SECURITY ANALYSIS OF GESTURE PASSWORDS

by

CAN LIU

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Janne Lindqvist

And approved by

New Brunswick, New Jersey

January, 2020

ABSTRACT OF THE DISSERTATION

Security Analysis of Gesture Passwords

By CAN LIU

Dissertation Director:

Janne Lindqvist

Touchscreens, the dominant input type for mobile devices, require unique authentication solutions. Gesture passwords have been proposed as an alternative ubiquitous authentication technique. Gesture authentication relies on recognition, wherein raw data is collected from user input and recognized by measuring the similarity between gestures with different algorithms. Our work analyzed the different aspects of gesture password security. First, since preprocessing in gesture recognizers is implemented to improve recognition accuracy, we examined the effects of three variables in preprocessing: location, rotation, and scale. We found that an authentication-optimal combination (location invariant, scale variant, and rotation variant) reduced the error rate by 45% on average compared to the recognition-optimal combination (all invariant). Secondly, we designed, implemented and evaluated a novel secure, robust and usable multi-expert recognizer for gesture passwords: Garda. Compared to 12 alternative approaches for building a recognizer, Garda achieved the lowest error rate (0.015) in authentication accuracy, and the lowest error rate (0.040) under imitation attacks; Garda also resisted all brute-force attacks. Furthermore, we proposed the first approach for measuring the security of gesture that includes guessing attacks that model real-world attacker behavior. Our dictionary of guessing attacks achieves a cracking rate of 48% after 10^9 guesses, which is a difference of 36 percentage points compared to the 12% cracking rate of the brute-force attack. Lastly, we quantified the security of various recognition passwords, including gestures and signatures, based on the passwords' distribution, modeling and enumerating the unseen passwords across a dataset. We compared the security of these recognition passwords to text passwords and Android unlock patterns with a partial guessing metric, a password security metric based on datasets of user-chosen passwords. We found that the baseline security of gestures and signatures is much higher than the security of Android unlock patterns.

ACKNOWLEDGEMENTS

I want to express my sincere appreciation for my Ph.D. advisor, thesis committee members, colleagues, collaborators, friends, and family. I could not have completed my Ph.D. research without their guidance, advice, and support.

First of all, I would like to express my gratitude to my advisor, Prof. Janne Lindqvist. When I joined Janne's research team, I had little experience conducting academic research in the field of human-computer interaction. Janne guided me on how to form and approach a research problem, write a paper, and present work to the general public. My knowledge and research skills increased tremendously with his guidance. I am indebted to his dedication while directing my research and to his support for my work and decisions.

I am fortunate to have collaborated with many talented colleagues. Gradeigh D. Clark is one of the most reliable people and collaborators I have met; his insights on research questions and his writing skills have helped our projects proceed smoothly and efficiently. Shridatt Sugrim has helped me with most of the statistical and mathematical problems in our different research projects. Thanks for help to: Xianyi Gao, Hua Deng, Meghan McLean, and Christos Mitropoulos, who provided feedback and suggestions for the technical difficulties during my research. More thanks to Fengpeng Yuan, Yulong Yang, and Huiqing Fu for providing me suggestions and help during my first year of research.

Furthermore, I want to sincerely thank Prof. Wade Trappe, Prof. Richard Howard, Prof.

Saman Aliari Zonouz, Prof. Waheed Bajwa, Prof. Ivan Marsic, and Dr. Bernhard Firner for serving as the committee members for my Ph.D. qualification exam, thesis proposal, and thesis defense. They have provided me with valuable suggestions for improving my research and papers. I would also like to extend a special thanks to Prof. Richard Howard for providing insightful comments and bringing new angles to my research.

Additionally, I want to express my appreciation to my friends. They have made my Ph.D. study colorful and productive; besides research, their friendship is one of the most incredible achievements in my Ph.D. career! Special thanks to Yue Gu and Hua Shang. They have helped me so much in different urgent situations, and I know they are friends that I can always trust.

Finally, I would like to thank my wife, Yixi Zhang, and my family for always understanding and supporting my life and decisions. Without them, I cannot imagine how hard it would have been to conquer the difficulties in my Ph.D. adventure. I would also give special thanks to my kids, Henry and Flora, who bring infinite happiness to our lives.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The number of threats to personal information stored on mobile devices has increased as these devices grow more important in day-to-day life. Current authentication methods present their own set of concerns: text-based passwords involve manifold problems, ranging from password reuse (Jain, Ross, & Pankanti, 2006; Florencio & Herley, 2007) to weak password selection (Bonneau, 2012b). Meanwhile, biometric methods have difficulties from revoking the authentication token (Uludag, Pankanti, Prabhakar, & Jain, 2004) to misidentification (Uludag et al., 2004) – for example, a fingerprint scanner fails to recognize a scarred finger.

Gesture passwords are a recently proposed authentication method for mobile devices, attracting interest due to their ability to overcome these usability problems (Sae-Bae, Ahmed, Isbister, & Memon, 2012; Aslan, Uhl, Meschtscherjakov, & Tscheligi, 2014; De Luca et al., 2013; Sherman et al., 2014; Y. Yang, Clark, Lindqvist, & Oulasvirta, 2016; Tian, Qu, Xu, & Wang, 2013; J. Yang, Li, & Xie, 2015; Aumi & Kratz, 2014; Wu, Konrad, & Ishwar, 2013). A gesture, in the context of this work, as well as previous work (Y. Yang et al., 2016; Sherman et al., 2014), is defined as a series of two-dimensional lines drawn on the surface of a touchscreen with one or more fingers. This gesture is then encoded as a set of

X and Y coordinates and compared to a template for authentication.

Studies of this emerging method have found it to have multiple advantages. Prior work has shown that gestures can be performed quickly – 22% faster than text passwords (Y. Yang et al., 2016), and that their memorability is unique under multi-account interference (Y. Yang et al., 2016). They possess a high amount of information content and complexity (Sherman et al., 2014), and biometric features are distinct enough between users that even users of the same gesture can be uniquely identified (Tian et al., 2013; Shahzad, Liu, & Samuel, 2013; Sae-Bae et al., 2012). Users have expressed preferences towards using gesture passwords when presented with the option over other methods (Sae-Bae et al., 2012; Y. Yang et al., 2016).

Any gesture-based authentication system needs to have an algorithm to interpret the users' gestures – a gesture recognizer (Clark & Lindqvist, 2015). Gesture recognition involves two important stages: preprocessing, and the recognizer itself. The preprocessing step is used to strip information from a gesture by controlling variables, making it easier to perform the recognition task. The recognizer is used to determine if a gesture is drawn by a particular user. The recognizers that used in prior work including from Support Vector Machine (Bo, Zhang, Li, Huang, & Wang, 2013; Hayashi, Maas, & Hong, 2014; Frank, Biedert, Ma, Martinovic, & Song, 2013) to Dynamic Time Warping (Sae-Bae et al., 2012; Aslan et al., 2014; J. Yang et al., 2015). However, the selection of these methods are mostly depend on the theoretical advantages or popularly used in previous work. Therefore, we developed a robust gesture recognizer, Garda, based on analysis and comparison of preprocessing and recognition methods among the different recognizers.

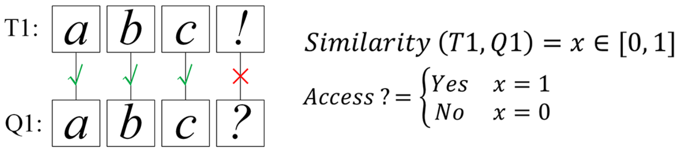
The most essential evaluation metric for an authentication system is its security. Most research on security in gesture passwords tends to focus on procedural changes to authentication systems, for example, increasing security by increasing the number of features used in authentication. Direct studies of gesture security have used information theory (Sherman et al., 2014; Y. Yang et al., 2016), shoulder surfing (Sherman et al., 2014; Aumi & Kratz, 2014; Shahzad et al., 2013; De Luca et al., 2013), or equal error rates rates between legitimate users and attackers trying to emulate the gesture (Sae-Bae et al., 2012; Aslan et al., 2014; De Luca et al., 2013; Sherman et al., 2014; Y. Yang et al., 2016; J. Yang et al., 2015; Wu et al., 2013; Aumi & Kratz, 2014). In these scenarios, the threat model is an attacker performing an observation of the gesture and trying to repeat it.

No published work on gesture security has discussed how resistant these passwords are to automated guessing attacks. This is because gesture passwords pose unique difficulties for researchers studying a key part of automated attacks: password guessability. Guessability has been the best method for modeling the guessing behavior, and refers to a metric that quantifies a given passwords security by counting how many guesses it takes an attacker to crack it (Kelley et al., 2012; Weir, Aggarwal, Collins, & Stern, 2010; Weir, Aggarwal, Medeiros, & Glodek, 2009a; Uellenbeck, Dürmuth, Wolf, & Holz, 2013). Guessability has been popularly employed to measure the security of text passwords, in studies wherein attackers use leaked dictionaries to perform attacks (Morris & Thompson, 1979; Florencio & Herley, 2007; Bonneau, 2012a, 2012b). Guessability is easier to determine, since authentication with text-based passwords is a *matching problem* that allows for the password to be exactly reproduced. But unlike text passwords, gesture passwords are defined as a *recognition problem* (Sherman et al., 2014; Clark & Lindqvist, 2015). A gesture password cannot be exactly matched: it must be identified despite slight differences in the user's input each time the password is entered. This is the main reason that automated guessing attacks have not been simulated for gestures. In our work, we conducted the first study analyzing the guessability of gesture passwords.

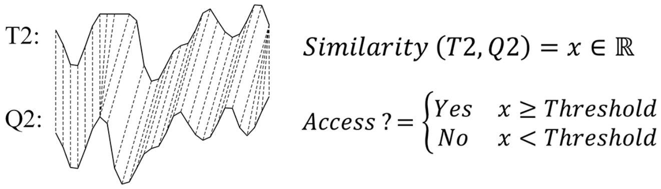
Once we evaluate the security of gesture passwords against automated guessing attacks, the next question is: how can we quantitatively compare the security of gesture passwords to the that of other dominant passwords, like text passwords, PIN, and Android pattern unlock? Gesture passwords, after all, differ significantly from these latter methods in how they operate. While classical methods of user-chosen passwords vary among themselves – patterns are graphical, while PINs use a limited 0-9 character set – each method is an example of a *matching password*, defined to be a password that can be compared directly to determine a correct entry. Mobile computing, however, has enabled the rise of a new type of password: *recognition passwords*, which derive security from time-series data and require a recognition algorithm to output a numeric measure of how correct a password attempt is. Figure 1.1 demonstrates the differences.

There are two methods often used for evaluating password security: 1) resistance against guessing attacks (Weir, Aggarwal, Medeiros, & Glodek, 2009b; Ma, Yang, Luo, & Li, 2014; Peslyak, 2017; Steube, 2017; INSIDEPRO, n.d.) and 2) evaluation of the password distribution (Uellenbeck et al., 2013; Bonneau, 2012c). The first method aligns with attacker behavior, but the efficiency depends on several factors that weaken generalization of results: attacking strategy, dictionary choices, target passwords, and size of the dictionaries and other datasets.

The second method is based on an analysis of the practical password distribution, which provides an entropic measure of password security along with an estimation of the proba-


T1:
a b c !
✓ ✓ ✓ ✗
Q1:
a b c ?
$Similarity\ (T1, Q1) = x \in [0, 1]$
$Access\ ? = \begin{cases} Yes & x = 1 \\ No & x = 0 \end{cases}$

(a) Matching Password


T2:
Q2:
$Similarity\ (T2, Q2) = x \in \mathbb{R}$
$Access\ ? = \begin{cases} Yes & x \geq Threshold \\ No & x < Threshold \end{cases}$

(b) Recognition Password

Figure 1.1: Example of matching password versus a recognition password. A matching password can only have two discrete outcomes in all cases. A recognition password outputs a real-valued number that must be compared to an authentication threshold in order to be recognized. This is typically done with algorithms like Dynamic Time Warping, as depicted in (b).

bilities of passwords not observed in the dataset. Partial guessing metric (Bonneau, 2012c), which evaluates password strength based on the user-chosen distribution, has been used to evaluate the security of Android pattern unlock (Uellenbeck et al., 2013; Song, Cho, Oh, Kim, & Huh, 2015) and text passwords (Bonneau, 2012c). These evaluations have yielded results about the size of the password space under attacker scrutiny, which allows for comparison of the security of two different authentication methods.

The challenge with adapting password distribution for recognition passwords is how to enumerate the password space. The time-series data needs to be discretized into a character set that can be used to enumerate the size of the full space, and to allowing for modeling passwords not yet seen in the dataset (Bonneau, 2012c; Uellenbeck et al., 2013). Discretization allows us to ”translate” a recognition password into matching password. This method enabled us to conduct quantitatively evaluating the security of gesture passwords against matching passwords based on the same criterion: partial guessing metric.

1.2 ORGANIZATION

The research in this thesis is organized as follows: Chapter 2 describes the related work on gesture authentication systems, password guessability, and attacks against gesture authentication. Chapter 3 introduces our study on designing a robust gesture authentication method. Chapter 4 presents our study analyzing the weak subspace and the guessability of gesture passwords. Chapter 5 presents a study on the challenges and methods on quantitatively evaluating gesture passwords based on partial guessing metric. Chapter 6 offers discussion of the presented three studies. Chapter 7 summarizes the takeaways of this thesis.

1.3 CONTRIBUTION

Our current work presents the following contributions:

1. We are the first to present the effects of preprocessing invariances on the security and usability of gesture-based authentication systems.

2. We present a comprehensive study of different approaches to implementing gesture-based authentication systems and designed a novel multi-expert recognizer: Garda.

3. We have evaluated 13 different recognizers on three criteria: 1) authentication performance (i.e. equal error rate) with five datasets; 2) imitation attacks with two datasets; 3) brute force attacks.

4. Based on the largest set of gesture passwords ever assembled in research, we present a comprehensive analysis on the weak subspace for gesture passwords.

5. We have used the weak subspace to build a novel dictionary attack method against

gesture passwords.

6. We have developed a novel method to analyze the full theoretical password space of gesture passwords and estimate the size of the weak subspace for gesture passwords.
7. We present the first attempt to quantify the security of user-chosen secrets for recognition passwords, and we compare their security to that of matching passwords based on a realistic model of attacker behavior: a partial guessing metric.
8. We discretized recognition passwords to solve their many-to-one mapping problem.
9. We estimate the baseline of the security of recognition password based on the largest available datasets.

CHAPTER 2

BACKGROUND AND RELATED WORK

We summarize previous work and discuss the relevant background of gesture-based authentication methods, password guessability, and attacks against gesture authentications.

2.1 Gesture-based Authentication Methods

There are several approaches and analyses related to gesture authentication methods. Sherman et al. (Sherman et al., 2014) introduced a novel information theoretical metric to quantify the security of free-form gestures and demonstrated a way to authenticate multi-touch gestures with Protractor (Li, 2010). They do not examine the influence of gesture variants on authentication performance. Clark and Lindqvist (Clark & Lindqvist, 2015) presented a systematic rendition of how to evaluate gesture recognition methods. Yang et al. (Y. Yang et al., 2016) studied free-form gestures and text passwords in the field and demonstrated how gestures outperform text passwords in mobile authentication. De Luca and Lindqvist (Luca & Lindqvist, 2015) gave an overview of several usability and security issues with smartphone authentication and different approaches to solve them.

Although gesture-based authentication systems can be implemented in various ways (Clark & Lindqvist, 2015), the research community (Tian et al., 2013; Aslan et al., 2014; J. Yang et al., 2015; Aumi & Kratz, 2014; Wu et al., 2013) has mostly focused on implementing recognizers with Dynamic Time Warping (DTW) (Myers & Rabiner, 1981). Most of these works omit the details of how gestures are preprocessed, which became an important point in our work, since preprocessing methods change the size of the password space for gestures. Free-form gestures are also used in a system based on cosine similarity (Sherman et al., 2014; Y. Yang et al., 2016).

Biometric features of human hands have been leveraged as pre-defined gestures to perform gesture-based authentication (Sae-Bae et al., 2012). This method has high error rates (about 10%), except with user-defined gestures. BoD shapes (De Luca et al., 2013) are 2D gestures collected on the back of a device using two phones connected back to back. Similarly, GEAT (Shahzad et al., 2013), used biometric features of a user's gesture, such as finger velocity and stroke time to distinguish different users. XSide (De Luca et al., 2014) is a stroke-based authentication mechanism that uses the front or back of smartphones.

Zheng et al. (Zheng, Bai, Huang, & Wang, 2014) designed an authentication system by recognizing a user's tapping password behavior based on a list of features including acceleration, pressure, size, and time – collected during authentication. De Luca et al. (De Luca, Hang, Brudy, Lindner, & Hussmann, 2012) also introduced an authentication scheme based on a user's touch pattern. Burgbacher et al. (Burgbacher & Hinrichs, 2014) introduced an authentication scheme based on gesture keyboards. Shirazi et al. (Sahami Shirazi, Moghadam, Ketabdar, & Schmidt, 2012) introduced a 3D magnetic gesture recognition system. Schaub (Schaub, Walch, Könings, & Weber, 2013) examined five existing graphical password schemes and found that the design space is expressive enough to capture all aspects of a graphical password.

Multi-expert systems have been applied to authentication in other contexts before. For example, Czyz (Czyz, Sadeghi, Kittler, & Vandendorpe, 2004) present a multi-expert system for face authentication with sequential fusion of scores of faces' successive video frames. The systems final decision is strengthened by several face authentication schemes. Dimauro et al. (Dimauro, Impedovo, Pirlo, & Salzo, 1997) presented a multi-expert verification system for processing bank checks. It combines three algorithms: structure-based, component-oriented approach, and a highly-adaptive neural network based method.

Finally, the HCI community has also studied free-form and user-defined gestures from different angles. Oh et al. (Oh & Findlater, 2013) found that user-defined gestures may be ambiguous so they implemented a mixed-initiative approach to improving gestures' quality. Nacenta et al. (Nacenta, Kamber, Qiang, & Kristensson, 2013) found that user-defined gestures are easier to remember than pre-designed gestures.

In summary, there has been no prior work on systematically comparing different approaches for implementing gesture recognizers for authentication. Our work contributes by implementing thirteen different approaches, comparing them with multiple datasets, and subjecting them to different attack scenarios. Similarly, we sought to address previous lack of analysis regarding the effects of preprocessing methods on the size of the gesture password space. Our work ex- plains the effects of preprocessing steps on the size of the password space, and we provide a method to evaluate the security of gesture passwords through enumerating the theoretical password space.

2.2 Password Guessability

Text-based password guessability is a well-studied problem that involves biases in the user-chosen distribution of passwords. One example of early work (Morris & Thompson, 1979) examined 3289 text passwords and found that 86% of the passwords had structural commonalities: short in length, only contained lowercase letters or digits, and were common dictionary words. More recently, a large-scale study on web passwords judged the average user password to be of poor quality and revealed overlap in user passwords across multiple accounts (Florencio & Herley, 2007). To combat the weakness of entropy, guessing entropy has been proposed to model the practical attacker behavior and quantify password security (Bonneau, 2012b). An analysis of 70 million passwords shows a clear bias in user-chosen distribution of text passwords (Bonneau, 2012b). An analysis of popular text password cracking methods reveals that quantifying password strength with a single cracking algorithm is not reliable (Ur et al., 2015). Recently, a neural network model of human-chosen passwords was proposed, which resulted in an efficient and highly compressed password guessing method (Melicher et al., 2016a). An password strength estimator - zxcvbn - can estimate the current best-known attacks with 1.5 MB compressed data and up to 10^5 guesses (Wheeler, 2016).

Outside the realm of textual passwords, researchers have also probed the guessability of graphical passwords. With Draw-A-Secret (DAS) (Jermyn, Mayer, Monrose, Reiter, & Rubin, 1999), people were found more likely to select symmetric graphics as passwords, reducing the size of the effective password space (Thorpe & van Oorschot, 2004). This was the first work to push forward the idea that the size of the graphical password space can be demonstrably reduced. Almost a decade later, Androids Pattern Unlock, itself a special case of DAS, was shown to skew the user-chosen distribution based on starting points and crossing patterns (Uellenbeck et al., 2013). Instead of grid cells, Pass-Go (Tao & Adams, 2008) used intersections on a grid as anchors, with a study showing that 49% of the passwords are alphanumeric or well-known symbols (Tao & Adams, 2008).

Besides recall-based schemes, graphical passwords based on human ability to recognize images also have weak subspaces based on user choice. Passfaces (Brostoff & Sasse, 2000) presents a panel of human faces and a user creates a password by clicking a sequence of images. A field study of Passfaces showed that users tend to select the faces of their own race and the weakest 25% of user passwords were cracked within 13 attempts (Davis, Monrose, & Reiter, 2004). Instead of using human faces, Story (Davis et al., 2004) offers the user a panel of generic images to select as their password. Although Storys user choice issue proved less severe than that of Passfaces, research revealed that it still allowed for selection preference between genders, and the weakest 25% passwords were guessed in 112 attempts (Davis et al., 2004). PassPoints (Wiedenbeck, Waters, Birget, Brodskiy, & Memon, 2005) is a cued-recall graphical password or click-based graphical password (Biddle, Chiasson, & Van Oorschot, 2012), with the major weaknesses being hotspots (van Oorschot & Thorpe, 2011) and patterns (Chiasson, Forget, Biddle, & van Oorschot, 2009). Hotspots are the image areas or points that users are more likely to choose and the latest attacks based on them using Markov models can crack 36% of the PassPoints passwords by 2^{31} guesses. (van Oorschot & Thorpe, 2011). Patterns are simple shapes that are likely to be chosen as PassPoints passwords: around 80% of PassPoints passwords fall into primarily defined simple shapes (Chiasson et al., 2009).

In summary, considerable work has shown weak subspaces for text and graphical passwords. We sought to expand this research to demonstrate and examine the weak subspace for gesture passwords.

2.3 Attacks Against Gesture Authentication

So far in the literature, attacks against gesture authentication have been performed by participants trying to emulate gestures based on a variable amount of knowledge about the gesture provided by researchers (Tian et al., 2013; Aslan et al., 2014; J. Yang et al., 2015; Aumi & Kratz, 2014; Wu et al., 2013; Sherman et al., 2014). These types of attacks are generally referred to as shoulder surfing attacks. Smudge attacks (Aviv, Gibson, Mossop, Blaze, & Smith, 2010) were used on touchscreens to identify user passwords based on oil patterns left by their fingers. Even a robot-based attack (Serwadda & Phoha, 2013) has been deployed against an touchscreen pattern authentication scheme (De Luca et al., 2012), where the robot is able to emulate human patterns of use on touchscreens. However, this robot attack does not provide guessing attacks against human-chosen secrets.

In summary: security of gesture passwords has been measured with mutual information and equal error rates. We provide a method more closely aligned with real world attacker behavior through guessing attacks. We present an efficient dictionary attack method as well as a brute-force method for attacking.

2.4 Passwords Security Evaluation

Morris and Thompson (Morris & Thompson, 1979) were the first researchers to document how user-chosen text passwords are vulnerable to dictionary attacks due to users' choice of overlapping patterns with specific meanings rather than random text strings. Dictionary attack efficiency was improved by the development of probabilistic context-free grammar (Weir et al., 2009b), John the Ripper (Peslyak, 2017), Hashcat (Steube, 2017), Markov chains (Ma et al., 2014), and neural networks (Melicher et al., 2016b). The key difference in these techniques is how the guessing attack is generated, but the idea remains the same: humans are more likely to choose from a weak subspace instead of the full space of available passwords.

Partial guessing metric has become an established metric for security analysis. Bonneau (Bonneau, 2012c) proposed partial guessing metrics to model real-world attacks, wherein attackers only crack a portion of weak passwords and give up on guessing more difficult accounts. Uellenbeck et al. (Uellenbeck et al., 2013) applied partial guessing metric to evaluate the security of Android unlock patterns and PINs. EmojiAuth (Golla, Detering, & Dürmuth, 2017) performed guessing attack based on Markov chain model to crack the Emoji-based authentication. However, they measured the guessing metric by cracking subsets of their collected data rather than through an estimation of the user-chosen distribution. Song et al. (Song et al., 2015) proposed a strength meter for Android unlock patterns and evaluated it using partial guessing metric. Aviv et al. (Aviv, Budzitowski, & Kuber, 2015) studied the impact on the security of Android unlock pattern when the grid size was increased from 3×3 to 4×4 and found that this change does not improve the security of the Android unlock patterns. Kiesel et al. (Kiesel, Stein, & Lucks, 2017) evaluated text password security using partial guessing metric for secure and mnemonic passwords.

Many studies examine the guessability and security of matching passwords. However, to the best of our knowledge, there is no existing work on analyzing the security of recognition passwords, because of their many-to-one problem for passwords. Here, we present the first attempt to quantitatively analyze recognition passwords with partial guessing metric. With partial guessing metric, we estimated the baseline of the security of recognition passwords and the recognition passwords is more secure than Android unlock patterns.

CHAPTER 3

GARDA: GESTURE AUTHENTICATION FOR MOBILE SYSTEMS

3.1 Overview of Chapter

This chapter details how we proposed and evaluated a novel multi-expert gesture recognizer for authentication: Garda. Its development allowed us to analyze how preprocessing the variables for gesture recognizers can impact their authentication performance.

First, we introduce three recognition performance metrics for evaluating the per- formance of recognizers: Equal Error Rate, brute-force attack, and imitation attack.

Next, we introduce the theoretical details of Garda and the other recognizers, as well as the datasets that we used for our analysis. The other recognizers include: Protractor, Edit Distance on Real sequence (EDR), Longest Common Subsequences (LCS), Dynamic Time Warping (DTW); Protractor-, EDR-, LCS-, DTW- kernel based SVM; discrete HMM, segment HMM, continuous HMM; and Gaussian Mixture Models.

Finally, we describe how we explored the individual and combined effects of the three invariants of preprocessing methods. We also show the result of our comparison between the 13 recognizers based on the three metrics.

3.2 Method

3.2.1 Feature Extraction

The gestures in our datasets are collected by mobile devices. Although the sampling rates and dimensions differ, gestures are consistent across their datasets, in that multiple measurements exist for each gesture, and (x, y) coordinates and timestamps appear in every dataset. These are the only features that were compared.

Recognition Performance Metrics

We use Equal Error Rate (EER) and a Receiver Operating Characteristic (ROC) curve to evaluate the performance of gesture recognizers for authentication. The EER is a point on the ROC curve at which the False Acceptance Rate (FAR), the ratio of accepted false attempts to the total number of attempts, is equivalent to the False Reject Rate (FRR), the ratio of rejected true user attempts over the total number of attempts (Fawcett, 2006). This represents a usability-security trade-off point, at which the number of rejected true attempts equals the number of attackers permitted. The ROC curve and the corresponding FAR and FRR values reflect the behavior of an authentication method with varying thresholds.

