Journal of Motor Behavior*, 41:3–9, 2009.
- [96] R. W. Soukoreff and I. S. MacKenzie. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in hci. *Int. J. Hum.-Comput. Stud.*, 61:751–789, December 2004.
- [97] M. Spiliopoulou. Web usage mining for web site evaluation. *Commun. ACM*, 43:127–134, August 2000.
- [98] R. C. Sprinthall. *Basic Statistical Analysis*. Prentice Hall, 2011.
- [99] Amazon.com: number of active customer accounts 2007-2013. <http://www.statista.com/statistics/237810/number-of-active-amazon-customer-accounts-worldwide/>.
- [100] E. Stobert, A. Forget, S. Chiasson, et al. Exploring usability effects of increasing security in click-based graphical passwords. In *Annual Computer Security Applications Conference (ACSAC)*, 2010.
- [101] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th international conference on World Wide Web, WWW '04*, pages 675–684, 2004.
- [102] S. Tong. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia*, 2001.
- [103] P. C. Van Oorschot, A. Salehi-Abari, and J. Thorpe. Purely automated attacks on passpoints-style graphical passwords. *Trans. Info. For. Sec.*, 5(3):393–405, Sept. 2010.
- [104] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.



- [105] V. N. Vladimir. *The Nature of Statistical Learning Theory*. Springer, Berlin Heidelberg, New York, 1995.
- [106] A. T. Welford. *Fundamentals of Skill*. Methuen, London, 1968.
- [107] J. O. Wobbrock, E. Cutrell, S. Harada, and I. S. MacKenzie. An error model for pointing based on Fitts' law. In *ACM CHI'08*, pages 1613–1622, 2008.
- [108] J. O. Wobbrock, K. Shinohara, and A. Jansen. The effects of task dimensionality, endpoint deviation, throughput calculation, and experiment design on pointing measures and models. In *ACM CHI'11*, pages 1639–1648, 2011.
- [109] Z. Xu, K. Bai, and S. Zhu. Taplogger: inferring user inputs on smartphone touchscreens using on-board motion sensors. In *WISEC'12*, pages 113–124, 2012.
- [110] R. V. Yampolskiy and V. Govindaraju. Behavioural biometrics: a survey and classification. *Int. J. Biometrics*, 1(1):81–113, 2008.
- [111] H. Yang and X. Xu. Bias towards regular configuration in 2D pointing. In *ACM CHI'10*, pages 1391–1400, 2010.
- [112] S. Zahid, M. Shahzad, S. A. Khayam, and M. Farooq. Keystroke-based user identification on smart phones. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID'09, pages 224–243, 2009.
- [113] S. Zhai. Characterizing computer input with Fitts' law parameters: the information and non-information aspects of pointing. *Int. J. Hum.-Comput. Stud.*, 61:791–809, December 2004.

- [114] S. Zhai, J. Kong, and X. Ren. Speed-accuracy tradeoff in Fitts' law tasks—on the equivalency of actual and nominal pointing precision. *Int'l J. of Human-Computer Studies*, 61(6):823–856, 2004.
- [115] Y. Zhang, F. Monroe, and M. K. Reiter. The security of modern password expiration: An algorithmic framework and empirical analysis. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.
- [116] N. Zheng, A. Paloski, and H. Wang. An efficient user verification system via mouse movements. In *ACM CCS'11*, pages 139–150, 2011.