Sugrim et al. (Sugrim, Liu, McLean, & Lindqvist, 2019) proposed Frequency Count of Scores (FCS) as a complementary metric of ROC for better understanding the performance authentication systems. However, the main purpose of our work is comparing the different recognizers, FCS can not be used for directly comparison.

We performed two types of attacks to evaluate the security of the recognizers: brute-force and imitation attacks. Resistance to brute-force attacks characterizes a recognizer's resistance to random guessing while resistance to imitation attacks characterizes a recognizer's resistance to shoulder surfing type of attacks. These features correlate with the EER value of the recognizer: the lower the EER is, the stronger the method will be when distinguishing genuine and imitation gestures.

We note that previous works on free-form gestures for authentication such as Sherman et al. (Sherman et al., 2014) and Yang et al. (Y. Yang et al., 2016), have provided data on the usability and memorability of these types of gestures in the lab and the field. However, our work is focused on recognition performance in an authentication system, thus, other usability evaluations presented in the previous work are beyond our scope and we refer to Sherman et al. (Sherman et al., 2014) and Yang et al. (Y. Yang et al., 2016) for details.

Invariance Benchmark: Recognition-Optimal Combination

$-family gesture recognition schemes such as $1 (Wobbrock, Wilson, & Li, 2007), Protractor (Li, 2010), $P (Vatavu, Anthony, & Wobbrock, 2012), and $N (Anthony & Wobbrock, 2010) implement preprocessing steps for removing a gesture's rotation, scale, and location. This is done to minimize the variations in performance of the same gestures by different people and to correspondingly increase recognition accuracy. We define the choice to make these three gesture variables invariant as the *recognition-optimal* combination.

Previous free-form gesture authentication systems based on Protractor (Sherman et al., 2014; Y. Yang et al., 2016) used this recognition-optimal combination in their implementations. They demonstrated the effectiveness of distinguishing gestures from different people, and showed great ability resisting shoulder surfing attacks (Sherman et al., 2014). However, this past work does not show whether this is optimal for authentication purposes. In our analysis, we use the recognition-optimal combination as a benchmark.

3.2.2 Brute-Force Attack Method

We generated a brute-force attacks with the following steps: 1) Randomly generate two sequences for x and y. 2) Filter the two sequences using a low pass filter with a cutoff frequency at 10 Hz. Remove the first few points that are distorted by the time delay of the filter. 3) Resample the generated gesture by the sampling rate.

The choice for the 10 Hz cutoff is not arbitrary. In our analysis, gestures are resampled to the same length (256). We assume the time to perform a gesture is one second, since we focus on guessing the gesture's shape. By examining the distribution of frequencies for each gesture, we found the majority of gesture frequencies are concentrated under 10 Hz.

3.2.3 Imitation Attack Method

We used the imitation attack samples from two public datasets: SUSig (Kholmatov & Yanikoglu, 2009) and MCTY-100 (Ortega-Garcia et al., 2003). The attackers were asked to observe legitimate authentication attempts. They were also asked to practice the attacks as many times they wanted. After the attackers were confident of their imitations, they performed the attacks.

3.3 Recognition Algorithms

We examined and implemented 13 specific algorithms: Protractor, Edit Distance on Real sequence (EDR), Longest Common Subsequences (LCS), Dynamic Time Warping (DTW); Protractor-, EDR-, LCS-, DTW- kernel based SVM; discrete HMM (dHMM), segment HMM (sHMM), continuous HMM (cHMM); and Gaussian Mixture Models. The knowledge obtained with testing and evaluating these algorithms led us to implement two novel

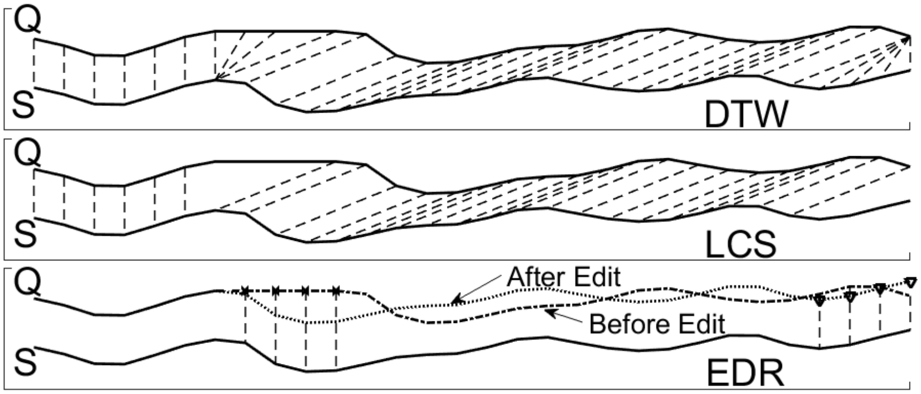


Figure 3.1: Examples of three alignment methods: DTW, LCS, and EDR. In DTW, all points in the two sequences must be matched with each other; In LCS, only the same subsequences are matched; In EDR, only the difference between two sequences are counted.

multi-expert approaches: SVMGarda and Garda.

Protractor Algorithm

Protractor (Li, 2010) is a common method used in free-form gesture authentication systems (Sherman et al., 2014; Y. Yang et al., 2016). It is one of the $-family algorithms (Wobbrock et al., 2007; Li, 2010; Vatavu et al., 2012) that uses geometric similarity between two gesture trials. Recall that there are three types of invariance in a gesture: location, scale and rotation invariance. The procedure of Protractor is: resampling a gesture to fixed points, removing the three variants (location, scale and rotation), and finding the maximum value of cosine distances between recall gesture and template gestures.

Time Series Sequence Matching

Several gesture authentication systems use DTW (Sae-Bae et al., 2012; Aslan et al., 2014; De Luca et al., 2013; J. Yang et al., 2015; Aumi & Kratz, 2014). LCS (Bergroth., Hakonen., & Raita, 2000) and EDR (Chen, Özsu, & Oria, 2005) are also similar time series sequence methods, also discussed in more detail below. Although LCS and EDR have not been used in published gesture authentication before, we incorporated them in the event that they performed with higher authentication Accuracy than DTW. Figure 3.1 gives examples of these three methods.

Dynamic Time Warping (DTW) is a dynamic programming method for aligning time sequences. Figure 3.1 shows how DTW seeks the minimum total distance between elements in sequences Q and S by finding the non-decreasing matching pairs of elements. The crucial point is that no elements are discarded through the DTW matching, despite how large the distance between a pair of the elements could be.

Longest Common Subsequences (LCS) (Bergroth. et al., 2000) is another similarity measurement approach for sequences. The LCS method (Bergroth. et al., 2000; Morse & Patel, 2007) is used to find the longest common subsequence of two sequences by elastic matching. As shown in the middle of Figure 3.1, sequence S and Q find the longest subsequence by connecting all small continuous subsequences together. This differs from DTW, which counts the distances of matched sequence points. LCS counts the percentage of matched points in the two sequences – for example, if the length of S and Q is L_S and the number of matched points between the two sequences is L_m, the LCS distance is L_m/L_S.

Edit Distance on Real sequence (EDR) (Chen et al., 2005) is a method that measures the difference between two sequences by counting the number of insert, delete, and replace operations that are needed to transform one sequence into the other. As shown in the bottom of Figure 3.1, the EDR distance of the two sequences S and Q is the number of operations that delete the four points with '$\times$' markers and four insert points with '$\bigtriangledown$'. Unlike LCS or DTW, EDR is a method that does not reward matches; rather, it penalizes gaps and mismatches.

Support Vector Machine

Support Vector Machine (SVM) (Cortes & Vapnik, 1995) is another technique used in gesture authentication systems (Bo et al., 2013; Hayashi et al., 2014; Frank et al., 2013). We used a non-linear SVM classification algorithm with a kernel function to transform the input data into a higher-dimensional feature space. The essential property of Protractor and time series sequence matching algorithms (i.e. LCS, DTW, EDR) is that they measure the similarity between gestures, meaning we can use them as kernel functions for SVM.

Gaussian Mixture Model

Gaussian Mixture Models (GMM) have been used in voice recognition (Campbell, Campbell, Reynolds, Singer, & Torres-Carrasquillo, 2006) (Campbell, Sturim, & Reynolds, 2006) (Reynolds, Quatieri, & Dunn, 2000). GMM is used to estimate any probability density function. A key feature of GMM is that it only considers the distribution of sample values of a sequence and does not consider the order of a sequence. Assume $X = \{x_1, ..., x_N\}$ is a set of sample points of gesture sequences with D-dimensional features. The Gaussian Mixture Model of X is a weighted sum of M component Gaussian densities below:

$$p(X|\lambda) = \textstyle\sum_{i=1}^{M} \omega_i p_i(X)$$

where ω_i is the mixture weight with the constraint $\sum_{i=1}^{M} \omega_i = 1$. λ is the GMM parameter, $p_i(X)$ is the component Gaussian densities, which is calculated by a mean vector μ_i, and a covariance matrix, Σ_i:

$$p_i(X) = \tfrac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(X-\mu_i)'(\Sigma_i)^{-1}(X-\mu_i)}$$

Since the $p(X|\lambda)$ is parameterized by $\omega_i, \mu_i, \Sigma_i$, we represent the GMM parameter as $\lambda = \{\omega_i, \mu_i, \Sigma_i\}, i = 1, ..., M$.

For one user's gesture password, we use expectation-maximization (EM) algorithm (Reynolds et al., 2000) to iteratively train the GMM with the template gestures, and obtain the maximum likelihood estimation of the template gesture model λ_{tmp} for the specific user.

We then calculate the similarity score between two gestures by doing:

$$Similarity(X_q) = logp(X_q|\lambda_{tmp})$$

X_q is the recall gesture and λ_{tmp} is the template gesture model.

Hidden Markov Model

Hidden Markov Model (HMM) is a stochastic model used in many types of gesture recognition systems (Lee & Kim, 1999; Elmezain, Al-Hamadi, & Michaelis, 2009; Deng & tat Tsui, 2000). There are two main directions to build an HMM for gestures.

One way is by using the directions between adjacent gesture points as observations of a HMM, and training the model accordingly (Lee & Kim, 1999). We call this a discrete HMM (dHMM), wherein we divide the directions into 16 equally-spaced arcs. The number of observations for a single gesture part is, therefore, 16. We found the seven states left-to-right no jump HMM can output the lowest EER out of many other HMM models. The transition matrix is restricted to what is seen in Figure 3.2.

The other way one can develop an HMM is by segmenting the gesture to identify the critical points as observations of HMM (Deng & tat Tsui, 2000). We label this as segmentation HMM (sHMM), wherein we segment and classify the gestures into basic parts. Afterwards, we train the HMM based on those basic gesture parts in each gesture.

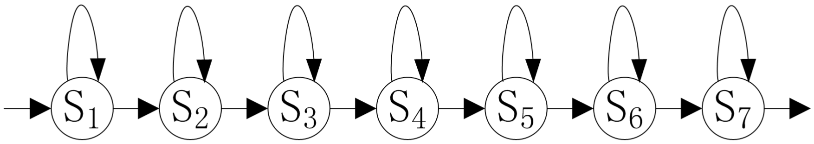


Figure 3.2: 7 States Left-to-Right No Jump HMM structure. Every state of the HMM has a self transition loop and only can move forward to next neighbour state.

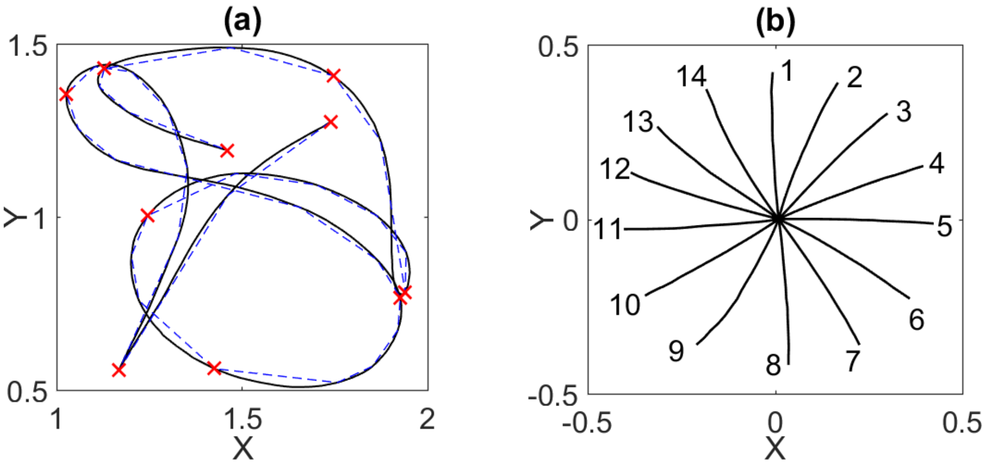


Figure 3.3: (a) Example of gesture after RDP algorithm and (b) Clustered basic gesture parts. In figure (a), the solid trajectory shows the trajectory of the original gesture, and the dashed line shows the trajectory of the remaining samples after the first round of RDP. The red cross markers ($'\times'$) are the samples after the second round of RDP. We find that the gesture samples after the first RDP are still dense and, after a second RDP, are ready to be used as splitting points. (b) Basic gestures all start from (0,0) in different directions and can be regarded as 14 observations in HMM.

Three steps constitute the building of the sHMM. 1) Segmentation: we use the Ramer-Douglas-Peucker (RDP) (Heckbert & Garland, 1997) algorithm for two rounds to reduce the number of points in the gesture. RDP is an approximation method for finding a similar curve to the original sequence using fewer points. The dashed line and '×' in Figure 3.3(a) shows the approximation gesture after the first and second round of RDP, respectively. With the approximation points '×', we segmented the gestures to several basic gesture parts. 2) Clustering. We cluster the segmented gestures into a few basic parts. We first normalize the gesture segments to the same size, then implement K-means as an unsupervised learning method to classify the basic gesture parts into several classes. We found, through observation, that K=14 was the optimal number for basic gesture types. Figure 3.3(b) shows the 14 basic gesture parts. Consequently, the gestures can be represented as combinations of these 14 basic gesture parts. 3) Train HMM. We follow the steps in dHMM to build the HMM for each gesture password from these 14 parts.

In addition to the above, we implemented a continuous HMM (cHMM). Specifically, we divided a gesture evenly into N basic gesture parts. For each basic gesture part, there are several gesture points – we modeled the probability density function of these points with GMMs. Each basic gesture part can be represented by a combination of M Gaussian distributions. By training the HMM with the Baum-Welch algorithm (Jelinek, Bahl, & Mercer, 2006) we obtain HMM verifiers for each gesture type. The Baum-Welch algorithm is a special case of the Expectation-Maximization (EM) algorithm (Arthur P. Dempster, 1977) and is used to find a locally optimal solution to the HMM training problem.

3.4 Garda Authentication Systems

In this section, we describe our novel gesture authentication system. We developed two different types of multi-expert systems. The first one, called Garda, combines Protractor and GMM. The other one, called SVMGarda, combines the Protractor kernel SVM and GMM. Figure 3.4 gives an overview of our Garda and SVMGarda authentication systems. Unlike other recognizers, which are based on one feature of recognition, Garda and SVMGarda are better because they recognize gestures in two features: the gesture shape feature (by GMM) and adjacent points feature (by Protractor or by Protractor kernel SVM).

Multi-expert (ME) classification systems work by combining the results of different

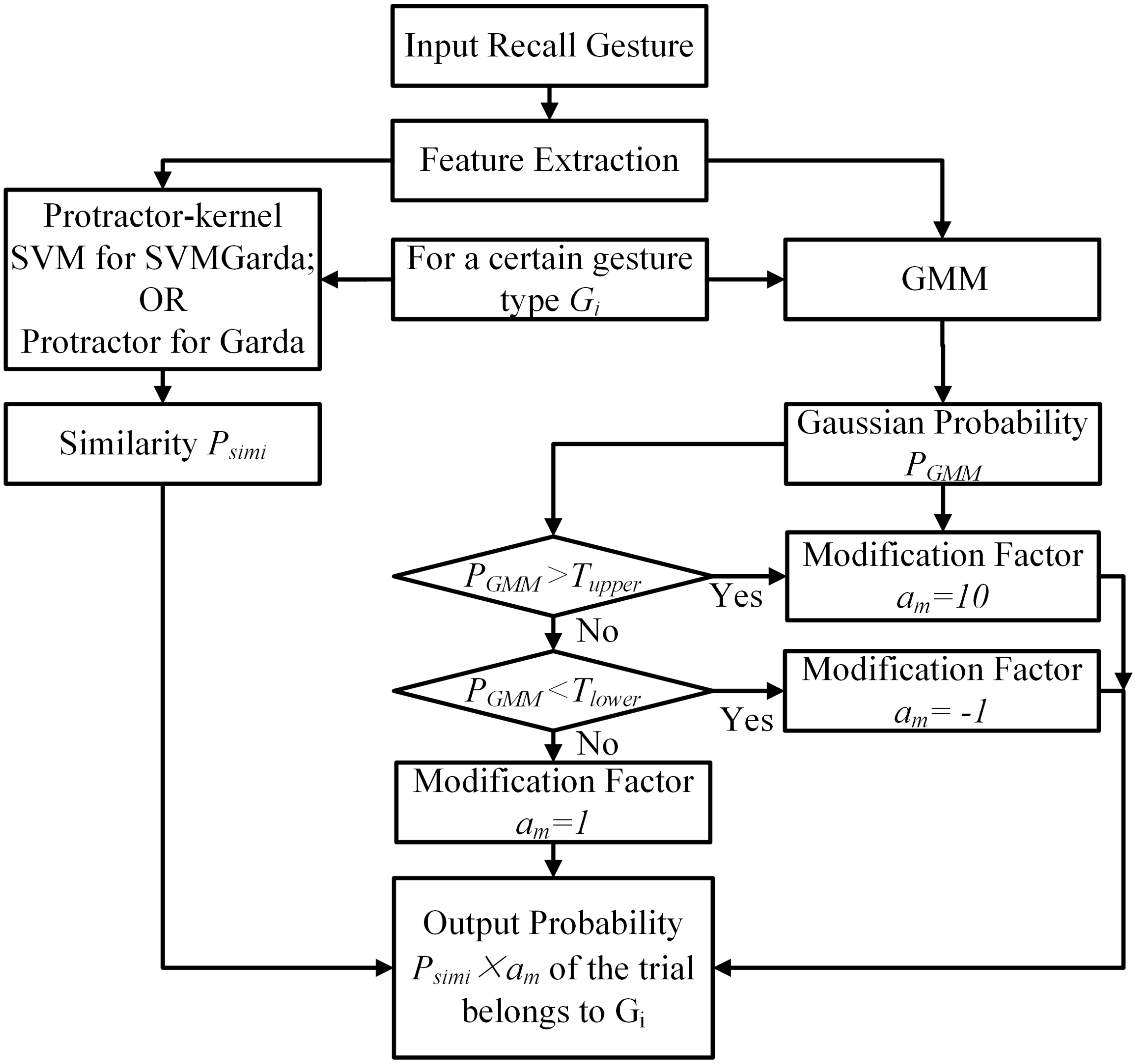

Input Recall Gesture
Feature Extraction
Protractor-kernel SVM for SVMGarda; OR Protractor for Garda
For a certain gesture type G_i
GMM
Similarity P_{simi}
Gaussian Probability P_{GMM}
$P_{GMM} > T_{upper}$
Yes
No
Modification Factor $a_m=10$
$P_{GMM} < T_{lower}$
Yes
No
Modification Factor $a_m= -1$
Modification Factor $a_m=1$
Output Probability $P_{simi} \times a_m$ of the trial belongs to G_i

Figure 3.4: SVMGarda and Garda authentication system. SVMGarda system uses the Protractor kernel SVM method in the top left part of authentication section. Garda system uses the Protractor method. The similarity of Garda between the recall and template gesture is measured by Protractor and GMM separately. The similarity output of Protractor is P_{simi}. It is modified by different factors based on the comparison result of Gaussian probability of GMM P_{GMM}, and two thresholds T_{upper} and T_{lower}. If $P_{GMM} > T_{upper}$, it means that GMM recognizer is confident that two gestures are similar, so the final similarity score of Garda is $10 \times P_{simi}$. If $P_{GMM} < T_{lower}$, it means the GMM recognizer is confident that two gestures are not similar at all and the final result is $-P_{simi}$. Otherwise, GMM cannot make a confident judgment, so the final similarity of Garda is the same as P_{simi}.

classifiers to make the final classification decision (Rutkowska, 2004). Since the different classifiers independently measure the similarities of gestures with different features, the multi-expert systems combine those results to achieve a lower EER than other recognizers.

The rationale for selecting Protractor instead of the other time sequence matching method in Garda is that the Protractor has a lower EER compared to LCS, EDR, and DTW. We combined GMM and Protracto, because the two methods focus on different features of gestures. GMM does not take into account the order of gesture points, focusing exclusively on the shape of a gesture. Conversely, Protractor concentrates on the trajectory of a gesture, especially the connections between adjacent gesture points. Together, they account for a gesture performed in a certain shape and according to a certain trajectory.

Three steps compose both Garda and SVMGarda. First, the recall gesture is sent to the GMM authentication verifier. We set two thresholds T_{upper} and T_{lower} for the Gaussian Probability P_{GMM}. If $P_{GMM} > T_{upper}$, we can make the decision that this recall gesture belongs to a certain gesture type G_i. If $P_{GMM} < T_{lower}$, we can decide that this gesture does not belong to gesture type G_i. Otherwise, we cannot decide the gesture's type. Second, the recall gesture is sent to a time-series authentication recognizer. For the Garda system, we use Protractor to measure the similarity. For SVMGarda system, we use the Protractor kernel SVM to measure the similarity P_{simi} that a recall gesture belongs to a certain gesture type G_i. In the last step, we combine the two probabilities together by modifying P_{simi} with a modification factor a_m, which is determined by the GMM verifier.

3.5 Datasets

Small participants number in a dataset may not reflects the real performance of authentication systems (Sugrim, Liu, & Lindqvist, 2019). To mitigate the limitation of the gesture datasets, we tested the recognizers in seven different datasets.

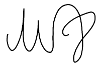
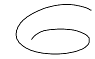
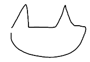
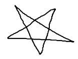
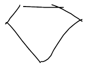
Dataset	Recall Set (Trial #)	Gesture (Types #)	Screen Size (Inches)	Samples
Freeform (Set 1)	11-12	56	10.1	
Freeform (Set 2)	13-17	54	10.1	
$1 Demo	11-30	16	3.8	
Vatavu (Set 1)	11-20	18	6.1	
Vatavu (Set 2)	11-20	20	6.1	
MMG corput	11-30	16	13.3	
HHReco	11-30	13	6.2	
SUSig	11-30	94	3.7	
MCYT -100	11-50	100	6.3	

Table 3.1: Summary of the seven datasets used in our analysis. The template set of each dataset is always the first ten trials (# 1 to #10) of every type of gesture. The screen size refers to the screen size of device on which the gesture samples were collected for that dataset. SUSig and MCYT-100 datasets also included attacks, which were used in our attack evaluations.

Table 3.1 shows a summary of the datasets we used in our analysis: (i) Freeform gesture dataset (Sherman et al., 2014), (ii) $1 Demo dataset (Wobbrock et al., 2007), (iii) Vatavu's gesture datasets (Vatavu, Vogel, Casiez, & Grisoni, 2011), (iv) MMG corpus dataset (Anthony, Vatavu, & Wobbrock, 2013), (v) HHReco dataset (Hse & Newton, 2004), (vi) SUSig (Kholmatov & Yanikoglu, 2009), and (vii) MCTY-100 (Ortega-Garcia et al., 2003). For consistency, we opted to use the first 10 trials from each dataset as Template sets. The remaining trials are called the Recall set. We use the first five datasets to test the authentication accuracy of different recognizers, and use the last two datasets to examine the ability of the recognizers to resist imitation attacks.

3.6 Results

In this section, we first present the results of our analysis of the invariances (rotation, scale, location) and their effect on authentication performance. We follow up with results on the performance of the different recognizers and results of the brute-force and imitation attacks. Finally, we present the implementation and evaluation of Garda as it operates on a mobile device.

We used two-way ANOVA for testing the statistical significance between the authentication performances of individual gesture invariances and the benchmark, since both the invariance combination types and the template numbers can affect the authentication result. We used repeated measures one-way ANOVA to test the statistical significance between the authentication performances of the 13 recognizers on different datasets. We used Bonferroni corrected p-values for the post-hoc test for controlling the familywise error.

3.6.1 Invariances Analysis

Figure 3.5 shows the individual effects of rotation, location and scaling on authentication performance. In this analysis, we used the EER of the recognition-optimal combination as the benchmark.

A two-way ANOVA test indicated a statistically significant difference between individual gesture invariance authentication performance and the benchmark ($\chi^2(251) = 11.6$, $p < 0.001$), while the interaction effect between the two factors is not statistically significant ($\chi^2(251) = 0.17$, $p = 1$). Thus, we did not consider the interaction effect between the the invariance combination types and the template numbers.

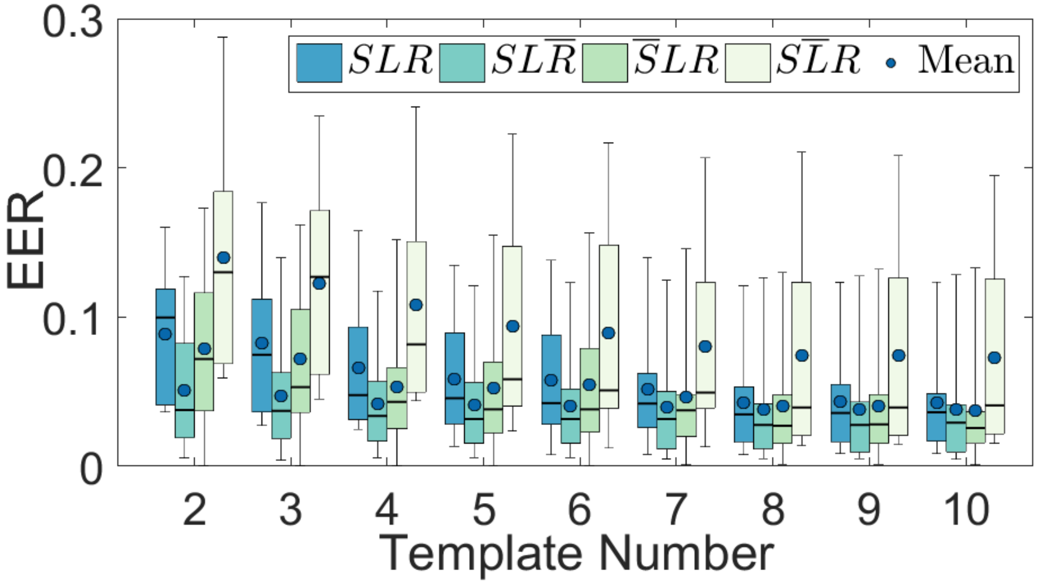


Figure 3.5: EER values under combinations (SLR), ($SL\overline{R}$), ($\overline{S}LR$), and ($S\overline{L}R$). S, L, and R mean the gesture's scale, location, are rotation are invariants. $\overline{S}$, $\overline{L}$, and $\overline{R}$ mean the these three are variants. The whiskers show the maximum and minimum EER of each group. We can see that rotation alone as variant ($SL\overline{R}$) has obvious positive effect authentication accuracy; scale alone as variant ($\overline{S}LR$) has no obvious effect authentication accuracy; location alone as variant ($S\overline{L}R$) has obvious negative effect on the authentication accuracy.

Rotation Invariance

Figure 3.5 demonstrates that designating the gesture rotation as variant can improve the recognition performance (i.e. reduce the EER). This result holds true even with a different number of template gestures across different gesture datasets. This can be explained by the reliability a users' drawing habit – people will tend to input their gesture into the tablet the same way every time. Very rarely, however, do gestures in the same dataset have the same rotation angle as other gestures.

The statistical test between $SL\overline{R}$ and SLR with template number (2 to 10) shows there is no statistically significant difference between the two ($p = 0.373$, $d = 0.42$). The lack of significance stems from the fact that the EER values become relatively close as the number of templates becomes larger (8, 9, 10). A recognizer is more likely to identify a genuine trial when there are more genuine templates to test against. This reduces, overall, the effect of invariance combinations on EER. If we do not consider EERs with large template number (8, 9, 10), then there is statistically significant difference between $SL\overline{R}$ and SLR ($p = 0.013$, $d = 0.56$). As such, allowing for rotation variance can reduce the possibility of accepting false gestures.

Scale Invariance

Figure 3.5 shows that the scale variable has a slight, positive influence on recognition performance. The primary reason is that Protractor is based on the cosine distance between two gesture sequences. The cosine distance measures the directional difference between two vectors and ignores the Euclidean distance. Thus, the difference among the size drawn gestures will not affect the relative similarity scores. The statistical test between $\overline{S}LR$ and SLR shows no statistically significant difference ($p = 1$, $d = 0.14$). Treating scale as variant does not have a statistically significant influence on authentication performance.

Location Invariance

Figure 3.5shows that taking location as variant has a negative influence on recognition accuracy; this is because it is hard for users to draw at the same location on the screen when repeating their gestures. The recognition performance in Garda, therefore, is not based on the absolute similarity among trials of the same person; it depends on the relative difference of similarities between genuine trials and imitations. Taking gesture location as variant would reduce the similarity of genuine gestures and increase the similarity of imitation gestures. This would make it more difficult to distinguish the genuine and imitation

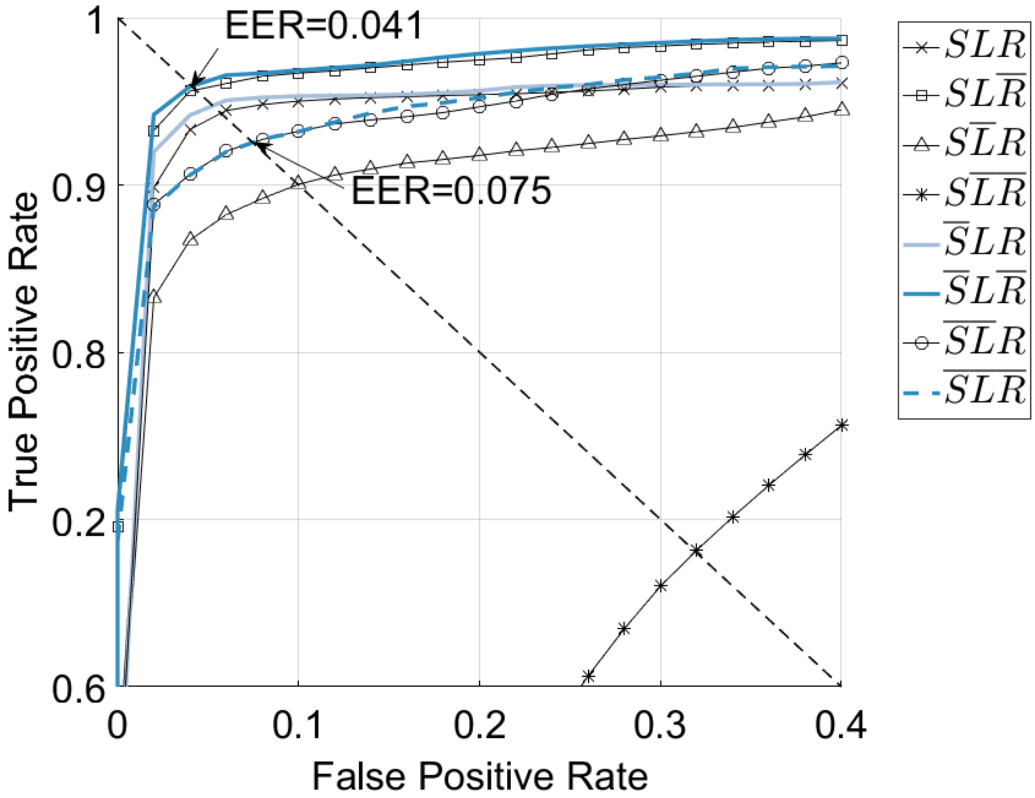


Figure 3.6: The average ROC curves of the five gesture datasets with two to ten template gestures for the eight combinations of the three variants (Scale, Location, Rotation). S, L, and R mean the gesture's scale, location, are rotation are invariants. $\overline{S}$, $\overline{L}$, and $\overline{R}$ mean the these are variant. We can see that the combination ($\overline{S}L\overline{R}$) has the lowest EER (=0.041) across the five datasets. While the recognition-optimal case (SLR) can only achieve EER=0.075. We conclude that combination ($\overline{S}L\overline{R}$) is the optimal authentication selection of three variants.

gestures, to the detriment of the authentication performance. The statistical test between $S\overline{L}R$ and SLR shows a statistically significant difference ($p = 9.788 \times 10^{-4}$, $d = 0.61$).

Variants Combinations Effects on Recognition

Figure 3.6 shows that the combination ($\overline{S}L\overline{R}$) achieves the lowest EER (0.041) on average. It can be explained by the individual effects of the three variables: rotation variant has statistically significant positive effect on EER, location variant has statistically significant negative effect on EER, and scale variant has slightly positive effect on the EER. Compared to the common case of the three combinations (SLR), where the EER = 0.075, the $\overline{S}L\overline{R}$ reduces the EER by 45.3$\%$.

Additionally, the ROC curves of $\overline{S}L\overline{R}$ and $SL\overline{R}$ are very close. It verifies our analysis that the gesture scale variable has no statistically significant effect on EER.

We also can observe that the four combinations that take location as variant ($S\overline{L}R$, $S\overline{LR}$, $\overline{SL}R$, and $\overline{SLR}$) are the lowest four ROC curves. This means that, with the same conditions for gesture scale and rotation, taking gesture location as variant will always have a negative effect on authentication accuracy. Generally, the reason is that people cannot keep their gesture's location at a relative fixed place on the touch screen. This means that a system differentiating between different gesture locations can distort the similarities among the genuine and fake gestures.

3.6.2 Performance of Different Recognition Methods

To select the optimal recognition method for gestures, we tried 13 different methods in four groups: sequence matching group (Protractor, EDR, LCS, DTW), One-class SVM group (Protractor-, EDR-,LCS-,DTW-kernels), HMM group (dHMM, sHMM, cHMM), and multi-expert group (SVMGarda, Garda). Since our analysis of gesture invariances is based on Protractor, we also used Protractor as the baseline for the comparison of these 13 methods. We evaluated the above methods with regard to EER values, authentication times, brute-force and forgery attacks.

We used 10 gesture trials as the template gesture set to minimize the bias from the template selection. The reason is that the selection of template gestures may effect the recognition performance. As a result, more gesture templates used ensure that less bias

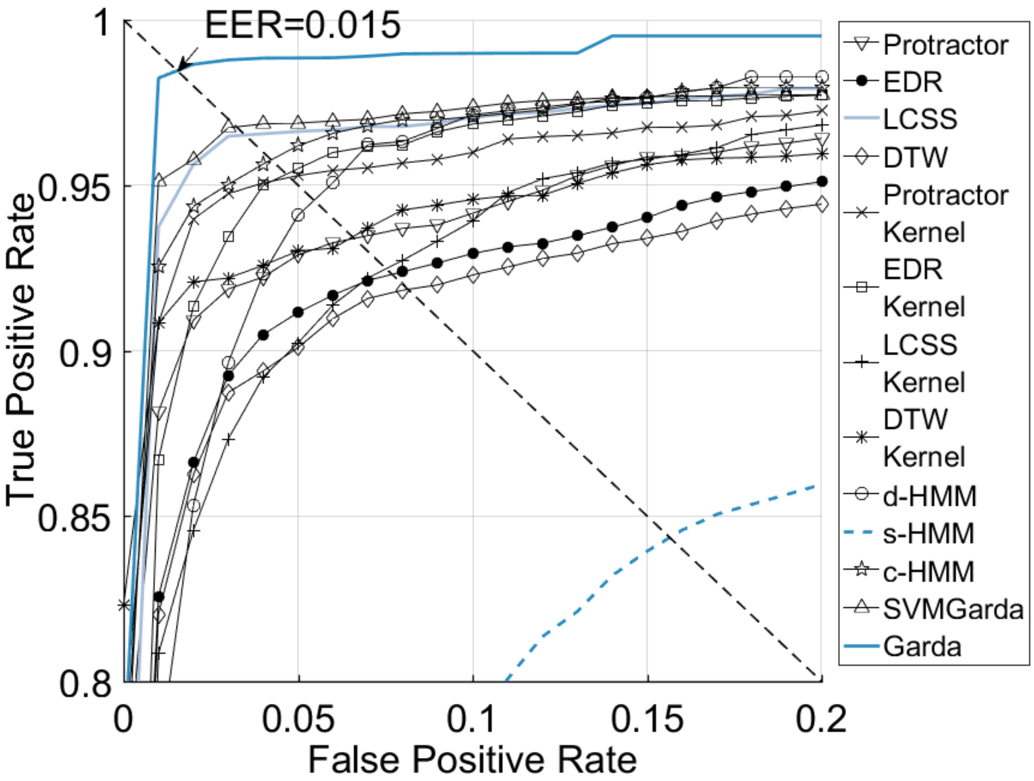


Figure 3.7: The average ROC curves for the 13 recognition methods over the five gesture datasets. We found that Garda has the lowest EER (0.015). Since the ROC curve of Garda is closest to the up-left corner, it should be the most tolerant of the change of authentication threshold. We conclude that Garda is the best among the 13 methods.

that exists in the selected templates.

Figure 3.7 shows that Garda is the best among the 13 recognizers. We averaged the ROC curves of the five gesture datasets for the 13 recognizers. We found that Garda achieves lowest EER (0.015) in the averaged ROC. The ROC curve of Garda is the closest to the up-left corner, meaning that Garda can recognize the most genuine gestures correctly while rejecting attacks as well as other recognizers.

Table 3.2 shows that Garda has most of the lowest EERs among the 13 recognition methods through the five datasets. The authentication time for Garda is around 2 milliseconds, which is also among the lowest authentication times. Specifically, both of the two

	Freeform (Set 1)	Freeform (Set 2)	MMG	$1 Demo	HHReco	Vatavu (Set 1)	Vatavu (Set 2)
Protractor	0.027	0.105	0.032	0.021	0.015	0.008	0.005
EDR	0.062	0.137	0.185	0.044	0.032	0.031	0.010
LCS	0.036	0.121	0.056	0.020	0.013	0.011	0.002
DTW	0.062	**0.142**	**0.204**	0.049	0.026	0.011	0.002
Protractor-K	0.035	0.141	0.046	0.032	0.047	0.004	0.011
EDR-K	0.048	0.055	0.119	0.034	0.037	0.009	0.003
LCS-K	0.036	0.118	0.141	0.058	0.040	0.006	0.006
DTW-K	0.053	0.113	0.063	0.092	0.028	0.001	0.013
SVMGarda	0.026	0.116	***0.032***	0.014	0.012	0.011	0
d-HMM	0.036	0.113	0.065	0.045	0.038	0.023	0.011
s-HMM	**0.109**	0.130	0.198	**0.165**	**0.144**	**0.140**	**0.104**
c-HMM	0.044	0.104	0.052	0.015	0.034	0.005	0
Garda	***0.019***	***0.047***	0.033	***0.012***	***0.007***	***0***	***0***

Table 3.2: EER values against estimated authentication time. Each recognition method is implemented in MATLAB and tested on five gesture datasets in terms of EER and authentication time (t) in milliseconds. Since the authentication time is based on MATLAB computations, it can be only used for a relative comparison among different recognizers. In each dataset, the lowest EER is shown in bold and italic, while the highest EER is only bold. Generally, the ME group always has the lowest EER among different datasets and authentication methods. Between the two ME methods, their EER performances are dependent on the different datasets. However, since Garda has much lower EER than SVMGarda in Freeform (set 2) and the computation cost of Garda is always lower than SVMGarda, we conclude that Garda is the best among the 13 methods.

multi-expert methods, Garda and SVMGarda outperform the other recognizers. Garda is also more stable than SVMGarda. For example, in Freeform (Set 2), Garda achieved the lowest EER (0.047), while SVMGarda achieved a relatively higher EER (0.116).

Based on repeated measures ANOVA analysis on the 13 recognizers, there is a statistically significant difference between EERs when choosing different recognizers ($\chi^2(90) = 9.71$, $p < 0.001$). From our post-hoc analysis, we found there are no statistically significant differences between Garda and Protractor ($p = 0.121$, $d = 0.52$), Garda and LCS ($p = 0.0749$, $d = 0.68$), Garda and DTW ($p = 0.0545$, $d = 1.17$), Garda and Protractor kernel SVM ($p = 0.0528$, $d = 0.89$), and Garda and SVMGarda ($p = 0.208$, $d = 0.47$).

Similarity in EER, however, does not necessarily bear on the effectiveness of an authenticator; for instance, Protractor cannot prevent brute-force and imitation attacks as well as Garda, irrespective of whether its EER is statistically significantly different. Figure 3.7 also shows that the ROC curve of Garda is much more ideal than the other recognizers.

We found that the authentication times of the one-class SVM group are much longer than the sequence matching group while the EER values are relatively similar. The reason for the longer authentication time is that, with the similarity matrix, SVM methods require an extra step to examine the similarity scores with training gesture trials and converting the similarity to probabilities. Since one-class SVM group distinguishes the gestures based on the sequence matching kernel functions, its ability to distinguish gestures should be the same as sequence matching group methods. Thus, the one-class SVM group has similar EER as sequence matching group methods.

The sHMM method consistently performs with the worst EER, due to its process of segmenting a gesture into several parts based on sharp turns and then classifying those parts into 14 observations. sHMM tends to lose a lot of useful information through this segmentation, and this makes the gestures more likely to be misclassified.

Performance Under Brute-Force Attacks

Figure 3.8 shows how the authentication methods resist brute-force attacks. We found that the recognizers' performance are polarized. On the one hand, EDR, LCS, DTW, dHMM, sHMM and Garda resisted all of the brute-force attacks. On the other hand, the brute-force attack cracked most of the gestures of the other recognizers. Specifically, we found that the features of recognizers that can resist attacks are distance based (such as EDR, LCS, and

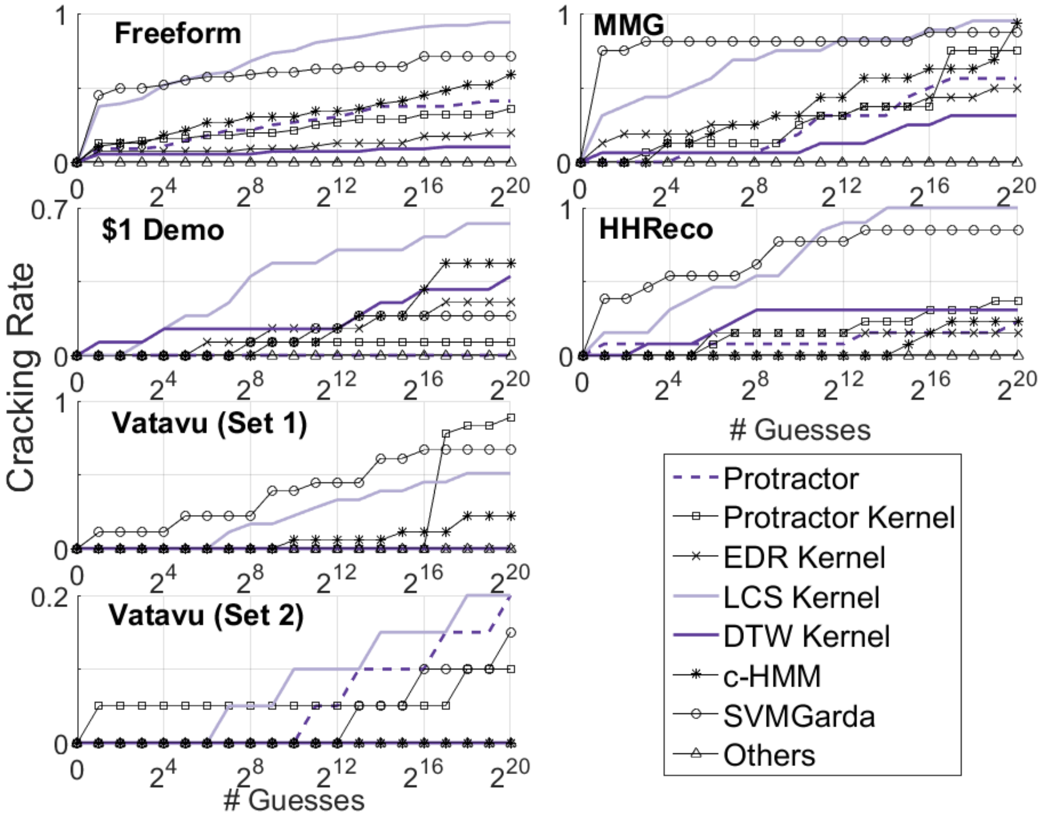


Figure 3.8: Guessing success rates of brute-force attacks of the 13 authentication systems on different datasets. The ”Others” includes Garda, EDR, LCS, DTW, dHMM, and cHMM. From the cracking rates, Garda, EDR, LCS, DTW, dHMM, and cHMM have the best ability on resisting brute force attacks since the success cracking rates of them are 0% through the 6 datasets.

DTW) and time series HMM-based (dHMM, sHMM). SVM was generally weak against these attacks.

Performance Under Imitation Attacks

We used the datasets MCYT-100 and SUSig to examine the 13 methods’ performance under imitation attacks. These datasets have samples from both legitimate users and imitation attacks from skilled attackers. We used the first ten legitimate trials for each user as templates and the rest as legitimate authentication attempts. The skilled attackers observed the genuine trials and practiced them until they felt confident with attacks.

Method	MCYT-100	SUSig
Garda	0.045	0.035
Protractor	0.151	0.350
EDR	0.155	0.366
LCS	0.101	0.420
DTW	0.189	0.371
Protractor-K	0.163	0.335
EDR-K	0.103	0.402
LCS-K	0.084	0.344
DTW-K	0.093	0.397
SVMGarda	0.096	0.479
d-HMM	0.233	0.333
s-HMM	0.338	0.445
c-HMM	0.125	0.202

Table 3.3: The EER of the 13 authentication methods in MCYT-100 and SUSig datasets under skilled forgery attacks. Only Garda has considerable advantages in EER.

Table 3.3 shows that only Garda outperforms the rest 12 methods on resisting forgery attacks. EER values of Garda are 0.045 and 0.035 in MCYT-100 and SUSig, respectively. In contrast, the lowest EER values for the other methods are 0.084 (LCS Kernel) and 0.202 (c-HMM). In summary, Garda outperforms all of the other approaches against imitation attacks, and none of the other approaches perform well against both of the datasets.

3.6.3 Mobile Device Implementation and Evaluation

We implemented Garda, described in Figure 3.4, on an Android platform. Our test device was a Samsung Galaxy Note 10.1, which had a 1.9 GHz Quad CPU and 3 GB of RAM. For our evaluation, we created 20 new gesture passwords (two templates per gesture) on the mobile device and recorded the processing time for training and authentication with a different number of gesture passwords.

Figure 3.9 shows our mobile device evaluation results. With the number of gestures (users) increasing, we see that the training time increases while the authentication time

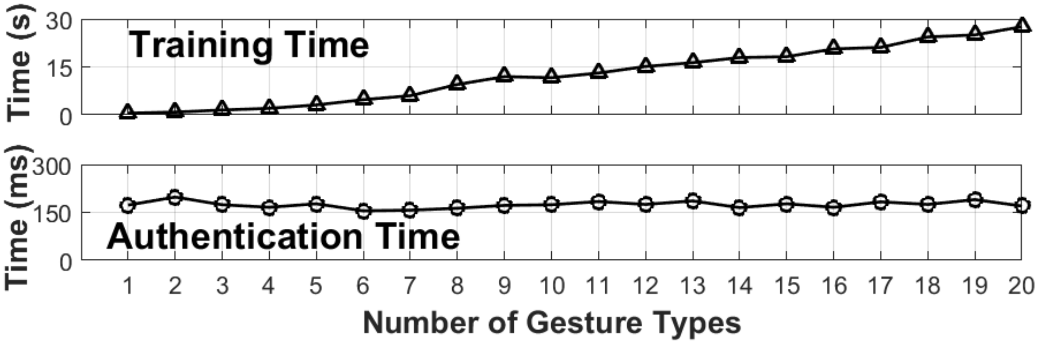


Figure 3.9: The processing time for training and authentication under different number of gesture passwords. The upper figure shows the time for training, the lower shows the time for authentication. Along with the increasing number of gesture passwords, the training time is gradually increasing while the authentication time stays stable.

stabilizes around 150 ms. The training time increases because Garda uses the Expectation-Maximum algorithm to train the UBM based on all the gesture trials in the dataset and re-train the Gaussian Mixture Model (GMM) for all gesture types based on this new UBM. More gestures lead to more time spent training models. In the authentication phase, we only need one GMM for a given gesture, so the authentication time remains stable. Training time is unlikely to pose a usability problem: there are typically not multiple passwords on a mobile device, and our training time, even with 20 different gestures, amounted to only 30 seconds and could be run in-background over other tasks. Efficiency of implementation is another feature recommending Garda as a gesture recognizer.

3.7 Summary

After developing Garda as a multi-expert gesture recognizer, we also implemented and evaluated Garda on a mobile device. Our results show that our implementation can largely improve the performance of gesture-based au- thentication systems. Garda was the final result of a rigorous evaluation of 13 different methods to implement gesture recognizers.

We applied several datasets and two different attacks against the recognizers. Finally, we conducted the first analysis determining how tuning the variables of preprocessing methods in gesture recognizers can impact their authentication performance. The presented authentication-optimal combination can reduce up to 45.3% of EER on average compared to recognition-optimal configuration used in previous work.

CHAPTER 4

GUESSING ATTACKS ON GESTURE PASSWORDS

4.1 Overview of Chapter

Analyzing the guessability of gesture passwords involved four key stages:

First, we introduced two types of datasets for our analysis: dictionary dataset and test dataset. We used a dictionary dataset to analyze the most common gesture passwords that users selected as passwords. The test dataset is used to test the attack algorithms of our guessing methods.

Second, we introduced the preprocessing steps and dynamic time warping method for gesture recognition. Based on the analysis, we proposed an enumeration method for full space analysis of gestures.

Third, we introduced the steps for estimating the size of weak subspace of gestures based on the common selected gestures by people.

Finally, based on the weak subspace, we developed an effective dictionary of attacks for gestures. We implemented brute-force attacks based on previous study on graphical password's symmetric features, and used it as the benchmark of our guessing attack analysis.

4.2 Method

In this section, we introduce gesture datasets used in our analysis. Then, we present a generalized procedure for gesture recognition based on DTW. We propose a method for estimating the sizes of full and weak subspaces of gesture passwords. Finally, we present our algorithms for performing dictionary and brute force attacks.

4.2.1 Data Acquisition and Classification

Effective guessing attacks require data about user choices; the best options for guessing attacks on gesture would be drawn from the most common gesture passwords that users select. Before building the machinery required to perform guessing attacks, we needed a set of data to inform attacks and to evaluate the efficiency of our attack algorithms.

For these purposes, we aggregated two sets of data. We labeled the sample set from which we derive our dictionary as the Dictionary dataset: it is composed of three gesture datasets, obtained from previous work on gesture passwords (Sherman et al., 2014; Y. Yang et al., 2016) and collected from 232 users. The testing set, from which we ran our attack algorithms, was called the Test dataset. Test dataset is composed of two datasets of gestures that we have newly obtained from 109 volunteer participants.

Dictionary Dataset

Previous work on user-generated gesture passwords classified created gestures into six rudimentary groups: Digit, Shape, Lines, Letter, Symbol, and Words (Y. Yang et al., 2016). We differed from this precedent, defining our groups as:

- Digit: 0,1,2,3,4,5,6,7,8,9;

- Geometric Shape: triangles, circles, cylinders, etc.;
- Letter: upper and lower case of 26 letters;
- Mathematical Function: basic signal functions;
- Mathematical Symbol: characters used in mathematics;
- Music Symbol: treble clef, simple notes, etc;
- Special character: keyboard special characters.

In our analysis, we merged the Lines and Shape groups from previous research into the Geometric Shape group. We then expanded the Symbol group by breaking it into specific meanings for four new groups: Math function, Math symbol, Music symbol, and Special character. We opted to remove the Word group; after aggregating the sum of datasets, we concluded that words do not appear as frequently as posited in prior work (Y. Yang et al., 2016).

In the Dictionary dataset, we found that 407 out of 529 gestures were created by participants because the gestures carried some semantic meaning. These gestures, that we hypothesize are chosen because of their meaning, are what we define to be the weak subspace of gestures. In addition to the gestures with semantic meaning, 122 gestures in the Dictionary dataset and 90 gestures in Test dataset (General) carried no semantic meaning with regard to any of the seven groups. Since these gestures do not have obvious meanings, and it is not obvious whether they would be reused by other people ta, these gestures did not receive any specific treatment.

Test Dataset

We conducted a user study to collect the Test dataset. The study was approved by the Institutional Review Board (IRB) of Rutgers University. The experiment consisted of collecting two datasets: Test dataset (General) and Test dataset (Weak).

Data Collection. The Test dataset (General) was used to test the efficiency of the guessing attack. We asked participants to create gesture passwords for a user account without any instruction as to what to choose. We asked each participant to create one unistroke gesture and one multistroke gesture. Table 4.1 shows that we collected 109 unistroke and 109 multistroke gestures from the participants. 136 of the user-generated passwords (78 unistroke and 58 multistroke) fall into the weak subspace groups identified in Section 4.2.1.

The Test dataset (Weak) was used to tune the performance of the attacking algorithm, as our guessing attacks required information about the most common variations a human might add when performing a gesture password. This would enable it to create N trials of a single gesture with only slight differences between subsequent attempts – for example, creating a rectangle of slightly longer width each time for 100 tries. The attacking algorithm needs an idea of when to stop making adjustments to a gesture beyond a point that humans tend not to do – in the rectangle example, this might entail adjusting the rectangle to the full width of the screen. This means we need a large number of samples to determine parameters for the variable ways people draw a given gesture in each one of the groups in Section 4.2.1. We collected Test dataset (Weak) to meet this purpose. The weak subspace contains 84 specific gestures, we asked participants to give us trials of 30 out of the 84 gestures. Totally, we collected 3270 gestures in Test dataset (Weak).

Gesture Type	Unistroke	Multistroke	subtotal
Weak Subspace Feature	78	50	128
Symmetric Feature	36	47	83
Weak & Symmetric	34	29	63
Any Feature	109	109	218

Table 4.1: Summary of free-form gestures in the Test dataset (General). "Weak Subspace Feature" are the free-form gestures that fall into the weak subspace groups in Dictionary dataset. "Symmetric Feature" are the gestures that have horizontal or vertical symmetry. "Weak & Symmetric" are the gestures that have both of the two features. The "Weak Subspace Feature" size (128 gestures) is larger than "Symmetric Feature" size (83 gestures). It verifies the analysis of the Dictionary dataset, where the coverage of weak subspace features in gestures are wider than symmetric features.

To mimic typical password creation tasks, we asked participants to first create a gesture password and then recreate it to confirm it.

Participants. We recruited participants with flyers on our university campus. We required the participants to be 18 years old or over and familiar with touchscreen devices. We recruited 109 participants with ages ranging from 18 to 45 (Mean=20.67, SD=3.46). 50 were male and 59 were female. 88 individuals were pursuing an undergraduate degree, 6 were pursuing graduate degrees, and the remaining 15 had a graduate degree.

Apparatus. The gestures were collected on a Nexus 10 tablet with Android 4.4.2.

External Validity

One concern was that the different sizes of the Dictionary dataset and the Test dataset (General) might affect how commonly a weak subspace group appears; if true, this effect would pose a problem, because it would impact the generalizability of our analysis. It is possible that, at very different sizes of datasets, one or more of the seven groups could be over- or underrepresented in our data. At the very least, it could affect the ability to draw valid comparisons between the Dictionary and Test (General) datasets. We computed the relative frequencies of the seven groups of weak subspace gestures in both the Dictionary dataset and Test dataset (General). The relative frequencies of one group of weak subspace gestures are calculated as in Eq.(4.1).

$$\text{Relative frequency of group} = \frac{\#\ of\ weak\ group}{\#\ of\ entire\ weak\ subspace} \tag{4.1}$$

Table 4.2 shows that the relative frequencies of the Test dataset (General) are close to that of the Dictionary dataset, despite a different sample size and different participants across separate studies. We can conclude, therefore, that we are targeting similar distributions of gesture groups here, and that they are suitable dictionary and attack targets, separately. We can also conclude that the Test dataset (General) is a suitable attack candidate for the Dictionary dataset.

We are careful to note that the demographic breakdown of the Dictionary set and the Training set is preferential to people who work on campuses and college students. This affects our ability to make strong statements about how representative these group categories are when applied to the general population. It is possible that age, education, or nationality could be factors that affect the size of each gesture group. This potential variation would have to be considered in the future, and could be easily performed using the attack methodologies outlined by our work.

4.2.2 Gesture Recognition Approach

In this section, we present the gesture recognition approach used in our analysis. First, we introduce the preprocessing procedures and their respective effects on the full space. Then, we will introduce Dynamic Time Warping (DTW) as the recognition method.

Group Name	Dictionary Dataset	Test Dataset (General)
Digit	9.1%	9.4%
Geometric Shape	44.5%	45.3%
Letter	28.0%	28.9%
Math Function	9.1%	10.2%
Math Symbol	4.4%	5.5%
Music Symbol	2.7%	0.8%
Special Character	2.2%	0%

Table 4.2: Relative frequencies of seven groups of weak gestures in Dictionary dataset and Test dataset (General). It shows that the relative frequencies in two independent datasets are very similar. It means that the two datasets are similar to each other and that the size of each dataset is not significantly affecting how often each group appears.

Preprocessing

Human users cannot exactly repeat the same gestures on a touchscreen. To improve recognition accuracy, various preprocessing methods are often used to prevent relative (minor) differences between two gestures from obfuscating the underlying similarity of a single users gesture-making. The purpose of our work is to analyze the independent effect of preprocessing steps on a systems ability to make these measurements and comparisons. Therefore, we sought to analyze as many processing steps as possible. These independent preprocessing steps could then be added or removed based on design requirements. In summary, our analysis on preprocessing methods is applicable to approaches that measure the similarity between time series. Figure 4.1 shows a generic preprocessing procedure. There are five steps: 1) Stroke connection; 2) Resampling; 3) Location invariance; 4) Rotation invariance; 5) Scale invariance.

Stroke Connection. Unistroke gestures, which consist of a single continuous stroke to draw (for example, the number '8'), are easy for recognition algorithms to handle – it is a single sequence of (x,y) data coordinates. Most, if not all recognition algorithms,

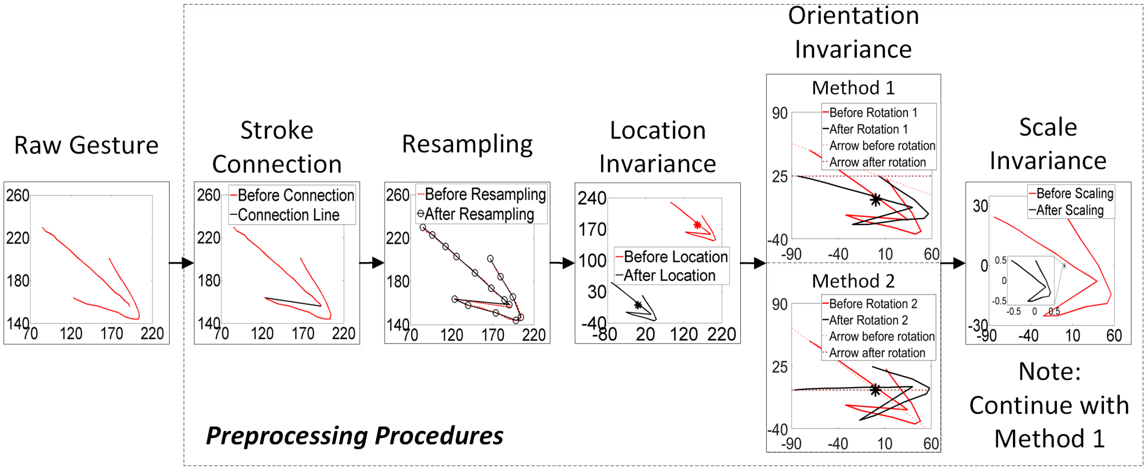


Figure 4.1: The preprocessing procedures are generic steps in gesture recognition. First, user has inputted a gesture on the touchscreen (Raw Gesture). Stroke Connection: a multistroke gesture's strokes are connected to one sequence by time order. Resampling: the gesture is resampled to a fixed number of sample points. Location Invariance: the gesture is translated to the origin point. Orientation Invariance: the gesture is rotated to the same direction. Scale Invariance: the gesture is scaled to the normal size.

work best on this type of data. Multistroke gestures, which require more than one stroke to draw (for example, a # symbol), can complicate the recognition problem. These gestures are represented as multiple, separate pairs of coordinates in the Dictionary dataset. Using most recognition algorithms on these multistroke gestures would require matching every sequence exactly, which can lead to increased computation: where M is the number of strokes, there are $M!$ combinations for a recognition algorithm to consider. To simplify this problem, we connect gesture strokes end-to-end as displayed in Figure 4.1. This transforms a multistroke gesture into a single stroke gesture and now only requires a single run of the recognition algorithm, reducing $M!$ combinations down to 1. However, if a user draws the multistroke gesture in the wrong order, this is considered to be incorrect. The effect of stroke connection is that the gestures can only have one continuous trace.

Resampling. Gestures that are sampled from the touchscreen have coordinate sequences that are of variable length, sequence length being a function of the sampling rate and the speed by which the user enters the gesture. Resampling gestures to a constant value, however, removes variation and simplifies analysis. The question is, then, what is the best resampling rate to use in our analysis? The Resampling step of Figure 4.1 shows an illustrative example of this process. By examining the effect of different resampling rates (R=2, 4, 8, 16, 32, 64, 128) on the Equal Error Rate (EER) in the Dictionary dataset, we found that after $R = 16$, the EER values are both low and stable. We use EER to evaluate whether the parameter values have an impact on recognition accuracy. We do not use it to determine or optimize the best values based on specific datasets. The selection of suitable parameter values can significantly influence recognition performance. For example, if R is too small, information contained in a circle gesture will be lost and may become indistinguishable from that of a square. The effect of resampling is that the number of sample points is fixed to 16 and sequences with other lengths will not be considered.

Location Invariance. When a user draws a gesture on the screen, they rarely draw it at exactly the same position every time. The solution to this is to make the gesture location invariant by translating the centroid of each gesture to the origin (0,0) point of a Cartesian coordinate system. The Location invariance step in Figure 4.1 demonstrates an example of this. The effect of location invariance is that the centroid is kept constant at the origin.

Orientation Invariance. Users do not draw their gestures at the same orientation every time – the centroid of the gesture may be rotated slightly off center. The solution to this problem is to make the orientation invariant. Two methods are currently used in literature for removing orientation variance, and the upper and lower diagrams for Orientation Invariance in Figure 4.1 show examples of these two methods, which are summarized as:

1. Rotate the gesture until the direction vector, directed from the first point to the last point, is parallel with the x-axis.
2. Rotate the gesture until the direction vector, directed from the centroid to the first point, is parallel with the x-axis (Wobbrock et al., 2007).

The two methods differ slightly: in Method I, only the first and last points of gesture are involved in rotating the gesture, whereas all the gesture points are involved in rotating the gesture in Method II. This could lead to a difference when it comes to resisting attacks, and therefore needs to be studied in detail.

Scale Invariance. Users rarely draw their gestures in the same size every time. As in prior steps, the solution is invariance, this time with regards to scale. As shown in Figure 4.1, scale invariance divides the gesture sequence by the range (the difference between minimum and maximum) of the gestures X and Y coordinates for each respective coordinate pair (Sherman et al., 2014; Y. Yang et al., 2016). The effect of scale invariance is that the gesture is bounded in a square $(-0.5, 0.5)$.

Recognition Method: Dynamic Time Warping

This section briefly introduces Dynamic Time Warping (DTW) (Myers & Rabiner, 1981), which is used as our gesture recognition algorithm. We then describe the Sakoe-Chiba Band (Sakoe & Chiba, 1978) for speeding up DTW calculations. We note again that we chose to use DTW as the recognizer, since it is already utilizedin several extant systems (Sae-Bae et al., 2012; Tian et al., 2013; Aslan et al., 2014; De Luca et al., 2012; J. Yang et al., 2015; Aumi & Kratz, 2014; Wu et al., 2013).

Implementation of DTW. Dynamic Time Warping (Myers & Rabiner, 1981) is used to

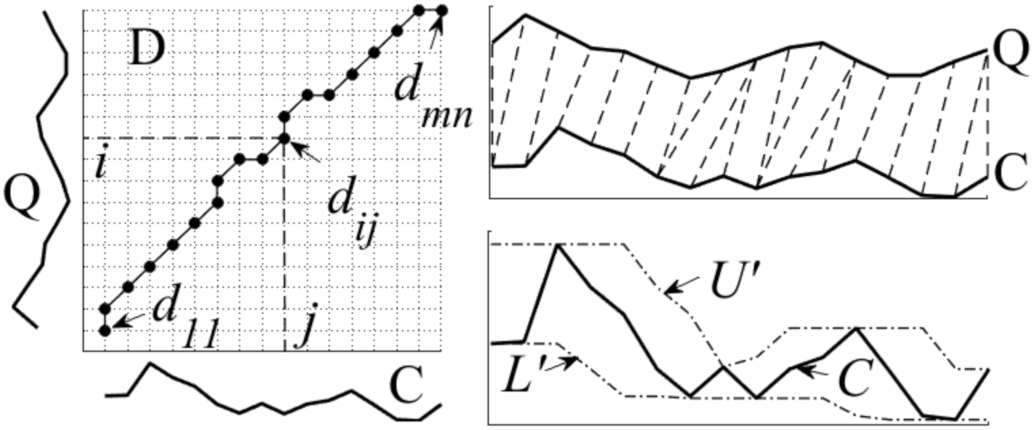


Figure 4.2: An illustration of the DTW algorithm and the Sakoe-Chiba band based on one dimension of our gesture data. The left figure shows how to dynamically search along the matching path in the grid. The top right figure displays how, according to the left matching path, the points in Q map to points in C. The bottom right figure displays the Sakoe-Chiba band using the broken line; U' and L' refer to the upper and lower envelopes, respectively. The envelope is wider when sequence changes and narrower when it plateaus.

measure the similarity between the 1-D sequences of two gestures: $Q = q_1, q_2, ..., q_i, ..., q_n$ and $C = c_1, c_2, ..., c_i, ..., c_m$. In our case, we construct a $n - by - m$ distance matrix D with an element $d_{ij} = ||q_i - c_j||$. DTW finds the non-decreasing path in D, starting from d_{11} and ending at d_{mn}, which has the minimum total value along this path. The left part of Figure 4.2 shows an example of the dynamic matching path between Q and C. Since the classic DTW algorithm dynamically searches the target matching path in the $n - by - m$ matrix, its computation cost is $O(nm)$.

Sakoe-Chiba Band. DTW requires considerable computation. We can improve on this with the Sakoe-Chiba band (Sakoe & Chiba, 1978). This is a known global constraint region for calculating the DTW similarity between two sequences. The aim of the band is to generate an envelope boundary for the underlying sequence. When the underlying candidate sequence is changing rapidly, the envelope is wider; it becomes narrower when the query sequence plateaus (E. Keogh & Ratanamahatana, 2005). Specifically, the upper and lower bands are $U_i = max(c_{i-r} : c_{i+r}); L_i = min(c_{i-r} : c_{i+r})$. U_i, L_i, c_i is the i^{th} point of the upper, lower and candidate sequences. The lower part of Figure 4.2 shows an illustration of this. If another sequence Q does not fall into the Sakoe-Chiba band of C, the system determines Q is not similar to C. This allows us to shrink the search space to a more manageable size. A key issue is the proper selection of r, the number of points looked forward and backward of a given point. We set r equal to 10% of the sequence length, based on a review (Ratanamahatana & Keogh, 2004) of more than 500 papers.

4.2.3 Full Space Analysis

The full space refers to the total number of distinguishable gesture possibilities that can be differentiated by a gesture recognizer. Computing the full space for matching problems is easy, for example, there are 10^4 possible combinations of 4-digit PINs. This represents all the PINs from 0000 to 9999. However, with gestures, counting a full space of passwords is more difficult. Two gestures, for instance, can be technically different if even one coordinate pair is not the same. The coordinate values of a gesture drawing are real numbers with decimal places, which means that there is an extremely large, near uncountable amount of gestures that are different if one modifies even one coordinate point slightly. In this section, we present the additional processing steps on raw gestures to estimate the number of all possible gestures.

Gesture Discretization

The key to solving this issue is to ensure that a gesture recognizer does not consider two gestures that have almost all the same points to be different. For example, the difference could be that just one x-coordinate pair differs by 0.1. A recognizer will state that the similarity of those two gestures is high; therefore, it is likely they are the same gesture. The question now becomes: what is the largest difference between two coordinate pairs before two gestures might be considered to be different?

A gesture coordinate pair can take on a continuous value between two sample points. This is displayed in Figure 4.3(a), wherein the red line represents a continuous connection between the points. Gesture discretization imposes a rule on the gesture values: gesture values cannot vary continuously, but must vary by a fixed amount. If gestures vary by a fixed amount between points, then they become countable. The first step is to figure out the Discretization Level, N, which is the number of discrete values that are possible between two gesture points.

Discretization Level. We need a discrete, countable number of values between two gesture points to be able to enumerate the space. To do this, we can subdivide the range between two coordinate points into N units. This means there are N possible values between two gesture points when counting all possible gestures as opposed to the large, near-infinite number of points when the gesture remains continuous. N should be the smallest possible value while still keeping two distinct gestures – like a circle and a square – from being counted as the same gesture. To achieve a reference for N, we examine the values of N (from 5 to 40 in steps of 5) based on the EER in Dictionary dataset, as we did when determining the resampling rate R. We chose $N = 15$ units since it is a local minima for error rates. We note that $N = 15$ units is not the optimal value for any particular gesture but optimal for the dataset as a whole. Figure 4.3(a) shows an example of this, imposing countable units.

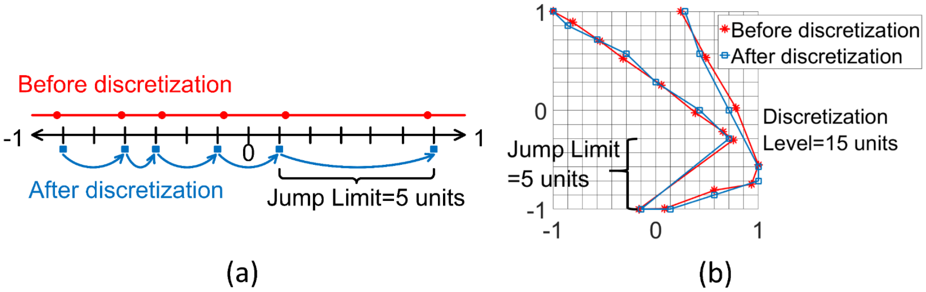


Figure 4.3: Example of gesture discretization. (a) In "Before discretization" above the axis, the round points are several values of gesture samples. It can be any value between the range [-1,1]. In "After discretization" under the axis, the values of gesture samples are rounded to the closest discrete values shown as square points. (b) shows an example of a gesture under discretization. The Discretization Level is 15 units, since the range [-1,1] is divided to 15 units. The Jump Limit is 5 units, since the maximum gap between the adjacent points is 5 units.

Jump Limit.

We have imposed a set number of values a gesture can take between sampled points when counting. However, a question remains: how large can a gesture value change? Figure 4.3 shows that, after gesture discretization, the adjacent discretized gesture points should not be too large. We require, after all, a semblance of continuity in order to preserve the shape of the gesture; otherwise a circle turns into a square. For example, if one sample point exists at the maximum end of the value range, its adjacent sample points is unlikely to be at the minimum end of the value range. If this is not controlled, then many details between the two points may be lost, and the gesture cannot be distinguished.

The Jump Limit, J defines the maximum possible difference between two adjacent gesture points. With resampling rate $R = 16$ and discretization level $N = 15$ units, we examined all the gaps between adjacent points in all the gestures in the Dictionary dataset. We found the largest gap is 12 units. Therefore, we set $J = 12$ units. The Jump Limit restricts that adjacent points of gesture variations must be less than the Jump Limit.

In summary, the size of the full gesture space is preprocessing-dependent and recognizer-independent. In the whole procedure to convert a raw gesture to a processed gesture in the full password space, the DTW recognizer is used to find reasonable reference values for the resampling rate and the discretization level. The resulting reference values cannot be optimized based on any particular recognizers or databases; this ensures that those reference values can also be applied to other recognizers that measure the similarity between time sequences, such as, Longest Common Subsequences (LCS), Edit Distance, Cosine Distance, and Euclidean Distance.

Enumeration of Full Space

We introduced the procedures to transform a raw gesture to a preprocessed gesture that can be used in the recognizer and consequently as part of an authentication system. In this section, we apply the effects introduced in the preprocessing and discretization steps to a function for enumerating the full space shown in Algorithm 1.

Algorithm 1 $GNume()$ is used to enumerate the full space with a specific resampling rate, discretization level, jump limit and orientation invariance method. We enumerate gesture sequences from the last point to the first point. First, we initialize the size of the full space $FS = 0$, the length of gesture $L = R$ and the sum of visited points in X and Y, $X_{sum} = Y_{sum} = 0$. Then, we find all of the possible values for the gesture's last points, X_e and Y_e. Since the gesture is discretized, the value of a gesture sample point can only be a number from 1 to N. By combining X_e and Y_e with different values, we obtain the $EndPointSet$ with size N^2, composed of (X_i, Y_i).

For each value in the $EndPointSet$, we check the orientation invariance method. If it

```
Require: The length of gestures in the full space, Resampling rate $R$. The Discretization
    level $N$, the size of discretization values. The maximum gap between two adjacent
    sample points in a discrete gesture, Jump limit $J$. The method to remove a gesture's
    orientation variant, method I or method II.
Ensure: $FS$ = Full gesture space size.
$FS \leftarrow 0$;
$L \leftarrow R$;
// length of gesture
$X_{sum}, Y_{sum} \leftarrow 0$;
// X, Y sums of visited points
$X_e, Y_e \leftarrow 1, 2, ..., N$;
// $X_e$,$Y_e$ are the coordinates of a gesture's end point.
$EndPointSet \leftarrow (X_e, Y_e)$;
// Combinations of $X_e$,$Y_e$.
for $(X_i, Y_i) \in EndPointSet$ do
    if Using Orientation Invariance Method I then
        $y_c = Y_i$;
        // Method I
    else
        $y_c = R \times N/2$;
        // Method II
    end if
    $FS \leftarrow GRecur(L, X_i, Y_i, X_{sum}, Y_{sum}, y_c, N, J) + FS$;
end for
return $FS$
```

Algorithm 1: Pseudocode for full space enumeration function GEnum(R, N, J, OrientationMethod). Starting with the gesture length and specific parameters, this code can enumerate all possible gestures in the full space with a given length.

uses method I, the Y-coordinate of a gesture's first and last points should be equal. Thus, the first value of the Y-coordinate, y_c, should be Y_i, the Y-coordinate of the last point. If it uses method II, the Y-coordinate of a gesture's first point should equal to the mean of all of the gesture's Y-coordinates, which is $R \times N/2$. Then, we use recursive function $GRecur()$ to enumerate all possible gestures with the given end point. Finally, the sum of FS is the size of the full gesture space.

Algorithm 2 $GRecur()$ is the the core recursive function in $GNume()$. The inputs of $GRecur()$ keep the same meanings as in $GNume()$. When $L > 1$, it means the whole

```
if $L > 1$ then
    $S \leftarrow 0$; //Initial gesture space size
    $X_{range} \leftarrow [max(1, X_i - J), min(N, X_i + J)]$;
    $Y_{range} \leftarrow [max(1, Y_i - J), min(N, Y_i + J)]$;
    for $X_j \in X_{range}$ and $Y_j \in Y_{range}$ do
        $L = L - 1$;
        $X_{sum} = X_{sum} + X_j$;
        $Y_{sum} = Y_{sum} + X_j$;
        $S \leftarrow GRecur(L, X_j, Y_j, X_{sum}, Y_{sum}, y_c, N, J) + S$;
    end for
    return $S$;
else
    $Flag_1 \leftarrow Y == y_c$; // If Orientation is Invariant?
    $Flag_2 \leftarrow X_{sum}/R == N/2$; // If Location is Invariant in X?
    $Flag_3 \leftarrow Y_{sum}/R == N/2$; // If Location is Invariant in Y?
    if $Flag_1$ and $Flag_2$ and $Flag_3$ then
        return 1; // Is a preprocessed gesture
    else
        return 0; // Not a preprocessed gesture
    end if
end if
```

Algorithm 2: Pseudocode for the recursive function GRecur(L, X_i, Y_i, X_{sum}, Y_{sum}, y_c, N, J) in Algorithm 1. This code enumerates the gesture in reverse order of the sequence until the first point is reached. Then, it will examine if the enumerated sequence is a preprocessed gesture by checking the rotation and location invariance.

gesture sequence is not generated yet, so we need to keep recurring the function. When L reaches 1, it means the full gesture sequences are generated. We need to check if the generated sequence is a preprocessed gesture. Based on the restriction from the orientation invariance step, the Y coordinate of the gesture's first point should be equal to either its last point (Y_i) or the mean of the gesture points ($R \times N/2$). From the location invariance step, the mean of the gesture's X- and Y-coordinates, which is X_{sum}/R and Y_{sum}/R, must stay constant and equivalent to the center of a gesture's value range ($N/2$). Only if the sequence fulfills the above criteria, the sequence can be regarded as a preprocessed gesture.

4.2.4 Weak Subspace Analysis

The weak subspace is a subset of passwords that people are more likely to choose. If an attacker has knowledge about people's password preferences, the attacker will perform more efficient dictionary attacks compared to simple brute force attacks.

In this section, with the full space of gesture passwords as the reference point, we study the weak subspace for gesture passwords. First, we discuss how to determine the "The Authentication Similarity Threshold" in our analysis. Then, we show the procedure for "Weak Subspace Size Estimation". We find different variations of gestures in "Extract Representative Gesture". To estimate the similarities of representative gestures, we need to "Restrict Gesture Searching Region" to improve the search efficiency, and use Monte Carlo method to "Estimate Weak Subspace Size". Finally, we remove the potential double-counting issue by "Check Overlapping Regions".

Authentication Similarity Threshold

In gesture-based authentication systems, a robust threshold is crucial for recognition performance. For example, if the threshold is set too low, illegitimate users may be able to mimic authorized users. Similarly, if the threshold is set too high, authorized users may be rejected due to slight variations in repeating their gesture. Since the optimal threshold varies for each dataset, we did not select the threshold by examining the EER of our datasets as we did when determining the resampling rate and related parameters. Instead, we examined the distribution of the DTW similarity scores among gestures throughout the Dictionary dataset and selected a reasonable value which can authenticate a majority of the legitimate users.

Weak Subspace Size Estimation

We estimate the size of the weak subspace as follows. First, we manually classify all of the weak subspace gestures by their symbols. Then, we extract several gestures of different drawing styles for each particular symbol. For each gesture of a particular drawing style under a specific symbol, we estimate the total number of similar gestures of it that have similarities less than the authentication threshold. The sum of all total numbers of the similar gestures of each drawing style for each particular symbol will form the size of the weak subspace. Figure 4.4 shows the procedure. Generally, there are four steps: 1) Extract representative gestures; 2) Restrict gesture searching region; 3) Estimate weak subspace size; and 4) Check overlapping regions.

Extract Representative Gestures

The reason for extracting representative gestures is that gestures for one symbol are not evenly distributed. For example, given a symbolic gesture, there could be ten gestures of one user's drawing style and twenty gestures of another user's style. In our weak subspace analysis, we should treat the two drawing styles from two people equally. Thus, we merge the gestures where the DTW distance is close and extract representative gestures.

First, we manually classify the gestures in Dictionary dataset by the different symbols. Then, for gestures from a specific symbol, we cluster the gestures by K-means (MacQueen et al., 1967; Lloyd, 2006) and evaluate the clustering result through the Silhouette value (Rousseeuw, 1987). Specifically, we start K-means at $K = 2$ and send the clustering result to be evaluated by the Silhouette. Silhouette value measures how similar a gesture is to its own cluster compared to the other clusters (Rousseeuw, 1987). Thus, we can find the optimal value

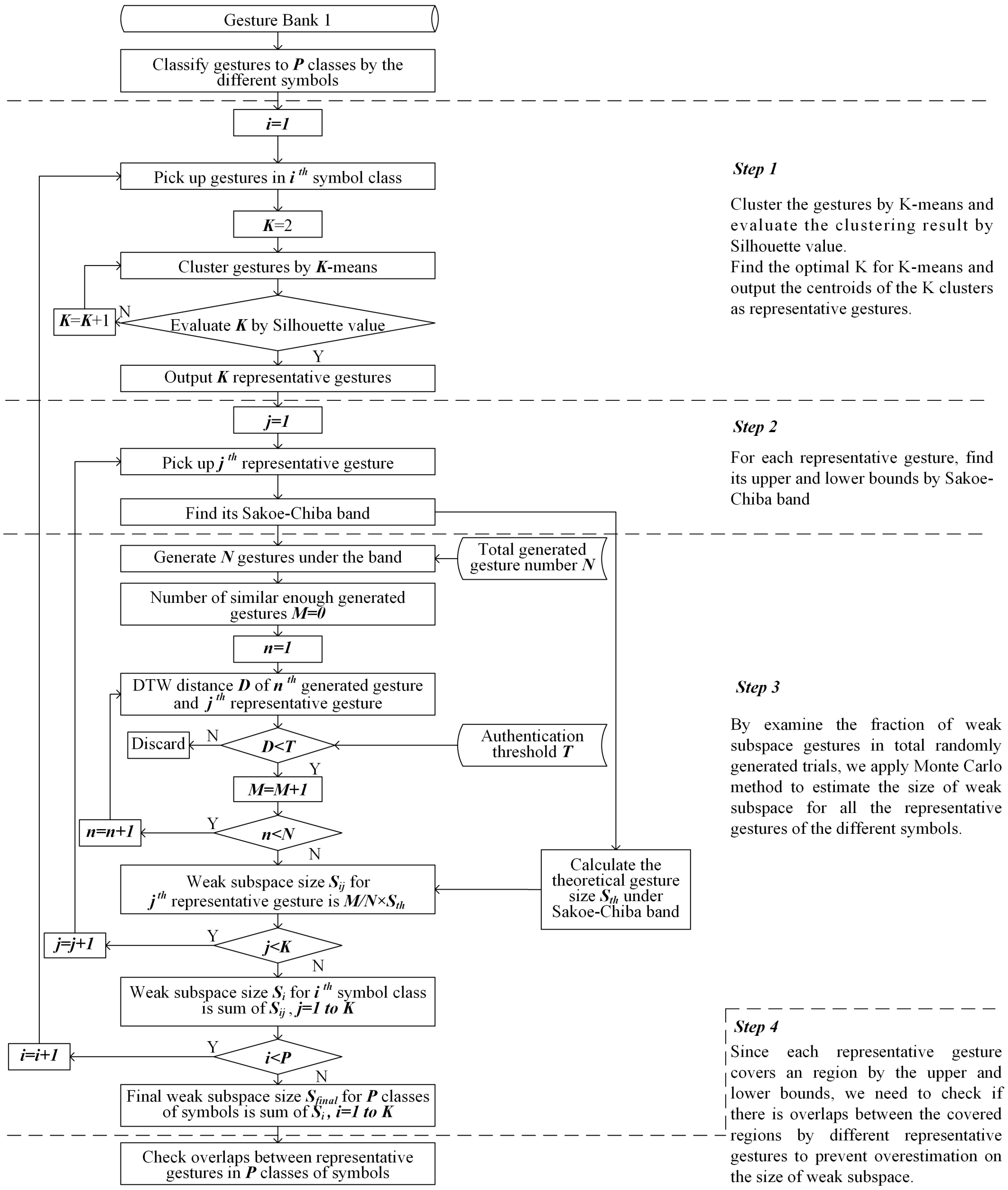

Gesture Bank 1
Classify gestures to P classes by the different symbols
i=1
Pick up gestures in i th symbol class
K=2
Cluster gestures by K-means
Evaluate K by Silhouette value
N
K=K+1
Y
Output K representative gestures
Step 1
Cluster the gestures by K-means and evaluate the clustering result by Silhouette value.
Find the optimal K for K-means and output the centroids of the K clusters as representative gestures.
j=1
Pick up j th representative gesture
Find its Sakoe-Chiba band
Step 2
For each representative gesture, find its upper and lower bounds by Sakoe-Chiba band
Generate N gestures under the band
Total generated gesture number N
Number of similar enough generated gestures M=0
n=1
DTW distance D of n th generated gesture and j th representative gesture
D<T
N
Discard
Authentication threshold T
Y
M=M+1
n<N
Y
n=n+1
N
Weak subspace size S_ij for j th representative gesture is M/N×S_th
Calculate the theoretical gesture size S_th under Sakoe-Chiba band
Step 3
By examine the fraction of weak subspace gestures in total randomly generated trials, we apply Monte Carlo method to estimate the size of weak subspace for all the representative gestures of the different symbols.
j<K
Y
j=j+1
N
Weak subspace size S_i for i th symbol class is sum of S_ij , j=1 to K
i<P
Y
i=i+1
N
Final weak subspace size S_final for P classes of symbols is sum of S_i , i=1 to K
Check overlaps between representative gestures in P classes of symbols
Step 4
Since each representative gesture covers an region by the upper and lower bounds, we need to check if there is overlaps between the covered regions by different representative gestures to prevent overestimation on the size of weak subspace.

Figure 4.4: The procedure to estimate sizes of the weak subspace based on data from Dictionary dataset. Step 1 is *Extract Representative Gestures*, which uses K-means and Silhouette value to extract representatives for gestures of each symbol; Step 2 is *Restrict Search Region*, which finds Sakoe-Chiba bands for each representative gesture; Step 3 is *Weak Subspace Estimation*, which is accomplished by Monte-Carlo simulation; Step 4 is *Check Overlapping Regions*, which removes potential overlaps between representative gestures' Sakoe-Chiba bands.

of K for gestures within a specific symbol. Finally, Silhouette outputs the centroids of K gesture clusters of all gestures under one symbol as the K representative gestures.

Restrict Gesture Searching Region

To estimate the size of weak subspace based on representative gestures for a specific symbol, we need to find the number of similar gestures for each representative gesture.

We use the Sakoe-Chiba band of one representative gesture to restrict the searching region and find out similar gestures for the representative. With the Sakoe-Chiba band, which restricts the upper and lower band of the search region, we can calculate the size of theoretically possible gestures, S_{th}.

Estimate Weak Subspace Size

Even with the Sakoe-Chiba band restriction, there are still millions of possible gestures that can compose one representative gesture. As it is too expensive to compute the similarities of all of the representative gestures, we use a Monte Carlo method as an alternative estimation approach for this situation. First, we randomly generate N gestures within the Sakoe-Chiba band of a specific representative gesture. Specifically, we randomly generate the first gesture sample point and randomly generate the following points under the upper and lower bounds of Sakoe-Chiba band. Then, we measure the DTW distance, D, between the generated gestures and the representative gesture, and compare D to the authentication threshold T. If $D > T$, it means the gesture is not similar to the representative. If $D < T$, the gesture could be regarded as a similar gesture to the representative. We count the number M of similar gestures for the representative. Finally, we estimate the size of the weak subspace for one representative gesture by combining the ratio M/N and the theoretical

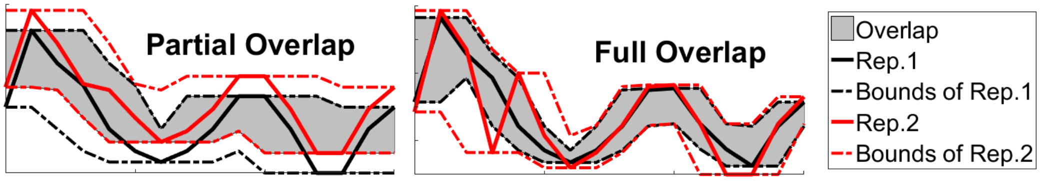


Figure 4.5: Examples of two types of overlaps in weak subspace enumeration. The left is partial overlap and the right is full overlap. The grey area is the overlap area where double-counting can happen.

size S_{th} of possible gestures within the Sakoe-Chiba band; this is $M/N \times S_{th}$.

By summation of the K representative gestures' weak subspace sizes for one specific symbol, we get the weak subspace size for a gesture symbol, S_i. Lastly, by adding the P classes of weak subspace sizes of different symbols, we calculate the final size of weak subspace S_{final} for gesture passwords.

Check Overlapping Regions

We use the Sakoe-Chiba band, which is an area around one gesture, to restrict the searching region for similar gestures for a single representative gesture. If the Sakoe-Chiba bands of two extracted representative gestures overlap, there is a possibility that the gestures in the overlap region could be double-counted by the two representative gestures as weak subspace gestures.

There are two types of overlaps between two representative gestures' bounds: (1) Partial Overlap, shown in the left of Figure 4.5, is when the bounds of $Rep.1$ and $Rep.2$ partially cross over; (2) Full Overlap, shown in the right of Figure 4.5, means that the bounds of $Rep.1$, fully fall into the area of the bounds of $Rep.2$.

We pairwise checked the representatives gestures' bands to examine whether there are any overlaps between them. We found that even among the representative gestures of the same symbol, there are no overlaps among the representative's bounds.

In summary, we have the estimated the size of the weak subspace for each gesture group and the total size of the weak subspace following the four steps above. Although we used DTW to measure the similarity between gestures, DTW can be easily extended to other time sequence similarity measurements, the only two differences are the similarity measurement in K-means and the authentication threshold T.

4.2.5 Guessability Analysis

We present our attack method based on the dictionary of the gestures in the weak subspace. We also present a baseline brute-force attack method, which we will use for comparison.

Dictionary Attack Method

The order of our attack is based on the order of total weak subspace gestures in each group. The more weak subspace gestures present in a group, the more likely it is that this gesture group will be selected. We select a number of weak gestures in the gesture groups instead of the number in a particular gesture pattern as the index for the attack order. The reason is that, since the size of the Dictionary dataset is relatively small, there is a limited number of gestures of a specific symbol meaning.

Dictionary Attack Gesture Generation Method. Generating a dictionary attack gesture uses the same process shown in Figure 4.4. The specific steps are:

1. The attacker selects a representative gesture in a gesture group and calculates its Sakoe-Chiba band. As a reminder, a representative gesture is extracted using K-means with a Silhouette value.

2. The attacker randomly generates an attack gesture under the representative gestures Sakoe-Chiba band.

3. Attacker measures the similarity between the attack gesture and all of the target gestures under attack. If the similarity falls under the authentication threshold, the corresponding gesture password is guessed correctly (cracked).

4. Attacker repeats the above steps for another guess.

Benchmark: Brute Force Attack Method

Previous work has shown how, for Draw-A-Secret (DAS) graphical passwords, the weak subspace can be enumerated by using symmetric features and a small number of distinct strokes (Jermyn et al., 1999). Based on prior findings from DAS (Thorpe & van Oorschot, 2004) and Pass-Go (Tao & Adams, 2008) graphical passwords, we extended these ideas for our baseline brute force cracking attack. We define the following heuristics: (1) users are more likely to create symmetric horizontal or vertical axes at or near the middle of a screen; (2) users tend to choose small number of distinct strokes. The small number of distinct strokes assumed for graphical passwords is not applicable to the multistroke case for gesture passwords since we connect a multistroke gesture end-to-end in the preprocessing procedure. As a result, both unistroke and multistroke symmetric gestures should be globally symmetric. Figure 4.6 (a) shows an example of central symmetric axis.

Brute Force Attack Gesture Generation Method.

The specific steps for the attack are as follows:

1. Randomly generate the eight X- and Y- coordinates of the first half of the gesture in

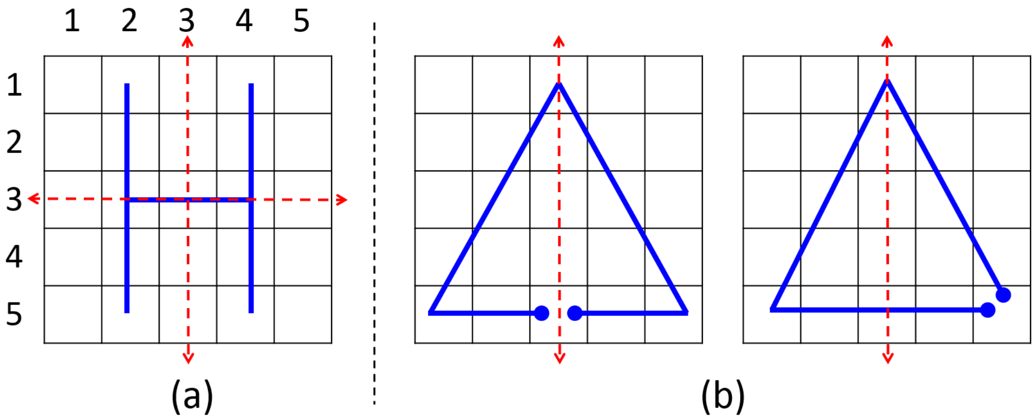


Figure 4.6: Different cases for a symmetric axis. (a) A gesture symmetric by an axis in the middle of screen. It shows the two global symmetric axis in our analysis. (b) Two different locations of a symmetric axis in symmetric gestures. They illustrate that the symmetric axis does not need to be located at the center of gesture sequence.

the entire space.

2. Randomly select the gesture's symmetric axis: vertical or horizontal.
3. With the half gesture and symmetric axis, generate the attack gesture sequence.
4. Relocate the symmetric axis along the gesture sequence. Randomly select a starting point in the gesture, then cut the gesture at the new starting point and concatenate the gesture's old end points.
5. Measure the similarity between the attacking gestures and all of the gestures under attack. If the similarity is under the authentication threshold, the corresponding gesture password is cracked.

Figure 4.6(b) shows that the symmetric axis does not need to be located exactly at the middle point of the gesture sequence. Although a symmetric axis in the rightmost of Figure 4.6 is not located at the midpoint of the gesture sequence, its gesture shape is still symmetric and it still should be considered as a symmetric gesture. As a reminder, we resampled all gestures to 16 points in preprocessing so the attacking method only needs to generate 16 points.

4.3 Results

In this section, we first present the sizes of full gesture space and weak subspaces. Then, we show the cracking results of the dictionary attack and the brute force attack on Test dataset (General) and Test dataset (Weak).

4.3.1 Sizes of Full Space and Weak Subspace of Gesture Passwords

By running Algorithm 1, we computed the size of the full theoretical gesture password spaces: 109.0 bits for *orientation method I* and 109.1 bits for *orientation method II*. Since the difference between the full space sizes for the two orientation methods is small, 109.0 bits vs. 109.1 bits, we conclude the variability in choosing either orientation method in gesture has little effect on the size of the space.

Table 4.3 shows the size of the weak subspace for two different orientation invariance methods based on our group clustering. The sizes of the weak subspace (both about 82 bits) are much smaller than the full space (both about 109 bits).

We found that the Geometric Shape group has the largest weak subspace and the Digit group has the smallest weak subspace. The larger the size of the subspace, the harder it is to crack it. We can predict that passwords in the Digits group are the easiest to crack while the passwords in the Geometric Shape group are the most difficult to crack among the seven groups.

By comparing the weak subspace sizes between the two orientation invariance methods, we found the difference between the methods on the weak subspace size is neglectable.

Group Name	Weak Gesture #	Orientation Invariant (Bits) Method I	Method II
Digit	37	67.1	67.7
Geometric Shape	181	82.0	81.4
Letter	114	75.3	77.3
Math Function	37	77.0	76.8
Math Symbol	18	71.6	74.6
Music Symbol	11	77.1	78.4
Special Char.	9	75.4	72.4
Total	407	82.1	81.7

Table 4.3: The sizes of the weak subspaces for free-form gestures. "Weak Gesture #" is the number of gestures that people selected as passwords. Compared to the size of the full gesture space (about 109 bits), the weak subspace (about 82 bits) is much smaller. Furthermore, the Geometric Shape group occupies the largest part of the total weak subspace size while Digits group has the smallest weak subspace size.

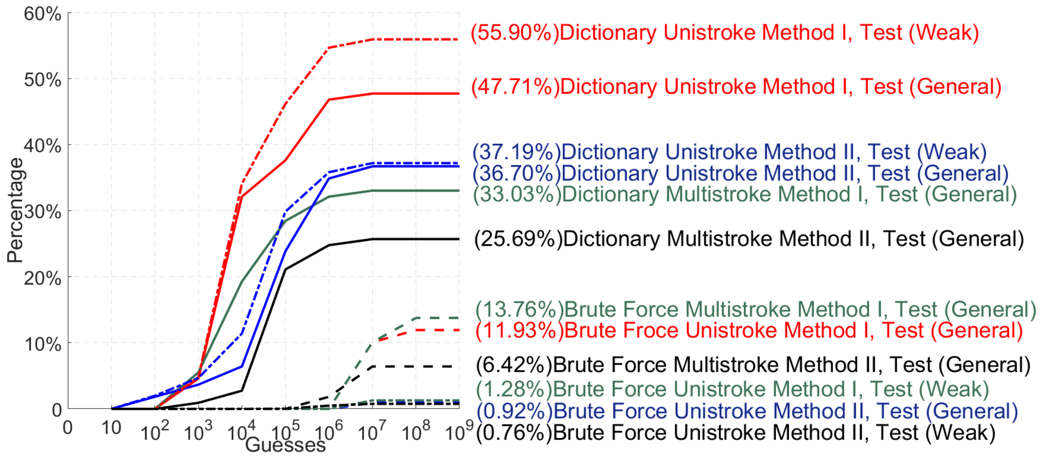


Figure 4.7: Cracking results for the dictionary and brute force attacks against free-form gestures (Test (General)) and specifically collected weak subspace gestures (Test (Weak)). For Test (General), by comparing the cracking rates, we find that our dictionary attack is more efficient cracking gestures compared to a brute force attack. The orientation invariance method I is more vulnerable to attacks than method II. For Test (Weak), a comparison between our dictionary and brute force attacks is meaningless since the dataset is specifically targeted for testing the dictionary set of weak subspace gestures.

4.3.2 Cracking Evaluation on Free-Form Gestures

To examine the performance of the dictionary attack and the brute force attack, we perform both attacks on gestures in Test dataset (General), which consists of free-form unistroke and multistroke gestures that users selected as their passwords. Figure 4.7 shows the cracking results of the two attacks based on the two orientation invariance methods.

Between Different Attacking Methods

Figure 4.7 shows that our dictionary attack is more efficient than the brute force attack. With both orientation methods, the dictionary attack cracked 35.78 percentage points more unistroke gestures and 19.27 percentage points more multistroke gestures than the brute-force attack. It is interesting that the difference between the two orientation methods does not influence different cracking rates between the attacking methods. By inspection, there does not appear to be a correlation between the gestures cracked under either orientation method and the the type of attack method.

The dictionary attack targets gestures inside the weak subspace, while the brute-force attack targets gestures with symmetric features. By checking the gestures with the particularly targeted features alone, the dictionary attack is more efficient at cracking weak subspace gestures compared to the brute-force attack cracking symmetric gestures. Figure 4.7 shows that the dictionary attack cracked 47.71% of unistroke gestures, which means 52 out of 109 gestures were cracked. Table 4.1 shows there are totally 78 weak subspace unistroke gestures in the Test dataset (General), so we find that 66.67% of weak subspace unistroke gestures were cracked. Similarly, the brute-force attack cracked 11.93% of unistroke gestures, that is 13 out of 109 unistroke gestures. Since there are 36 unistroke symmetric gestures in Test dataset (General), it means 36.11% of unistroke symmetric gestures were cracked. Therefore, the dictionary attack is more efficient at cracking weak subspace gestures than the brute force attack at cracking symmetric gestures.

Between Different Orientation Invariance Methods

Figure 4.7 shows that gestures with orientation invariance method II resists attacks better than method I. Both dictionary and brute force attacks cracked 11.01 percentage points more unistroke gestures and 7.34 percentage points more multistroke gestures with orientation method I than method II. It happens again that the difference between the two attacking methods does not influence the cracking rates between two orientation methods. By checking the cracked gestures, we still did not find any relationships between the orientation methods and attack methods. To conclude, the selection of the orientation invariance method has an effect on the gesture's ability to resist attacks.

Between Different Gesture Types

As Figure 4.7 shows, multistroke gestures resist dictionary attacks better than unistroke gestures. With both orientation methods, there are more than 10 percentage points of unistroke gestures that were cracked than multistroke gestures.

Under brute-force attack, there is no obvious difference between cracking rates for unistroke and multistroke gestures. Additionally, the absolute numbers of cracked unistroke and multistroke gestures are small. We cannot conclude that multistroke gestures are weaker than unistroke gesture on resisting brute-force attacks.

Note on Computational Limitations

The computational cost of measuring the similarity between gestures is a crucial limiting factor in our test. Unlike text passwords presented in previous work (Florencio & Herley, 2007; Bonneau, 2012b; Ur et al., 2015), where checking if a password is right or wrong is near instant, gesture passwords requires dynamically searching for the DTW distance. The

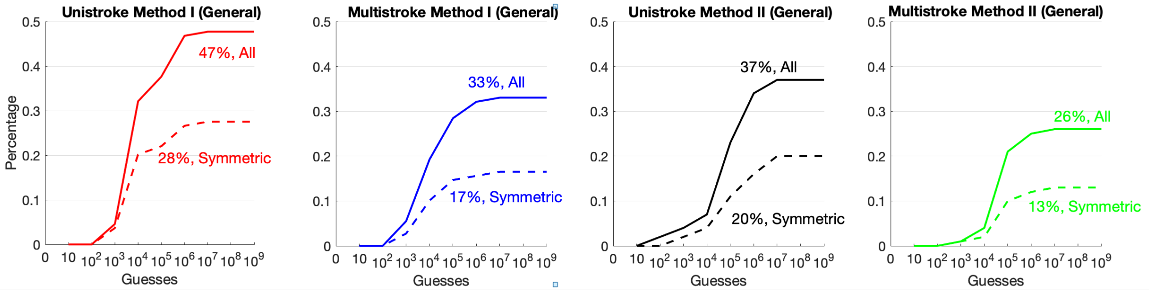


Figure 4.8: Ratio of cracked symmetric gestures with different orientation invariance methods in Test dataset (General). The cracked gestures with symmetric feature occupied about half of the cracked gestures in the Test dataset

computational complexity for text password matching is $O(N)$ and for gesture matching is $O(N^2)$, with N being the password length. We spent two weeks performing 10^9 attacks against 3488 gestures, that is 218 gestures in Test dataset 1 and 3270 gestures in Test dataset 2. Assuming we want to attack the same amount of text and gesture password with the same length, the gesture password attack needs to spend $3488N^2/N \approx 10^3 N$ times more computation than text passwords. However, text based password systems can use approaches such as scrypt (Percival, 2009) to slow down the verification process.

4.3.3 Crack Evaluation on Free-Form Gestures with Symmetric Feature

We learned the brute force guessing attack is less effective than the dictionary guessing attack. We also know the brute force attack method is based on the symmetric features of gestures. So we want to examine the ratio of cracked gestures with symmetric features to see if the low cracking efficiency of the brute force attack came from the ratio of gestures with symmetric features.

Figure 4.8 shows that around half of the cracked gestures in the Test dataset (General) have symmetric feature. It means that the low guessing attack efficiency of brute force attack does not come from the low ratio of symmetric gestures in the dataset.

4.3.4 Crack Evaluation on Weak Subspace Gestures

The above results show the dictionary attack outperforms the brute force attack. To examine the effectiveness of our dictionary attacks on cracking specifically weak subspace gestures, we perform the dictionary attack on Test dataset (Weak), which contains only weak subspace gestures that were intentionally covered by our gesture dictionary.

Figure 4.7 shows the cracking result for dictionary and brute force attacks on weak subspace gestures in Test dataset (Weak). The dictionary attack had a 55.9% cracking rate with orientation method I and 37.19% cracking rate with method II. The brute force attack had 1.28% and 0.76% cracking rate for the two orientation methods.

We observe the cracking rates using dictionary attacks in Test dataset (Weak) (55.9% and 37.19%) is higher than that rates in Test dataset (General) (47.71% and 36.70%). The reason is that Test dataset (Weak) intentionally contains only the weak subspace gestures, while Test dataset (General) contains both weak subspace gestures and other freely created gestures. This means Test dataset (Weak) performance is by default biased towards favoring the dictionary attack. Therefore, conclusions about the performance of the brute-force attack compared to the dictionary attack should be limited to the Test dataset (General).

Figure 4.9 shows the details of cracked gestures in Test dataset (Weak). The Digit gesture group is the most easily cracked while the Special Character gesture group is the most resistant. As Figure 4.9 shows, only 21.37% of Digit gestures can resist dictionary attacks. A possible reason is that Digit group has the smallest weak subspace (about 67 bits) in Table 4.3. The principal reason, however, is that there is not much observed variation in the way people perform gestures in the Digit group. In contrast, 52.99% of special

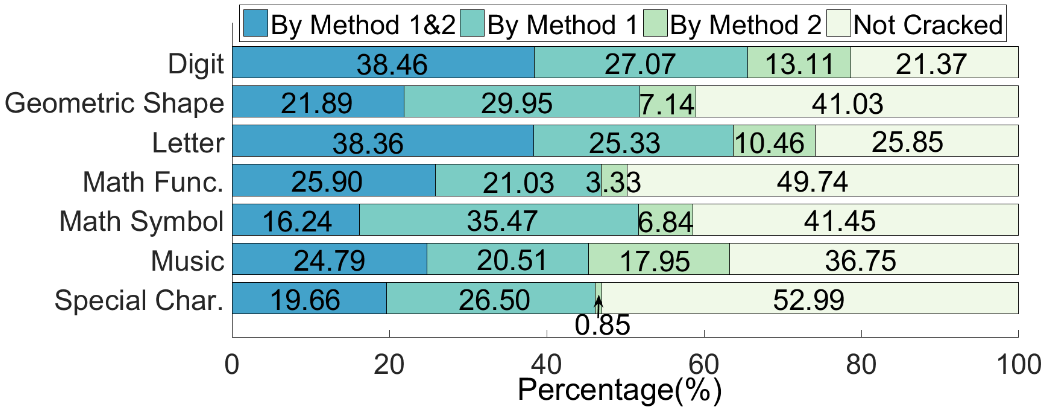


Figure 4.9: Categories of cracked gestures with different orientation invariance methods in Test dataset (Weak). The Special character group is the strongest and Digit group is the weakest on resisting the dictionary attacks.

character group gestures are not cracked by dictionary attacks. This can be explained by its large weak subspace size (about 75 bits) and smallest number in Table 4.3. The larger weak subspace size makes gestures more difficult to be cracked and the smaller number of weak gestures prevents the attack dictionary from covering all possible variations of the way people draw certain gestures.

4.4 Summary

Here, we have presented the first work on realistic guessing attacks against free-form gesture passwords. We developed a methodology for performing guessing attacks, showed how to enumerate the size of both the full and weak subspaces for gesture passwords, identified and categorized a list of most commonly used gestures from published work on gesture passwords, and recruited participants to obtain additional data for testing our method. We positioned guessing attacks against gesture passwords by creating a generalizable method based on how gestures are recognized. We extended this attack method for a dictionary attack based on common gestures, and we contributed a brute force attack, based on symmetric features, to use as a benchmark measurement.

CHAPTER 5

SECURITY ANALYSIS OF GESTURE PASSWORDS

5.1 Overview of Chapter

In this chapter, we demonstrate the challenges of evaluating the security of gesture passwords, and proceed to explain approaches to address those challenges step-by-step.

First, we present the challenges on quantitatively measure gesture passwords' security.

Then, we introduce the passwords dataset for our analysis, including gesture passwords, signature passwords, Android pattern unlocks, and text passwords.

Third, we present the details of discretizing the gesture passwords using SAX to represent the time series values by discrete symbols.

Fourth, we describe how to estimate the passwords distribution based on Markov Chain and especially how to estimate the upper bound of lower bound of passwords distribution.

Lastly, we introduce the partial guessing metric to measure the password security based on its distribution.

5.2 Challenges

This section contains descriptions of the logic involved in the later parts of the paper, so as to help the reader better understand the presentation order of the paper and the meaning behind certain choices that were made. The technical details of our work follow immediately after this section. We will present our discussion of challenges in the following order:

(1) How does an attacker behave to crack passwords?

(2) How to estimate the baseline for the security of recognition passwords?

(3) How to show the security estimation of recognition passwords is generalizable?

(4) How to assign the probability of a given password?

(5) How to deal with the recognition passwords that are not covered by current available datasets?

5.2.1 Attacker Behavior: The Threat Model

There are multiple ways an attacker can behave when targeting passwords. Attacker behavior can change based on the amount of information the attacker has and his/her overall objective in trying to crack a specific password. An attacker, for instance, may concentrate effort on trying to crack a single password without a thought to the many other passwords in the set. In this vein, an attacker could use observations to gain information about the password, data mine a particular user to obtain ideas about what the password might contain, or attempt to steal the password through other means.

This targeting behavior is not useful for trying to evaluate the general security of an authentication method, since it is not feasible for an attacker to exert this same level of effort for a massive group of users. In light of this, Bonneau (Bonneau, 2012c) outlines a framework for evaluating the security of text passwords based instead on the entire password distribution. In this framework, an attacker has access to a large number of accounts and is interested in maximizing the benefit of cracking a given account while minimizing the cost of cracking the same account. Bonneaus attacker does not have any prior information about the target user. One might imagine that this situation is akin to obtaining a large list of email addresses and trying to crack the password for each one of those emails. The attacker applies a number of guesses to each password, while also trying to crack some percentage of the entire account set. By stopping after a fixed number of guesses, the attacker does not waste resources on accounts with difficult passwords. By stopping after cracking some percentage of the targeted accounts, they will have achieved some minimal benefit.

Secret selection by humans is usually biased, creating a subset of more likely passwords in the distribution that we refer to as the weak set of passwords. An attacker who is informed about the password distribution is capable of ordering their guesses from most likely to least likely. Therefore, password systems with a more concentrated distribution are more likely to be cracked and thus are less secure. Figure 5.1 (a) illustrates three types of password systems with different distributions. An ideal password system would have a uniform distribution, with all passwords being equally likely, to prevent an attacker from gaining an advantage by ordering attacks. The security of a password system is tied to the distribution of user choice.

Given this attacker behavior, the security metrics of biometric systems like True Positive Rates and False Positive Rates across a dataset are misleading for assessing the security of recognition passwords. Unlike passwords, typical biometric systems do not involve a component of user choice. They are based instead on immutable human characteristics that are highly differentiable – fingerprints, for example – and cannot be changed at will. The primary security mechanism for biometrics is the high degree of separation among thousands of people. Therefore, it makes sense to discuss security with TPR and FPR for biometric systems. However, for a recognition password, security is primarily derived

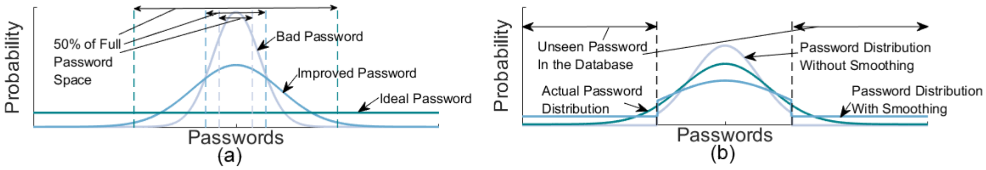


Figure 5.1: Password distributions that are affected by different factors. The left figure (a) shows the distributions that affected by user selections. The ideal password distribution is uniform, in which all passwords are equally probable. A weak password distribution usually has a narrow peak, meaning most user passwords fall in a small set: the weak set. An informed attacker would use the most probable passwords in the weak set when guessing. We can improve the passwords security by broadening the set that people are more likely to select from. The right figure (b) shows the distributions with different methods for addressing the unseen passwords issue. The actual passwords distribution represents the real passwords distribution. The distributions with and without smoothing methods represent the passwords distribution with the two methods for addressing the unseen passwords issue.

from the choice of the secret– that is, how likely it is that an attacker will select the same password. TPR and FPR measure the accuracy of a recognizer in separating out obviously different passwords; they are not a statement of security. The important metric here is the distribution of the user choice of passwords, and we therefore need an efficient recognizer to estimate that distribution.

5.2.2 Baseline of Security Estimation of Recognition Password

We now know how an intelligent attacker will behave towards a large number of accounts: attackers order their attacks efficiently so as not to waste effort on accounts that are too difficult to crack. Specifically, the attacker should select the most likely passwords and move from there, and then should try different users' passwords instead of trying different variations of the same user's password. Both of these needs lead to one question: how can one transform all variations of a given password into a simple representation?

The main challenge here is that recognition passwords are a many-to-one mapping.

Since discretizing the recognition passwords maps one users many password variations to one representation, it essentially simplified the password space by decreasing the number of possible passwords. In other words, security analysis based on the discretized password space provides a baseline of the password security in the original recognition passwords.

In our paper, we perform the discretization transformation using Symbolic Aggregate ApproXimation (SAX), which transforms time-series data into a symbolic set. After SAX transformation, passwords that are similar to each other become the same character string. In this way, it is easy to group the passwords together.

5.2.3 Generalizable Discretizing Approach for Recognition Passwords

SAX is a generalizable discretizing approach for recognition passwords since (1) it uses the time series data to recognize the different users' passwords and (2) it maintains the same distinguishability of recognition passwords as the other commonly used recognizers. First, the time series similarity data matching is widely used in recognition passwords (Sherman et al., 2014; Y. Yang et al., 2016; Clark, Lindqvist, & Oulasvirta, 2017; Liu, Clark, & Lindqvist, 2017a; Tian et al., 2013; Aslan et al., 2014; Aumi & Kratz, 2014; Liu, Clark, & Lindqvist, 2017b). Since SAX represents the time series data of recognition passwords via a series of disretized symbols and matches the similarity based on the sequences of discretized symbols, it also can be used to estimate the baseline of security for all recognizers that also utilize time series data similarity matching. Second, distinguishability between recognition passwords should be the crucial measurement to evaluate the recognition performance of the discretizing approach. We used Area Under ROC curve (AUROC) to show that SAX achieves the comparable ability at distinguishing the genuine users as the other commonly used recognizers, such as Protractor (Sherman et al., 2014; Y. Yang et al., 2016; Clark et al., 2017), DTW (Liu et al., 2017a; Tian et al., 2013; Aslan et al., 2014; Aumi & Kratz, 2014), Garda (Liu et al., 2017b).

5.2.4 Enumerating and Assigning Probabilities to the Passwords

As we represent the long time series data of recognition passwords with short discrete SAX symbols, we are capable of enumerating the distribution of the entire password set by listing all possible combinations of strings together. However, assigning probabilities to these newly generated passwords remains an issue.

The representation of a password as a string has benefits besides solving the counting problem: it can be used in combination with Markov chains. The guiding principle behind a Markov chain is that the next symbol in a human-chosen string depends on some number of the previously chosen symbols. This logic comes from the intuition that, when given a partial text string like $gestu...$, it is highly likely the whole string is $gesture$.

To crack a password or multiple passwords in this way, an attacker transforms every recognition password in a prior dataset into a text string. Then, the attacker trains the Markov chain using these text strings to estimate the probabilities for every possible string. This estimation method has been used in the past with text passwords (Ma et al., 2014; Narayanan & Shmatikov, 2005) and Android unlock patterns (Uellenbeck et al., 2013; Song et al., 2015). Thus, the attacker can estimate the probabilities of all passwords and deploy guessing attacks based on the estimated probabilities.

5.2.5 Estimating the Security Based on Datasets Cannot Cover All Passwords

One of the key difficulties in generating Markov chain estimates of a password space distribution is completeness. A model is complete when it assigns a non-zero probability to all possible passwords. If a model is incomplete, then some passwords have zero probability – we call these *unseen* passwords. When the completeness of a model is poor, the estimated password distribution will skip over passwords that are likely to be selected by people but are not covered by the trained Markov chain model.

There are two factors that may lead passwords to remain unseen. First, the passwords may be very unlikely to be selected by people. Second, the passwords may be likely to be selected by people, but the dataset may not cover them. Based on these two reasons, we used two strategies to deal with the covered passwords: (1) leaving the unseen passwords as zero probability, and (2) assigning small fixed probabilities to the unseen passwords. The first method eliminates impossible passwords, and the second method avoids a situation in which potential passwords are skipped. Figure 5.1(b) illustrates the password distribution that results after each of these two methods is applied. The password distribution that results from the application of a smoothing method has a peak wider than the actual password while the distribution that appears without smoothing has a peak narrower than the actual password. This means that the model of password distribution without a smoothing method underestimates password security, while the model that does use a smoothing method overestimates security. For matching passwords, we can directly apply Markov model on the passwords. The two approaches of dealing with the unseen passwords are able to provide both the upper bound and lower bound estimation of the matching passwords security.

For recognition passwords, the lower bound estimation of the security of recognition passwords by regarding the unseen passwords as zero probability provides another baseline of the security of recognition passwords. To summarize, the baseline of the security of recognition password is formed by two steps: (1) Discritizing the recognition password by SAX; (2) Leaving the unseen passwords in password space with zero probability.

5.3 Password Datasets

We did not conduct new studies to collect data. Instead, we aggregated the largest available gesture dataset and a large signature dataset to analyze recognition passwords. We requested and used gesture datasets from the following studies: FreeForm (Sherman et al., 2014), Wild (Y. Yang et al., 2016), and GuessAttack (Liu et al., 2017a). In each study, the participants are asked to create accounts with gestures as passwords. The gesture passwords for each account need are replicated at least once. In total, the datasets contain 2656 gesture password samples from 655 types of gesture passwords. The signatures dataset includes three publicly available datasets: SUSig (Kholmatov & Yanikoglu, 2009), MCYT-100 (Ortega-Garcia et al., 2003), and SVC2004 (Yeung et al., 2004). Signature contains 5180 signature samples from 234 participants. Table 5.1 summarizes the datasets.

To analyze Android unlock patterns as a reference for matching passwords, we requested and obtained 113 defensive Android unlock patterns and 573 offensive Android unlock patterns from prior study with 113 participants (Uellenbeck et al., 2013). Participants were asked to create Android unlock patterns that are difficult for other to crack as defensive patterns. Similarly, the participants created Android unlock patterns they felt were most likely to crack other users' patterns as offensive patterns.

Dataset	Participant #	Password #
FreeForm (Sherman et al., 2014)	57	57
Wild (Y. Yang et al., 2016)	47	380
GuessAttack (Liu et al., 2017a)	109	218
SUSig (Kholmatov & Yanikoglu, 2009)	94	94
MCYT-100 (Ortega-Garcia et al., 2003)	100	100
SVC2004 (Kholmatov & Yanikoglu, 2005)	40	40
Android (Defensive) (Uellenbeck et al., 2013)	113	113
Android (Offensive) (Uellenbeck et al., 2013)	113	573
Yahoo! (Blocki, Datta, & Bonneau, 2016)	-	69301337

Table 5.1: Summary of analyzed gesture and signature datasets. The top three datasets are gestures, the following three datasets are signatures. Yahoo! is a text password dataset and the rest two are Android datasets.

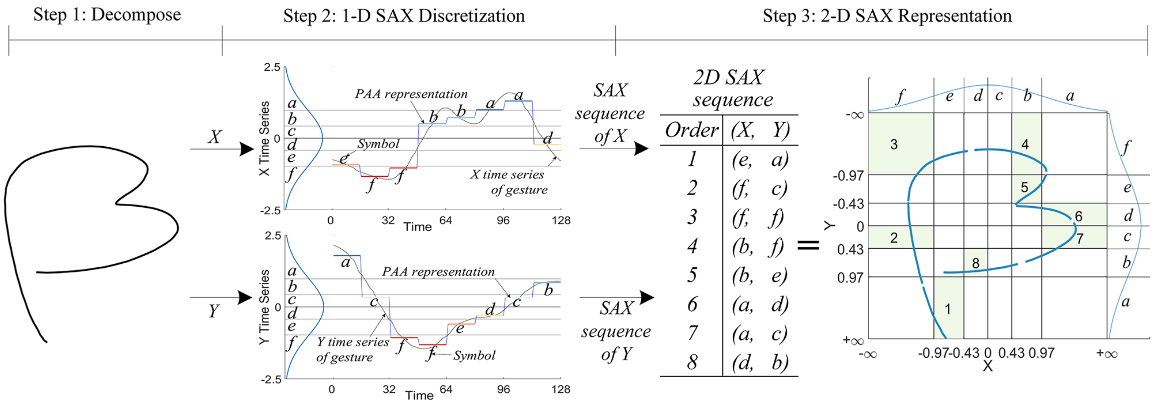
5.4 Discretization of Recognition Passwords

In this section, we describe the process of representing recognition passwords with SAX. We discuss the many-to-one mapping of gestures to SAX symbol sequences. Then, we discuss the how to use distinguishability between recognition passwords to determine the parameters used in 2-D SAX.

5.4.1 Represent by 2-D SAX

We used Symbolic Aggregate approXimation (SAX) (Lin, Keogh, Lonardi, & Chiu, 2003) to normalize and discretize the time sequence data of a recognition password so that it can be represented as short sequence of symbols. SAX uses a sequence of symbols with a fixed length ω to represent a time series data of length n, where $\omega \ll n$ (Lin et al., 2003). Figure 5.2 shows the steps for approximating a recognition password with 2-D SAX.

1. Decompose and normalize password. We decomposed a gesture into two 1-D time sequences: X time series and Y time series. The time series of X and Y are normalized to be zero mean and unit standard deviation.



2D SAX sequence

Order	(X, Y)
1	(e, a)
2	(f, c)
3	(f, f)
4	(b, f)
5	(b, e)
6	(a, d)
7	(a, c)
8	(d, b)

Figure 5.2: Illustration of representing a recognition password symbolically with 2-D SAX. First, the password is decomposed into X and Y 1-D coordinate time sequences. In Step 2, each time sequence is normalized to have its mean set to zero and the standard deviation equal to one. The time sequence is then evenly segmented using Piecewise Aggregate Approximation into eight subsequences. SAX then maps the means of the eight subsequences into the six symbols: *a,b,c,d,e,f*. The boundaries of the six symbols are calculated using the normal distribution on the left, where each symbol is defined to have equal probability. In Step 3, we combine the SAX sequence of X and Y to form a 2-D SAX sequence. The 2-D SAX sequence can be represented as a 2-D map as seen in the rightmost figure.

2. PAA representation. We approximate a password's 1-D time sequences by Piecewise Aggregate Approximation (PAA) (E. J. Keogh & Pazzani, 2000). PAA approximates a time series by segmenting it into ω equal-length subsequences and representing each subsequence by its mean.

3. 1-D SAX representation, we use SAX (Lin et al., 2003) to map the value of PAA representation into different symbols based on the partitioned ranges. Each value range is chosen to have the same probability according to the normalized distribution, which is an important step to guarantee that each symbols are chosen with the same probability. In our example in Step 3 of Figure 5.2, the normal distribution is divided into six equal probabilities ranges with five boundaries, {*-0.97, -0.43, 0, 0.43, 0.97*} and the six ranges are represented by six symbols, {*a,b,c,d,e,f*}. The five boundary

$$MINDIST(\hat{Q},\hat{C}) = \sqrt[n/\omega]{\sum_{i=1}^{\omega}(dist(\hat{q}_i,\hat{c}_i))}$$

$$dist(\hat{q},\hat{c}) = \begin{cases} 0, & \text{if } |\hat{q}-\hat{c}| \leq 1 \\ \beta_{max(\hat{q},\hat{c})} - \beta_{min(\hat{q},\hat{c})}, & \text{otherwise} \end{cases}$$

	a	*b*	*c*	*d*	*e*	*f*
a	0	0	0.54	0.97	1.4	1.94
b	0	0	0	0.43	0.86	1.4
c	0.54	0	0	0	0.43	0.97
d	0.97	0.43	0	0	0	0.54
e	1.4	0.86	0.43	0	0	0
f	1.94	1.4	0.97	0.54	0	0

Table 5.2: MINDIST function and an example of lookup table for *dist()* in the function. In the MINDIST function, n is the length of the original time sequence and ω is the length of the time sequence represented by SAX. $dist()$ is used to measure the distance between two symbols of SAX. β is the set of boundaries of the symbols in SAX. The right lookup table is an example of dist() when there are six possible symbols in SAX (i.e. $\beta = 6$)

values are obtained by using the inverse-CDF of the standard normal distribution, which outputs the point at which a certain amount of probability is contained. If there are six equiprobable regions, then $CDF^{-1}(\frac{1}{6}) = -0.97$, $CDF^{-1}(\frac{2}{6}) = -0.43$, and so on. The point at which zero probability is contained is $-\infty$ and the point of all probability is ∞.

4. 2-D SAX representation. 2-D SAX is simply a combination of SAX sequences of X and Y. It also can be represented by a 2-D matrix on the rightmost graph in Figure 5.2 following the order of the cells. The rightmost graph shows the area of a gesture represented by 36 cells with different value combinations in X and Y coordinates and the gesture is chopped into eight pieces based on PAA. For each gesture piece, SAX used its means in X and Y coordinates to assign the 36 SAX cells.

We used and extended MINDIST function, which is defined for 1-D SAX (Lin, Keogh, Wei, & Lonardi, 2007), to measure the similarity of 2-D SAX representations of recognition passwords. With the original MINDIST, we measure the similarity of two 1-D SAX sequences $\hat{Q} = \hat{q}_1, ..., \hat{q}_\omega$ and $\hat{C} = \hat{c}_1, ..., \hat{c}_\omega$ as shown in the left function in Table 5.2.

According to MINDIST function, the distance is straightforward to calculate once the boundaries are known. Assume there are six symbols: a, b, c, d, e, f as seen in Figure 5.2. The five boundaries separating the six equiprobable ranges in the normalized distribution are $\{$*0.97, 0.43, 0, -0.43, -0.97*$\}$, according to the inverse-CDF of the normal distribution. Since $|a - b| = 1$, $dist(a, b) = 0$ and $|a - c| = 2$, $dist(a, c) = \beta_{max(a,c)} - \beta_{min(a,c)} = \beta_a - \beta_c = 0.97 - (0.43) = 0.54$. The *dist()* function can be implemented by a lookup table as shown in Table 5.2.

In our 2-D SAX, we define $\hat{Q}$ and $\hat{C}$ represent time sequences of D dimensions, instead of one dimension sequences in 1-D SAX. The distance between sequences is the sum of 1-D SAX distance in each dimension.

5.4.2 Determining Parameters in 2-D SAX

We have described how 1-D and 2-D symbols can be transformed into a symbolic sequence using SAX. However, there are important parameters that need to be determined rigorously in order to obtain the tightest bound possible on the password distribution. In order to discretize recognition passwords with 2-D SAX, we need to determine two parameters: the length of a symbolic sequence, ω, and the alphabet of symbols, β. Increasing the values of the parameters increases the overall size of the password space – by analogy, creating a new character for the Roman alphabet increases the space size for text passwords. The larger the parameters, the larger the difference between the different variations of the same password. Increasing the size of the parameters conflicts with the main goal of discretization, which is to narrow the difference between the variations of the same password. Thus we would like to minimize ω and β. However, if we make the parameters too small, we will reach a point where no passwords are distinguishable. Thus we need to choose the smallest possible parameters that meet some minimum criteria of distinguish-ability (e.g. circles and squares are still seen to be different). In this section, we will introduce Receiver Operating Characteristic (ROC) curve and the Area Under Receiver Operating Characteristic curve (AUROC) as evaluation metrics for ω and β. Then, we will present the optimal parameters for 2-D SAX.

Receiver Operating Characteristic (ROC)

The ROC curve measures the recognition performance of recognition password systems. It plots the true positive rate and false positive rate by tuning the recognition threshold from minimum to maximum.

Area Under ROC (AUROC)

Because the different recognizers are evaluated on the same datasets, we can evaluate their distinguishability by the Area Under ROC curve. AUROC reflects the probability that a randomly chosen true password is ranked higher than a false password (Fawcett, 2006). It measures the distinguishability between the positive and negative samples. The higher of the AUROC, the more distinguishable between the samples.

Optimal Values of Parameters ω and β

Based on our analysis, we found that the length of a symbolic sequence, $\omega = 8$, and the alphabet of symbols, $\beta = 6$ balances the AUROC. Figure 5.3 (a) shows that when $\omega \geq 8$ and $\beta \geq 6$, the AUROC of the SAX recognizer for both gestures and signatures does not change significantly. This implies that when $\omega \geq 8$ and $\beta \geq 6$, the passwords from different users are only slightly more distinguishable.

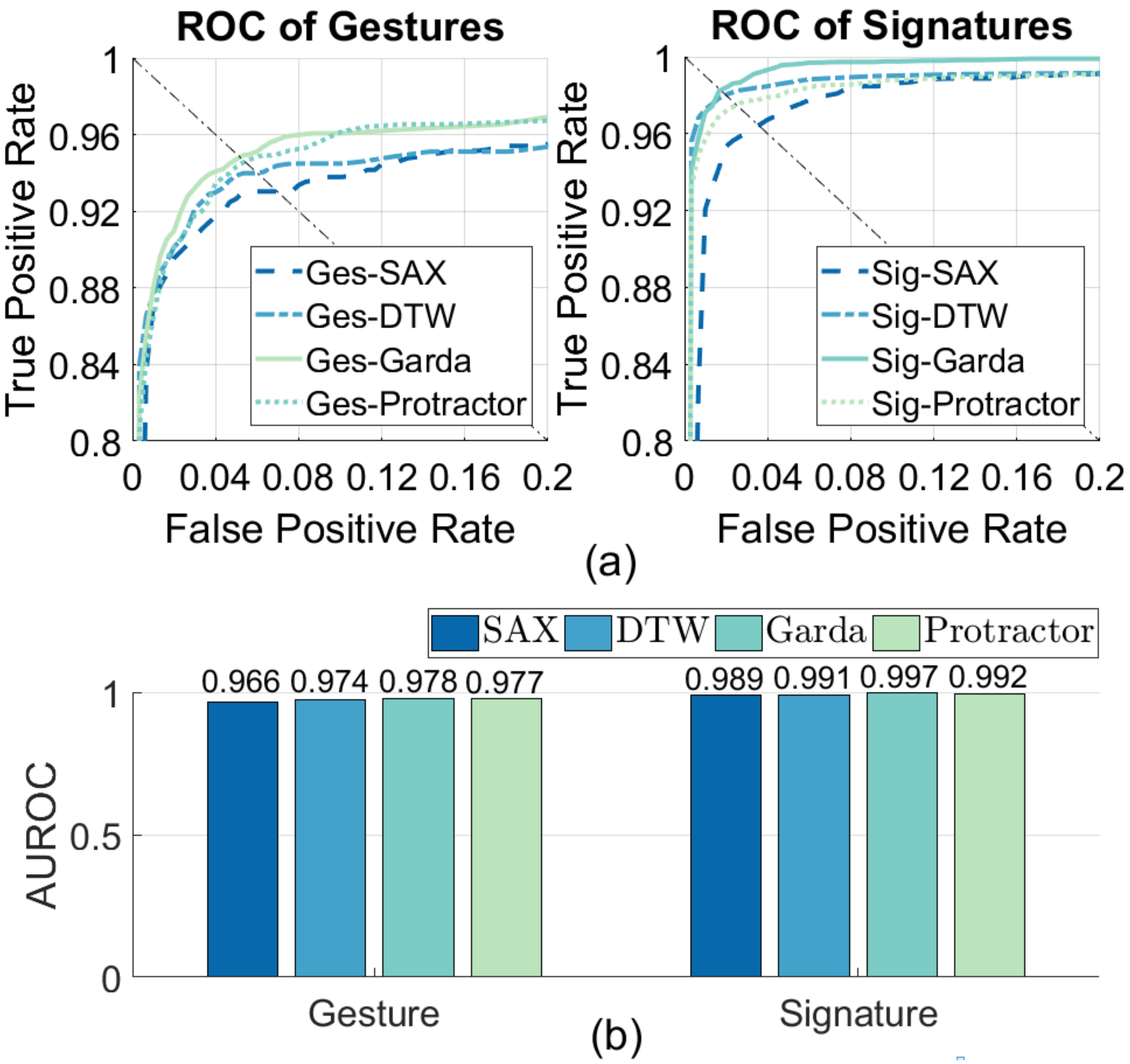


Figure 5.3: Grid search for the optimal parameters of SAX and ROC curves based on the optimal parameters of SAX. (a) Shows a contour plot showing isolines of AUROC for signature and gesture datasets based on different values of ω and β. When $\omega \geq 8$ and $\beta \geq 6$, the AUROC of SAX of gestures and signatures do not change by more than 0.001 and 0.015, respectively. The isolines are erratic since the AUROC is highly dependent on the combination of β and ω; these two parameters are not orthogonal. We observed the that AUROC gradually decreases when ω and β increase. (b) shows the ROC curves and corresponding AUROC of the four recognizers for gestures and signatures. We see SAX is only slightly worse than the other three recognizers and gets comparable values of AUROC to the other recognizers. The takeaway is there is no large difference between the four recognizers.

5.4.3 Recognition Performance of SAX

SAX strips information away from recognition passwords through discretization. The rightmost side of Figure 6 also shows that gestures are significantly distorted by SAX. The distinguishability of recognition passwords is a metric that shows whether SAX accurately models user behavior in drawing recognition passwords. We used the recognition performance of those discretized passwords with ROC and AUROC to examine if the discretized passwords still reliably distinguishable among users. If the recognition performance of SAX is as good as that of other state-of-the-art recognizers, we can conclude SAX is a valid discretizing method for recognition passwords.

The ROC curves for SAX have results comparable to those of other three recognizers. Figure 5.3 (b) shows signature and gesture ROC curves for the four recognizers: SAX, Protractor (Sherman et al., 2014; Y. Yang et al., 2016; Clark et al., 2017), DTW (Liu et al., 2017a; Tian et al., 2013; Aslan et al., 2014; J. Yang et al., 2015; Aumi & Kratz, 2014), and Garda (Liu et al., 2017b). Note that SAX is not meant to be the best recognizer – our goal is to demonstrate that the symbolic representation maintains enough detail to distinguish gestures well enough. In this regard, the fact that it has performance that is comparable to other recognition methods allows us to consider SAX a success in this case. Additionally, the AUROC values of SAX are close to those of the other three recognizers. It means SAX has an ability to distinguish positive samples from negative ones that is comparable to those of the other recognizers.

In summary, representing recognition passwords with SAX can reduce the password space while maintaining the distinguishability of passwords.

5.5 Recognition Password Distributions

We have mapped the recognition passwords to a countable password space using SAX. We now need to estimate the user-chosen password distribution. Collecting data is not enough to do this, as it is very difficult to collect millions of passwords to represent the distribution. With SAX, we can enumerate all possible string in the entire password space. A Markov chain will be trained with prior data to assign probabilities to the generated passwords.

5.5.1 Markov Chain

Markov chains have been used to estimate the distribution for both text passwords (Ma et al., 2014; Narayanan & Shmatikov, 2005) and Android unlock patterns (Uellenbeck et al., 2013; Song et al., 2015). The probability is computed as the product of conditional probabilities which represent the likelihood of transitioning from one symbol to the next in a sequence. These transition probabilities are estimated by their relative frequency of occurrence in a known data set. The guiding principle behind in a Markov chain is that the next symbol in a human-chosen string depends on some number of the previously chosen symbols.

An *n-gram* Markov chain predicts the next observation in a string based on the previous $n-1$ observations. To build Markov chains, there are two types of conditional probabilities that need to be estimated: 1) the probability of the starting symbol; 2) the probability of the transition to the next symbol given the previous $n-1$ symbols. For example, to build a 2-gram Markov chain for a sequence $s = \{s_1, s_2, ..., s_8\}$, we need to estimate: 1) the starting symbol probability $p(s_1)$; 2) the transition probability $p(s_{i+1}|s_i)\ i \neq 0$. Then, we can calculate the probability of the sequence s as $P(s) = P(s_1)P(s_2|s_1)...P(s_7|s_6)P(s_8|s_7)$.

5.5.2 Parameter Selection

To model the passwords based on an n-gram Markov chain, we need to determine n and the smoothing methods.

Selecting n

The value of n defines both the number of previous symbols on which a transition depends as well as the length of the start sequence. A larger value of n can yield more accurate predictions of the sequence by accounting for longer historical correlations. However, this gain in accuracy can also lead to over-fitting as many transitions may be assigned zero probability because the preceding sequence is never observed.

To determine a practical value of n for the amount of data we have available, we can estimate the how large the zero probability blocks are by computing the expected number of times a given start sequence will be observed as a function of n. To simplify this problem, we assume that each start sequence is equally likely. Therefore, the expected number of observations of any particular starting sequence is as $E(observation) = \frac{T}{(\beta^2)^{n-1}} = \frac{T}{(36)^{n-1}}$. T is the total number of passwords in the data set. For the gesture dataset, $T = 3245$, and for the signature dataset, $T = 5026$. When $n = 2$, a given start sequence is expected to be observed 90.14 and 139.6 times for the gestures and signatures respectively. When $n = 3$, the expected numbers for gestures and signatures drop to 2.50 and 3.88. When $n = 4$, they drop to 0.07 and 0.11. Since there are fewer than one observations for each start symbol when $n = 4$, it is clear that models with this depth or greater will assign zero probability to almost all passwords and are thus unusable. Thus, we only consider n-gram models with $n = 2$ and $n = 3$.

Smoothing

Aside from assigning a small value for n to deal with unseen passwords, a more common approach involves estimating the probabilities of the unseen passwords using smoothing methods. We tested two smoothing methods: 1) additive smoothing, and 2) Good-Turing smoothing (Gale & Sampson, 1995). Additive smoothing adds a constant small value λ to the counts of the Markov chain transition matrix. We assign λ to the unseen data, which are originally valued at zero, to make sure all theoretical possible data have some probability of occurring. Based on our tests, we selected $\lambda = 0.01$. Good-Turing smoothing uses the observed total probability of class $r + 1$ to estimate the total probability of class r. The total probability of class r is a class of probability of transitions from a symbol to another symbol that has occurred r times in total.

5.5.3 Optimizing Markov Chains for Recognition Passwords

We have proposed SAX to represent recognition passwords with discrete symbols, and we have used Markov chains to estimate the probability of recognition passwords. We cannot judge a priori which configuration of Markov chains is most suitable for modeling recognition passwords. The optimal Markov chain should be most efficient at cracking passwords. We will use guessing entropy (Cachin, 1997; Massey, 1994) to show the cracking efficiency of the Markov chains.

Guessing Entropy

Guessing entropy (Cachin, 1997; Massey, 1994) measures the average number of guesses required to crack an entire set of passwords $X = \{x_1, x_2, ..., x_N\}$ in the optimal guessing order. Specifically, a guessing entropy curve represents the percentage of a dataset that is cracked as the number of guesses increases. It reflects the strength of the target passwords. Generally, a given guessing entropy means that it takes an average of so many guesses to crack some proportion of the data (see Figure 5.4).

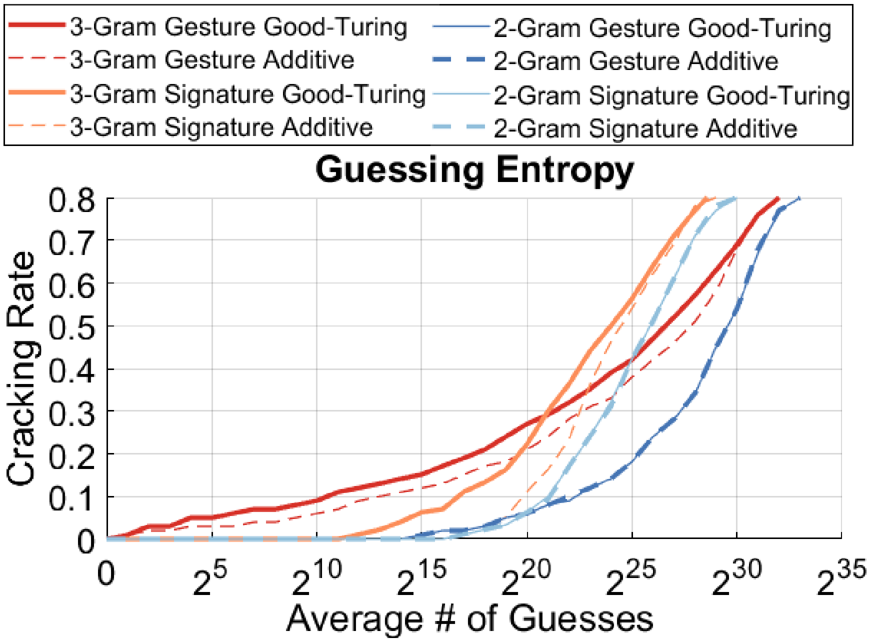


Figure 5.4: Guessing entropy of recognition passwords under six configurations of Markov chain: 2-gram Additive, 2-gram Good-Turing, 2-gram without smoothing, 3-gram Additive, 3-gram Good-Turing, and 3-gram without smoothing. Generally, 3-gram Markov chains for gestures and signatures are better than 2-gram models. There is no obvious difference between with and without smoothing methods.

Markov Chain Implementation

In order to estimate the guessing entropy with the Markov chain, we performed a 10-fold cross validation. We first combined the three gesture datasets (FreeForm (Sherman et al., 2014), Wild (Y. Yang et al., 2016), and GuessAttack (Liu et al., 2017a)) into one dataset. Then, we split it into ten subsets with roughly the same number of accounts. For each training process, we selected one subset as a testing set and used the other nine subsets as a training set.

Comparison of Markov Chains

For gesture and signature passwords, we tested both 2-gram and 3-gram Markov chains with and without the two smoothing methods, Good-Turing and additive, as Figure 5.4

shows. We needed to find Markov models to estimate both the upper- and lower-bound for the security of passwords.

The 3-gram Markov chain with Good-Turing smoothing achieves the highest cracking rates with the same average number of guesses for both gesture and signature passwords. We found the 3-gram Markov chains to be more efficient than the 2-gram models. Taking gestures as an example, the 3-gram Markov chain with the Good-Turing smoothing method is at least ten percentage points higher in efficiency than the 2-gram Markov chain with the Good-Turing smoothing method. However, the choice of smoothing method does not have a significant impact on guessing entropy. For example, we found that the 2-gram Markov chains with additive and Good-Turing smoothing nearly overlap for both gestures and signatures. The difference between Markov chains with and without smoothing is also not obvious with the exception of cases in which the target passwords have zero probabilities in the Markov chain. For example, we found that the guessing entropy of gestures based on a 3-gram model with and without Good-Turing smoothing are close to each other before 2^{26} guesses. Then, the guessing entropy of the Markov chain without smoothing is stable since the rest of the target passwords have zero probability. Therefore, we selected the 3-gram Markov model with Good-Turing smoothing and the 3-gram Markov model without smoothing to model the upper and lower bounds of the security of recognition passwords.

5.6 Partial Guessing Metric

Since guessing entropy, discussed above in Section 5.5.3, is based on the cracking rate of a specific password dataset, the security of the passwords will be over-or-under-estimated depending on the quality of that dataset.

Bonneau (Bonneau, 2012c) proposed a partial guessing metric (or $\alpha - guesswork$) for user-chosen passwords based on the password distribution to overcome the problems inherent in guessing entropy. The partial guessing metric models a practical attack situation in which the attacker has knowledge of the general password distribution $\chi = \{x_1, x_2, \cdots\}$ with the goal of cracking a certain percentage of the passwords.

We define $\mu_\alpha(\chi) = min(j|\Sigma_{i=1}^{j} p(x_i) \geq \alpha)$ as the minimal number of needed guesses to crack a α proportion of passwords, and define $\lambda_{\mu_\alpha(\chi)}(\chi) = \lambda_{\mu_\alpha}(\chi) = \Sigma_{i=1}^{\mu_\alpha} p(x_i)$ as the the actual cracked proportion of passwords with $\mu_\alpha(\chi)$ guesses. Then, the partial guessing metric is defined as

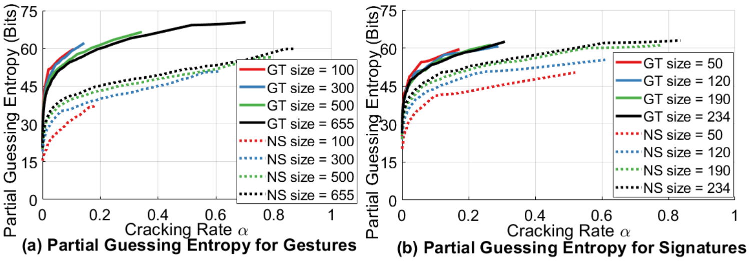
$$G_\alpha(\chi) = (1 - \lambda_{\mu_\alpha}) \cdot \mu_\alpha + \Sigma_{i=1}^{\mu_\alpha} p(x_i) \cdot i \tag{5.1}$$

where the first term reflects a fraction of passwords that are not cracked within a given number of guesses while the second term reflects the minimum expected number of guesses needed to crack the fraction α of possible passwords selected by people.

It is important to emphasize the key difference between guessing entropy and a partial guessing metric. Guessing entropy analyzes the cracking rate of a particular password set. In contrast, a partial guessing metric analyzes a fraction of the distributions of user-chosen passwords. The estimation would vary when the set is non-representative of the population or if there is skew introduced by participants. Imagine a set with ten passwords, nine of which are cracked within 100 guesses while the last one is cracked after 10^9 guesses. This leads to an extreme overestimation of guessing entropy higher than 10^8. The system is not secure since 90% of the passwords were cracked within 100 guesses[1].

---

[1] Partial guessing metric is *not* used to adjust the parameters of the Markov chain. Guessing entropy is used to adjust the model because we measure performance by successfully cracking as many passwords in



(a) Partial Guessing Entropy for Gestures (b) Partial Guessing Entropy for Signatures

Figure 5.5: Partial guessing metric of gestures and signatures that estimated based on different sizes of passwords datasets. We found the partial guessing metric curves with Good-Turing smoothing method decreases when the password datasets size increase. Similarly, we found the curves without smoothing methods increases when the password datasets size increase.

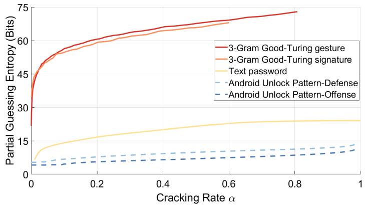


Figure 5.6: Partial guessing metric for different passwords. The baselines of security of gestures and signatures are considerably higher than Android unlock patterns.

5.7 Results

We first present the major result that was made possible by this present work: a quantitative evaluation of the security of recognition passwords by the partial guessing metric. Then, we compare the security between recognition passwords and Android unlock patterns.

5.7.1 Baseline of the Security of Recognition Passwords

Figure 5.5 shows that when the sizes of password datasets increase, the partial guessing metric estimation based on a Markov model with Good-Turing smoothing will decrease and the partial guessing metric based on a Markov model without smoothing will increase. This observation matches our analysis of the influence of the unseen passwords on the password distribution, as Figure 5.1 (b) shows.

Based on the observations of Figure 5.5, if we keep increasing the size of the datasets, the partial guessing metric curve of Good-Turing method will keep decreasing and the

---

the training set as possible without consideration to how much work is required to crack those passwords. A Markov chain cannot be trained with partial guessing metric since it requires probabilities to be assigned to an entire distribution, whereas we need the Markov chain to assign those probabilities.

curve without smoothing method will keep increasing. Eventually, they will converge to the partial guessing metric of the actual password distribution.

To summarize, with the current size of our recognition password dataset, we are able to provide the baseline of the partial guessing metric of gestures and signatures based on the Markov model without smoothing method.

5.7.2 Comparison to Android Unlock Patterns

Figure 5.6 shows that recognition passwords, gestures, and signatures have a higher partial guessing metric than Android unlock patterns. For gestures, the baseline of the partial guessing metric is 45 bits when the cracking rate $\alpha = 0.2$. This is 37 bits higher than the upper bound of defense-oriented Android unlock patterns with $\alpha = 0.2$. The password distribution of Android unlock patterns is modeled on the 3-gram Markov model with additive smoothing (Uellenbeck et al., 2013). Similarly, for signatures, the baseline of the partial guessing metric is 52 bits when the cracking rate $\alpha = 0.2$, which is also much higher than the corresponding metric for Android unlock patterns.

CHAPTER 6

DISCUSSION

6.1 Garda: Gesture Authentication for Mobile Systems

Our evaluation showed that our novel gesture authentication method – Garda – had the best authentication EER (0.015) and shortest authentication time among other sytems; Garda also resisted all of the brute-force attacks and most of the forgery attacks (EER = 0.040). Garda achieves this by combining the individual advantages of the Protractor and GMM-UBM recognition schemes. Protractor identifies gestures temporally, while GMM-UBM identifies the temporal-independent distributions of the coordinate points, so combining these two methods resulted in a more harmonious, effective authentication system.

sHMM performs the worst with an EER of 0.157. Following up gesture segmentation with a classification of the segmented parts removed crucial details from the genuine gestures. This had the side-effect of making them harder to distinguish. Of course, this is a function of the number of gesture segments: if we increase the segments, we can preserve more detail of the gesture. If we continue to increase segments, sHMM performs similarly to dHMM. This improves the EER from 0.157 to 0.056.

Six recognizers (EDR, LCS, DTW, dHMM, sHMM, and Garda) were capable of resisting all brute-force attacks, but for different reasons. EDR, LCS, and DTW use recognition methods based on the Euclidean distance between genuine and fake gestures. If the gesture points are randomly generated, as they are in the brute-force method, then the matching problem is almost impossible: a randomly generated collection of a hundred-or-so coordinate points need to fall close to a continuous sequence of gesture points clustered very closely together out of thousands of positions on the screen. Viewed this way, the successful results of these recognizers are not surprising . For dHMM and sHMM: the randomly generated gesture contains many sudden, random turns that can be used as segmentation points when compared to a user-generated gesture.

There are seven recognizers that were unable to resist brute-force attacks. Authentication thresholds in each scheme are trained to distinguish different users' gestures, and these thresholds are quite loose since they only need to recognize a single genuine gesture from entirely different gestures. There are three reasons for these three groups: Protractor-based methods (Protractor, Protractor-Kernel SVM, SVMGarda), SVM-based methods (Protractor, LCS, DTW, and EDR kernel SVM and SVMGarda), and cHMM. For Protractor-based methods: it depends on cosine distance, which only cares about the directions between two adjacent points. This is easier than matching gestures based on Euclidean distance. For SVM-based methods, it is the same reason as why the EER values are so high: the method transforms a large range of scores [0,inf] to a limited range of probabilities [0,1], increasing the opportunity for mis-classification. cHMM used the distribution of gesture points locations without considering the temporal order, making the brute-force attack more likely to crack cHMM since the attacks are based on randomly generated gestures.

For the imitation attacks, Garda achieved the lowest EER out of all the other methods. The reason that other methods cannot resist imitation attacks is that single time sequence authentication methods are unable to distinguish the variations in genuine gestures and the difference between the genuine and skilled attack gestures.

We can see that the Garda method is robust and practical for mobile authentication. First, Gardas authentication time is short (200 ms) and remains relatively stable as the number of gestures increases. Although we tested 20 gestures (20 different users) in the system to judge its viability, we note that it is unlikely a personal device would have so many different accounts. Even with 20 gestures, 30 seconds is affordable for a one-time training process. The mobile device only needs to store the trials, and GMM models; therefore, our system can have a short and stable authentication time and still be scalable for large numbers of gestures.

To best of our knowledge, we have presented the first analysis of how the invariances of gesture preprocessing impact the performance of gesture-based authentication systems. We found that an optimal combination (scale variant, location invariant, rotation variant) achieves the lowest EER (0.041) on average. This derives from the combination of individual effects of the three variables: rotation variant has a statistically significant, positive effect on EER; location variant has a statistically significant negative effect on EER; scale variant does not have a statistically significant effect on the EER.

The gesture variable combinations with EERs higher than $\overline{S}L\overline{R}$ can be explained by the individual effects of the three variables. First, since users are unable to draw their gestures at the same place every time, taking a gesture's location as variant always has a negative effect on authentication accuracy. Second, these three variables interact with each other. The binary state of one variable can influence other variables, and thus, influence the final recognition accuracy.

6.1.1 Limitations

We only evaluated the recognizers by authentication performance, brute-force attack, and imitation attack. Future investigations may evaluate the recognizers in regards to other aspects, like usability and other types of attacks. For example, the automated guessing attack can be used to examine the recognizers' performance under attacks that model real attackers. In addition to this open area of inquiry, we would like to address another limitation in that the public gesture datasets we used were not specifically created for authentication purposes. As the datasets' gestures may not follow the same distribution as gesture passwords, our results may not reflect user-selected passwords to the fullest extent.

6.2 Guessing Attacks on Gesture Passwords

Our dictionary attack method, when tested on a newly collected dataset from 109 participants, vastly outperformed the brute force attack. For unistroke gestures, this was a difference of 35.78 percentage points with Orientation Method I and a difference of 22.94 percentage points with Orientation Method II. Figure 4.7 shows that these types of gaps persist not matter whether unistroke, multistroke, or Orientation Method is considered: dictionary attacks are consistently better and always by double-digit percentages.

The takeaway from this is similarly clear, can be learned in research on cracking text passwords as well: dictionary attacks informed by using the most common passwords inside of a weak subspace generates more successful attacks than a random brute-force guesses. The disadvantage of the dictionary attack is also its strength: by targeting on the most common gestures it cannot crack gestures that are not covered by the weak subspace dictionary. Thus, the attacking system needs a large variety of free-form gestures to expand coverage of the weak subspace dictionary. In contrast, the brute force attack method does not need gesture samples in advance.

The size of the full space for gesture passwords is very large, despite the restrictions on the space imposed by our methodology. Table 4.3 shows that, given the propensity of users to select gesture passwords from the groupings we identified, the size of the full space is reduced down to 82.1 bits for Orientation Method I and 81.7 bits for Orientation Method II. Although there is a decline of over 20 bits from the full space size (109 bits) to the weak subspace size (82 bits), both the full space and weak subspace are of an impressive size. In spite of the degree of available gestures to choose from, even in an 80-bit large subspace, participants are still being routinely cracked at rates over 45%.

There is a sense that the more gestures that appear in Dictionary dataset, the larger the weak subspace size might be. As Table 4.2 shows, the distribution of weak set groups of Dictionary dataset and Test dataset are similar – independent of the size of each set. Currently, we computed an 82 bits weak subspace based on 407 weak gestures. Assuming we collected 100 times more gestures (40700 gestures), the 88.6 bit size would not change since we would need to assume a linear rise in the number of weak gestures that appear as well. The sizes of the weak subspaces are still much smaller than the full space size (about 109 bits), irrespective of the number of gestures. Therefore, we conclude that the larger size of Dictionary dataset may increase the size of weak subspace, but size does not have a significant influence on the relationship among the gesture groups in the weak set. The reason for this is that since the relative distribution of weak set groups are constant, the analysis of the weak subspace is also constant. Even with 100 times more gestures, the size of the weak subspace cannot enlarge 100 times because the weak subspace gestures are obtained by analyzing symbols with shared meanings (Shapes, Letters). Thus, the size of our Dictionary dataset is large enough to estimate the size of the weak subspace and output reliable results.

One possibility is that the sample size is skewing the results. However, this should work in the opposite direction: since we only have low numbers of gesture passwords, and given the space is so large, it should take many more examples of gesture passwords before (ideally) seeing a collision between two users independently sampling the password distribution. Instead, given the large space, we see frequent repetitions and overlap between independently and separately collected works of published gesture passwords, with our newly-collected data being cracked at a rate of 47.71%. Users are selecting gestures with meaning behind them likely because they are easier to remember and choose on-the-spot as a password. This selection bias has led to weak subspaces, however. One way to resist dictionary attacks for gesture passwords would be to advise participants to use combinations of symbols rather than singular symbols and signs. Unlike in text passwords, even simple combinations of symbols as a gesture password should make cracking far more difficult given the large increase in computational expense. The need to address user choice for creating gestures is clear. An alternative is to encourage, perhaps through policies, a methodology for generating gesture passwords that make them more random-appearing.

Each preprocessing step reduces the size of the password space. By design, that is their purpose – by reducing the total number of possibilities the intention is to make it easier for recognizers to distinguish similar gestures. This increases the likelihood of misinterpreting two gestures intended to be different as being the same. However, without preprocessing the space is much larger but the effect is that matching two gestures that are meant to be similar is far more difficult. It is easy to see why the size of the password space would be larger, though. A recognizer that does not allow location to change, for example, would likely determine two circles, drawn at different corners of the screen, to be different, despite the user's intentions. As a result, gestures drawn too far away are considered distinct. For example, a circle that would be worth only one gesture in our preprocessed space could be worth as much as four gestures (one for each corner of the screen) in an unprocessed space. The downside of location invariance is that a system's failure to accept the same gestures made in different locations will likely increase user frustrations. All of these points are in tension: the size of the password space, the recognizer efficiency, and usability of the system. Care has to be taken to balance all three.

6.2.1 Limitations

Our experiment is conducted in the laboratory, which may affect the distribution of user-chosen gesture passwords (Y. Yang et al., 2016). There has not been any evidence so far of a clear difference between the distributions of laboratory and field gesture passwords (Y. Yang et al., 2016). As such, we cannot estimate its influence on the cracking performance.

Multistroke gestures fared better than unistroke gestures at resisting cracking attempts, with the highest multistroke dictionary attack succeeding at 33.03% of passwords. The reason for this recognizer strength lies in the recognizer's construction of multistroke gestures. Multistroke gestures have disparate strokes connected together to form a single gesture, and these variances in drawing multiple strokes followed by connections make a gesture more difficult to account for or imitate.

We do not examine guessing attacks or brute force attacks against gesture passwords drawn with multiple fingers (multitouch). The datasets we obtained do not have a large number of samples for these multitouch passwords. This limits our ability to make generalizations about the weak subspace groups. Previous work (Sherman et al., 2014) showed biases in the distribution towards repeating unitouch figures with multiple fingers in the multitouch case. In this respect, an extension of research to analyze multitouch could be relatively simple, albeit with more expensive calculations. Multitouch gestures would be important further work to study.

6.3 Security Analysis of Gesture Passwords

Using the largest available password datasets, we estimated the baseline of recognition pass- word security by (1) representing the passwords by SAX and (2) assuming all of the unseen passwords to have zero probability of happening. For novel passwords, there are always concerns that the dataset is not large enough to reflect the full password distribution; however, it is still necessary to estimate the security of novel passwords before they are widely employed. We found that there are two reasons by which unseen passwords may occur: (1) the passwords are very unlikely to be selected by people, or (2) the passwords are likely to be selected by people, but are not covered by the password dataset. Unable to distinguish the two types of passwords, we dealt with all unseen passwords using the same approach. We therefore used two strategies: (1) assign zero probability to the unseen passwords, or (2) assign small probabilities to the unseen passwords based on Good-Turing smoothing method. As Figure 5.1 (b) illustrates and verified in Figure 5.5, the first strategy will narrow the password distribution, underestimate the security of the passwords, and provide the lower bound of the security of passwords. On the other hand, the second method will make the password distribution wider, overestimate the security, and provide the upper bound of the security. Through these analyses, we confirmed the baseline estimation of the security of recognition passwords comes from two aspects, one is the discretization of recognition passwords, one is the security estimation based on regarding the unseen passwords with zero probability. Therefore, we are able to estimate the security baseline of recognition passwords for a direct, numeric comparison to the security of matching passwords as well as any other recognition password.

We quantitatively compared the security of recognition passwords to that of Android unlock patterns. We have quantitatively shown that the security of gesture and signature passwords has a higher partial guessing metric than the Android pattern unlock method (Sherman et al., 2014; De Luca et al., 2014; Liu et al., 2017a). Prior work made arguments about security in three different ways: (1) quantifying the amount of expressive information contained in free-form gestures (Sherman et al., 2014), (2) calculating the size of the total password space (Sherman et al., 2014), and (3) segmenting the grid into patterns and calculating random entropy. However, these security measures are not as reliable as a partial guessing metric because they do not address how users or attackers behave (Bonneau, 2012c). The higher the partial guessing metric, the more secure the password system is. After assessing the number of guesses per account necessary for an attackerto crack an α-sized portion of all accounts on multiple different scales(Figure 5.4), we have shown with a rigorous approach and direct comparison that the lower bounds of gesture and signature passwords have a higher partial guessing metric than the upper bounds of Android unlock patterns.

We used distinguishability between recognition passwords to confirm the validity of discretizing the recognition passwords by SAX. A reasonable concern that arises in the discretization of recognition passwords asks whether the newly discretized passwords can represent original passwords. As an authentication method, the primary purpose of the recognition password feature is to distinguish various users' passwords. In other words, we had to keep both false positive errors and false negative errors low. Therefore, the distinguishability of passwords can be measured by ROC and AUROC. To show that SAX does not hurt the distinguishability of recognition passwords, we implemented state-of-the-art recognition methods for passwords and examined their ROC and AUROC results. We found that SAX achieves ROC and AUROC results comparable to other state-of-the-art recognition methods.

6.3.1 Limitations

Two of the gesture datasets (FreeForm (Sherman et al., 2014) and GuessAttack (Liu et al., 2017a)) and all of the signature datasets (Kholmatov & Yanikoglu, 2009; Ortega-Garcia et al., 2003; Yeung et al., 2004) in our study were collected in the laboratory, which in principle could affect the participants' selection of gestures and signatures as passwords (Y. Yang et al., 2016). By comparing the gestures collected from the laboratory (Sherman et al., 2014; Clark et al., 2017; Liu et al., 2017a) and in the wild (Y. Yang et al., 2016), we do not find any evidence of a clear difference between the passwords generated under these two environments. Thus, we cannot estimate the influence of experimental environments on our results.

The three gesture datasets were collected across different studies, and we aggregated them into one gesture dataset. Since all of the participants in the three datasets were asked to create gestures as passwords, different experimental setups will enlarge the diversity and coverage of the gesture passwords. Aggregating datasets from different studies does not damage our analysis of the password distribution.

Concerns remain about dataset size when it comes to estimating the partial guessing metric. The baseline estimation of partial guessing metric based on SAX and Markov model solved this question with fixed parameter values of SAX. However, a larger size for the dataset may also increase the parameter values of SAX since passwords that are clustered together become separated out more easily. This reduces the overall size of the weak set. Collecting additional passwords would not decrease the overall size of the alphabet. A larger number of passwords would require a larger alphabet to represent it if there are new attributes that need to be accounted for (see Figure 5.2 for a depiction of how the alphabet maps to a password). As such, if we collect many more passwords, we expect the size of the estimated password space to increase since the alphabet size will increase. This will increase both the baseline of the security estimation by the partial guessing metric.

The evidence of database relative frequencies analysis in Liu et al. (Liu et al., 2017a) shows that the distribution of our collected data are being sampled from a general distribution since we used exactly the same datasets as Liu et al. (Liu et al., 2017a). Basically, Liu et al. (Liu et al., 2017a) examined the relative frequencies of different gesture password categories across several gesture datasets in their cracking paper and found that the relative frequencies of groups (shapes, words, symbols) are equal despite being collected at different times by different groups (see Table 2 in Liu et al. (Liu et al., 2017a)).

CHAPTER 7

CONCLUSIONS

In this thesis, we conducted three stages of research to understand the use of gestures as an authentication method for mobile devices: (1) designed a robust gesture recognizer for authentication; (2) designed an automated guessing attack method against commonly used gesture recognizers; (3) evaluated and compared the security between different types of authentication methods.

We presented and evaluated a novel multi-expert gesture recognizer for authentication: Garda. We conducted rigorous evaluation of 13 different methods to implement gesture recognizers. We applied several datasets and two types of attacks against the recognizers. Finally, we conducted the first analysis of how tuning the variables of preprocessing methods of gesture recognizers can impact their authentication performance. All results show that Garda can largely improve the performance of gesture-based authentication systems. The presented authentication-optimal combination can reduce up to 45% of EER on average compared to the recognition-optimal configuration used in previous work.

Furthermore, we present the first work on performing automated guessing attacks for gesture passwords. We developed methodology for performing guessing attacks, showed how to enumerate the size of both the full and weak subspaces for gesture passwords, and identified and categorized a list of most commonly used gestures from published work on gesture passwords. We showed guessing attacks against gesture passwords by creating a generalizable method based on how gestures are recognized. We extended this attack method for a dictionary attack based on common gestures as well as contributed a brute force attack based on symmetric features to use as a benchmark measurement.

Lastly, we present the first work on estimating the security of recognition passwords through their distribution. We applied the analysis of passwords distribution with partial guessing entropyin our comparisons between the security of matching passwords and recognition passwords. We demonstrated a methodology for converting recognition passwords, based on time series data, into an equivalent alphabet set using Symbolic Aggregate ApproXimation (SAX). We validated the efficacy of the alphabet by showing that it could recreate the discretized passwords in such a way that they passed through multiple recognition methods (DTW, Garda, Protractor). We found, overall, that recognition passwords have better security than matching passwords.

BIBLIOGRAPHY

Anthony, L., Vatavu, R.-D., & Wobbrock, J. O. (2013). Understanding the consistency of users' pen and finger stroke gesture articulation. In Proceedings of graphics interface 2013 (pp. 87–94). Toronto, Ont., Canada, Canada: Canadian Information Processing Society. Retrieved from `http://dl.acm.org/citation.cfm?id=2532129.2532145`

Anthony, L., & Wobbrock, J. O. (2010). A lightweight multistroke recognizer for user interface prototypes. In Proceedings of graphics interface 2010 (pp. 245–252). Toronto, Ont., Canada, Canada: Canadian Information Processing Society. Retrieved from `http://dl.acm.org/citation.cfm?id=1839214.1839258`

Arthur P. Dempster, D. B. R., Nan M. Laird. (1977). Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society. Series B (Methodological), 39(1), 1-38. Retrieved from `http://www.jstor.org/stable/2984875`

Aslan, I., Uhl, A., Meschtscherjakov, A., & Tscheligi, M. (2014). Mid-air authentication gestures: An exploration of authentication based on palm and finger motions. In Proceedings of the 16th international conference on multimodal interaction (pp. 311–318). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2663204.2663246` doi: 10.1145/2663204.2663246

Aumi, M. T. I., & Kratz, S. (2014). Airauth: Evaluating in-air hand gestures for authentication. In Proceedings of the 16th international conference on human-computer interaction with mobile devices & services (pp. 309–318). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2628363.2628388` doi: 10.1145/2628363.2628388

Aviv, A. J., Budzitowski, D., & Kuber, R. (2015). Is bigger better? comparing user-generated passwords on 3x3 vs. 4x4 grid sizes for android's pattern unlock. In Proceedings of the 31st annual computer security applications conference (pp. 301–310). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2818000.2818014` doi: 10.1145/2818000.2818014

Aviv, A. J., Gibson, K., Mossop, E., Blaze, M., & Smith, J. M. (2010). Smudge attacks on smartphone touch screens. In Proceedings of the 4th usenix conference on offensive technologies (pp. 1–7). Berkeley, CA, USA: USENIX Association. Retrieved from `http://dl.acm.org/citation.cfm?id=1925004.1925009`

Bergroth., L., Hakonen., H., & Raita, T. (2000). A survey of longest common subsequence algorithms. In Proceedings of the seventh international symposium on string processing information retrieval (spire'00). Washington, DC, USA: IEEE Computer Society. Retrieved from `http://dx.doi.org/10.1109/SPIRE.2000.878178` doi: 10.1109/SPIRE.2000.878178

Biddle, R., Chiasson, S., & Van Oorschot, P. C. (2012, September). Graphical passwords: Learning from the first twelve years. ACM Computing Surveys, 44(4), 19:1–19:41. Retrieved from `http://dx.doi.org/10.1145/2333112.2333114` doi: 10.1145/2333112.2333114

Blocki, J., Datta, A., & Bonneau, J. (2016). Differentially private password frequency lists. In The network and distributed system security symposium 2016. Retrieved from `http://dx.doi.org/10.14722/ndss.2016.23328` doi: 10.14722/ndss.2016.23328

Bo, C., Zhang, L., Li, X.-Y., Huang, Q., & Wang, Y. (2013). Silentsense: Silent user identification via touch and movement behavioral biometrics. In Proceedings of the 19th annual international conference on mobile computing & networking (pp. 187–190). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2500423.2504572` doi: 10.1145/2500423.2504572

Bonneau, J. (2012a). Guessing human-chosen secrets (Doctoral dissertation, University of Cambridge). Retrieved from `https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-819.pdf`

Bonneau, J. (2012b). The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In Proceedings of the 2012 ieee symposium on security and privacy (pp. 538–552). Washington, DC, USA: IEEE Computer Society. Retrieved from `http://dx.doi.org/10.1109/SP.2012.49` doi: 10.1109/SP.2012.49

Bonneau, J. (2012c, May). The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In 2012 ieee symposium on security and privacy (p. 538-552). Retrieved from `http://dx.doi.org/10.1109/SP.2012.49` doi: 10.1109/SP.2012.49

Brostoff, S., & Sasse, M. A. (2000). Are passfaces more usable than passwords? a field trial investigation. In People and computers xiv — usability or else!: Proceedings of hci 2000 (pp. 405–424). London: Springer London. Retrieved from `http://dx.doi.org/10.1007/978-1-4471-0515-2_27` doi: 10.1007/978-1-4471-0515-2_27

Burgbacher, U., & Hinrichs, K. (2014). An implicit author verification system for text messages based on gesture typing biometrics. In Proceedings of the sigchi conference on human factors in computing systems (pp. 2951–2954). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2556288.2557346` doi: 10.1145/2556288.2557346

Cachin, C. (1997). Entropy measures and unconditional security in cryptography.

Campbell, W. M., Campbell, J. P., Reynolds, D. A., Singer, E., & Torres-Carrasquillo, P. A. (2006). Support vector machines for speaker and language recognition. Computer Speech & Language, 20(2), 210–229. Retrieved from `http://dx.doi.org/10.1016/j.csl.2005.06.003` doi: 10.1016/j.csl.2005.06.003

Campbell, W. M., Sturim, D. E., & Reynolds, D. A. (2006). Support vector machines using gmm supervectors for speaker verification. Signal Processing Letters, IEEE, 13(5), 308–311. Retrieved from `http://dx.doi.org/10.1109/LSP.2006.870086` doi: 10.1109/LSP.2006.870086

Chen, L., Özsu, M. T., & Oria, V. (2005). Robust and fast similarity search for moving object trajectories. In Proceedings of the 2005 acm sigmod international conference on management of data (pp. 491–502). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/1066157.1066213` doi: 10.1145/1066157.1066213

Chiasson, S., Forget, A., Biddle, R., & van Oorschot, P. C. (2009, October). User interface design affects security: Patterns in click-based graphical passwords. International Journal of Information Security, 8(6), 387–398. Retrieved from `http://dx.doi.org/10.1007/s10207-009-0080-7` doi: 10.1007/s10207-009-0080-7

Clark, G. D., & Lindqvist, J. (2015, Jan). Engineering gesture-based authentication systems. Pervasive Computing, IEEE, 14(1), 18-25. Retrieved from `http://dx.doi.org/10.1109/MPRV.2015.6` doi: 10.1109/MPRV.2015.6

Clark, G. D., Lindqvist, J., & Oulasvirta, A. (2017). Composition policies for gesture passwords: User choice, security, usability and memorability. In 2017 ieee conference on communications and network security (cns). IEEE.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297. Retrieved from `http://dx.doi.org/10.1007/BF00994018` doi: 10.1007/BF00994018

Czyz, J., Sadeghi, M., Kittler, J., & Vandendorpe, L. (2004). Decision fusion for face authentication. In Biometric authentication (pp. 686–693). Springer. Retrieved from `http://dx.doi.org/10.1007/978-3-540-25948-0_93` doi: 10.1007/978-3-540-25948-0_93

Davis, D., Monrose, F., & Reiter, M. K. (2004). On user choice in graphical password schemes. In Proceedings of the 13th conference on usenix security symposium (usenix security '04). Berkeley, CA, USA: USENIX Association. Retrieved from `https://www.usenix.org/legacy/events/sec04/tech/full_papers/davis/davis_html/usenix04.html`

De Luca, A., Hang, A., Brudy, F., Lindner, C., & Hussmann, H. (2012). Touch me once and i know it's you!: Implicit authentication based on touch screen patterns. In Proceedings of the sigchi conference on human factors in computing systems (pp. 987–996). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2207676.2208544` doi: 10.1145/2207676.2208544

De Luca, A., Harbach, M., von Zezschwitz, E., Maurer, M.-E., Slawik, B. E., Hussmann, H., & Smith, M. (2014). Now you see me, now you don't: Protecting smartphone authentication from shoulder surfers. In Proceedings of the sigchi conference on human factors in computing systems (pp. 2937–2946). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2556288.2557097` doi: 10.1145/2556288.2557097

De Luca, A., von Zezschwitz, E., Nguyen, N. D. H., Maurer, M.-E., Rubegni, E., Scipioni, M. P., & Langheinrich, M. (2013). Back-of-device authentication on smartphones. In Proceedings of the sigchi conference on human factors in computing systems (pp. 2389–2398). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2470654.2481330` doi: 10.1145/2470654.2481330

Deng, J.-W., & tat Tsui, H. (2000). An hmm-based approach for gesture segmentation and recognition. In Pattern recognition, 2000. proceedings. 15th international conference on (Vol. 3, p. 679-682 vol.3). doi: 10.1109/ICPR.2000.903636

Dimauro, G., Impedovo, S., Pirlo, G., & Salzo, A. (1997). A multi-expert signature verification system for bankcheck processing. International Journal of Pattern Recognition and Artificial Intelligence, 11(05), 827–844. Retrieved from `http://dx.doi.org/10.1142/S0218001497000378` doi: 10.1142/S0218001497000378

Elmezain, M., Al-Hamadi, A., & Michaelis, B. (2009, Nov). Hand trajectory-based gesture spotting and recognition using hmm. In 2009 16th ieee international conference on image processing (icip) (p. 3577-3580). Retrieved from `http://dx.doi.org/10.1109/ICIP.2009.5414322` doi: 10.1109/ICIP.2009.5414322

Fawcett, T. (2006, June). An introduction to roc analysis. Pattern Recogn. Lett., 27(8), 861–874. Retrieved from `http://dx.doi.org/10.1016/j.patrec.2005.10.010` doi: 10.1016/j.patrec.2005.10.010

Florencio, D., & Herley, C. (2007). A large-scale study of web password habits. In Proceedings of the 16th international conference on world wide web (pp. 657–666). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/1242572.1242661` doi: 10.1145/1242572.1242661

Frank, M., Biedert, R., Ma, E., Martinovic, I., & Song, D. (2013, January). Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. Trans. Info. For. Sec., 8(1), 136–148. Retrieved from `http://dx.doi.org/10.1109/TIFS.2012.2225048` doi: 10.1109/TIFS.2012.2225048

Gale, W. A., & Sampson, G. (1995). Good-turing frequency estimation without tears. Journal of Quantitative Linguistics, 2(3), 217–237. Retrieved from `http://dx.doi.org/10.1080/09296179508590051` doi: 10.1080/09296179508590051

Golla, M., Detering, D., & Dürmuth, M. (2017). Emojiauth: quantifying the security of emoji-based authentication. In The network and distributed system security symposium 2016. Retrieved from `http://dx.doi.org/10.14722/usec.2017.23024`

Hayashi, E., Maas, M., & Hong, J. I. (2014). Wave to me: User identification using body lengths and natural gestures. In Proceedings of the sigchi conference on human factors in computing systems (pp. 3453–3462). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2556288.2557043` doi: 10.1145/2556288.2557043

Heckbert, P. S., & Garland, M. (1997). Survey of polygonal surface simplification algorithms (Tech. Rep.). DTIC Document. Retrieved from `http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA461098`

Hse, H., & Newton, A. R. (2004). Sketched symbol recognition using zernike moments. In Proceedings of the pattern recognition, 17th international conference on (icpr'04) volume 1 - volume 01 (pp. 367–370). Washington, DC, USA: IEEE Computer Society. Retrieved from `http://dx.doi.org/10.1109/ICPR.2004.838` doi: 10.1109/ICPR.2004.838

INSIDEPRO. (n.d.). Passwordspro. Retrieved from `http://www.insidepro.com/`

Jain, A. K., Ross, A., & Pankanti, S. (2006, November). Biometrics: A tool for information security. Trans. Info. For. Sec., 1(2), 125–143. Retrieved from `http://dx.doi.org/10.1109/TIFS.2006.873653` doi: 10.1109/TIFS.2006.873653

Jelinek, F., Bahl, L. R., & Mercer, R. L. (2006, September). Design of a linguistic statistical decoder for the recognition of continuous speech. IEEE Trans. Inf. Theor., 21(3), 250–256. Retrieved from `http://dx.doi.org/10.1109/TIT.1975.1055384` doi: 10.1109/TIT.1975.1055384

Jermyn, I., Mayer, A., Monrose, F., Reiter, M. K., & Rubin, A. D. (1999). The design and analysis of graphical passwords. In Proceedings of the 8th conference on usenix security symposium - volume 8 (pp. 1–1). Berkeley, CA, USA: USENIX Association. Retrieved from `http://dl.acm.org/citation.cfm?id=1251421.1251422`

Kelley, P. G., Komanduri, S., Mazurek, M. L., Shay, R., Vidas, T., Bauer, L., ... Lopez, J. (2012). Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In Proceedings of the 2012 ieee symposium on security and privacy (pp. 523–537). Washington, DC, USA: IEEE Computer Society. Retrieved from `http://dx.doi.org/10.1109/SP.2012.38` doi: 10.1109/SP.2012.38

Keogh, E., & Ratanamahatana, C. A. (2005, March). Exact indexing of dynamic time warping. Knowledge and information systems, 7(3), 358–386. Retrieved from `http://dx.doi.org/10.1007/s10115-004-0154-9` doi: 10.1007/s10115-004-0154-9

Keogh, E. J., & Pazzani, M. J. (2000). Scaling up dynamic time warping for datamining applications. In Proceedings of the sixth acm sigkdd international conference on knowledge discovery and data mining (pp. 285–289). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/347090.347153` doi: 10.1145/347090.347153

Kholmatov, A., & Yanikoglu, B. (2005, November). Identity authentication using improved online signature verification method. Pattern Recogn. Lett., 26(15), 2400–2408. Retrieved from `http://dx.doi.org/10.1016/j.patrec.2005.04.017` doi: 10.1016/j.patrec.2005.04.017

Kholmatov, A., & Yanikoglu, B. (2009, September). Susig: An on-line signature database, associated protocols and benchmark results. Pattern Anal. Appl., 12(3), 227–236. Retrieved from `http://dx.doi.org/10.1007/s10044-008-0118-x` doi: 10.1007/s10044-008-0118-x

Kiesel, J., Stein, B., & Lucks, S. (2017). A large-scale analysis of the mnemonic password advice.. Retrieved from `http://dx.doi.org/10.14722/ndss.2017.23077` doi: 10.14722/ndss.2017.23077

Lee, H.-K., & Kim, J. H. (1999, Oct). An hmm-based threshold model approach for gesture recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(10), 961-973. Retrieved from `http://dx.doi.org/10.1109/34.799904` doi: 10.1109/34.799904

Li, Y. (2010). Protractor: A fast and accurate gesture recognizer. In Proceedings of the sigchi conference on human factors in computing systems (pp. 2169–2172). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/1753326.1753654` doi: 10.1145/1753326.1753654

Lin, J., Keogh, E., Lonardi, S., & Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In Proceedings of the 8th acm sigmod workshop on research issues in data mining and knowledge discovery (pp. 2–11). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/882082.882086` doi: 10.1145/882082.882086

Lin, J., Keogh, E., Wei, L., & Lonardi, S. (2007). Experiencing sax: a novel symbolic representation of time series. Data Mining and Knowledge Discovery, 15(2), 107–144. Retrieved from `http://dx.doi.org/10.1007/s10618-007-0064-z` doi: 10.1007/s10618-007-0064-z

Liu, C., Clark, G. D., & Lindqvist, J. (2017a, March). Guessing attacks on user-generated gesture passwords. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 1(3). Retrieved from `http://dx.doi.org/10.1145/3053331` doi: 10.1145/3053331

Liu, C., Clark, G. D., & Lindqvist, J. (2017b). Where usability and security go hand-in-hand: Robust gesture-based authentication for mobile systems. In Proceedings of the sigchi conference on human factors in computing systems. New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/3025453.3025879` doi: 10.1145/3025453.3025879

Lloyd, S. P. (2006, September). Least squares quantization in pcm. Information Theory, IEEE Transactions on, 28(2), 129–137. Retrieved from `http://dx.doi.org/10.1109/TIT.1982.1056489` doi: 10.1109/TIT.1982.1056489

Luca, A. D., & Lindqvist, J. (2015, May). Is secure and usable smartphone authentication asking too much? Computer, 48(5), 64-68. doi: 10.1109/MC.2015.134

Ma, J., Yang, W., Luo, M., & Li, N. (2014). A study of probabilistic password models. In Proceedings of the 2014 ieee symposium on security and privacy (pp. 689–704). Washington, DC, USA: IEEE Computer Society. Retrieved from `http://dx.doi.org/10.1109/SP.2014.50` doi: 10.1109/SP.2014.50

MacQueen, J., et al. (1967). Some methods for classification and analysis of multivariate observations. In Proceedings of the 5th berkeley symposium on mathematical statistics and probability (Vol. 1, pp. 281–297). University of California Press, Berkeley. Retrieved from `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.308.8619`

Massey, J. L. (1994, Jun). Guessing and entropy. In Proceedings of 1994 ieee international symposium on information theory (p. 204-). doi: 10.1109/ISIT.1994.394764

Melicher, W., Ur, B., Segreti, S. M., Komanduri, S., Bauer, L., Christin, N., & Cranor, L. F. (2016a). Fast, lean, and accurate: Modeling password guessability using neural networks. In Proceedings of the 25th conference on usenix security symposium (usenix security '16) (pp. 175–191). Austin, TX: USENIX Association. Retrieved from `https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/melicher`

Melicher, W., Ur, B., Segreti, S. M., Komanduri, S., Bauer, L., Christin, N., & Cranor, L. F. (2016b). Fast, lean, and accurate: Modeling password guessability using neural networks. In 25th USENIX security symposium (USENIX security 16) (pp. 175–191). Austin, TX: USENIX Association. Retrieved from `https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/melicher`

Morris, R., & Thompson, K. (1979, November). Password security: A case history. Commun. ACM, 22(11), 594–597. Retrieved from `http://doi.acm.org/10.1145/359168.359172` doi: 10.1145/359168.359172

Morse, M. D., & Patel, J. M. (2007). An efficient and accurate method for evaluating time series similarity. In Proceedings of the 2007 acm sigmod international conference on management of data (pp. 569–580). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/1247480.1247544` doi: 10.1145/1247480.1247544

Myers, C. S., & Rabiner, L. R. (1981, Sept). A comparative study of several dynamic time-warping algorithms for connected-word recognition. The Bell System Technical Journal, 60(7), 1389-1409. Retrieved from `http://dx.doi.org/10.1002/j.1538-7305.1981.tb00272.x` doi: 10.1002/j.1538-7305.1981.tb00272.x

Nacenta, M. A., Kamber, Y., Qiang, Y., & Kristensson, P. O. (2013). Memorability of pre-designed and user-defined gesture sets. In Proceedings of the sigchi conference on human factors in computing systems (pp. 1099–1108). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2470654.2466142` doi: 10.1145/2470654.2466142

Narayanan, A., & Shmatikov, V. (2005). Fast dictionary attacks on passwords using time-space tradeoff. In Proceedings of the 12th acm conference on computer and communications security (pp. 364–372). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/1102120.1102168` doi: 10.1145/1102120.1102168

Oh, U., & Findlater, L. (2013). The challenges and potential of end-user gesture customization. In Proceedings of the sigchi conference on human factors in computing systems (pp. 1129–1138). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2470654.2466145` doi: 10.1145/2470654.2466145

Ortega-Garcia, J., Fierrez-Aguilar, J., Simon, D., Gonzalez, J., Faundez-Zanuy, M., Espinosa, V., … others (2003). Mcyt baseline corpus: A bimodal biometric database. IEE Proceedings-Vision, Image and Signal Processing, 150(6), 395–401. Retrieved from `http://dx.doi.org/10.1049/ip-vis:20031078` doi: 10.1049/ip-vis:20031078

Percival, C. (2009, June). Stronger key derivation via sequential memory-hard functions. In Proceedings of bsdcan ’09. Ottawa, ON, Canada.

Peslyak, A. (2017). John the ripper. Retrieved from `http://www.openwall.com/john/`

Ratanamahatana, C. A., & Keogh, E. (2004). Everything you know about dynamic time warping is wrong. In Third workshop on mining temporal and sequential data (pp. 53–63). Retrieved from `http://wearables.cc.gatech.edu/paper_of_week/DTW_myths.pdf`

Reynolds, D. A., Quatieri, T. F., & Dunn, R. B. (2000, January). Speaker verification using adapted gaussian mixture models. Digit. Signal Process., 10(1), 19–41. Retrieved from `http://dx.doi.org/10.1006/dspr.1999.0361` doi: 10.1006/dspr.1999.0361

Rousseeuw, P. (1987, November). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20(1), 53–65. Retrieved from `http://dx.doi.org/10.1016/0377-0427(87)90125-7` doi: 10.1016/0377-0427(87)90125-7

Rutkowska, D. (2004). Multi-expert systems. In Parallel processing and applied mathematics: 5th international conference, ppam 2003, czestochowa, poland, september 7-10, 2003. revised papers (pp. 650–658). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from `http://dx.doi.org/10.1007/978-3-540-24669-5_85` doi: 10.1007/978-3-540-24669-5_85

Sae-Bae, N., Ahmed, K., Isbister, K., & Memon, N. (2012). Biometric-rich gestures: A novel approach to authentication on multi-touch devices. In Proceedings of the sigchi conference on human factors in computing systems (pp. 977–986). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2207676.2208543` doi: 10.1145/2207676.2208543

Sahami Shirazi, A., Moghadam, P., Ketabdar, H., & Schmidt, A. (2012). Assessing the vulnerability of magnetic gestural authentication to video-based shoulder surfing attacks. In Proceedings of the sigchi conference on human factors in computing systems (pp. 2045–2048). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2207676.2208352` doi: 10.1145/2207676.2208352

Sakoe, H., & Chiba, S. (1978, Feb). Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing, 26(1), 43-49. Retrieved from `http://dx.doi.org/10.1109/TASSP.1978.1163055` doi: 10.1109/TASSP.1978.1163055

Schaub, F., Walch, M., Könings, B., & Weber, M. (2013). Exploring the design space of graphical passwords on smartphones. In Proceedings of the ninth symposium on usable privacy and security (pp. 11:1–11:14). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2501604.2501615` doi: 10.1145/2501604.2501615

Serwadda, A., & Phoha, V. V. (2013). When kids' toys breach mobile phone security. In Proceedings of the 2013 acm sigsac conference on computer & communications security (pp. 599–610). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2508859.2516659` doi: 10.1145/2508859.2516659

Shahzad, M., Liu, A. X., & Samuel, A. (2013). Secure unlocking of mobile touch screen devices by simple gestures: You can see it but you can not do it. In Proceedings of the 19th annual international conference on mobile computing & networking (pp. 39–50). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2500423.2500434` doi: 10.1145/2500423.2500434

Sherman, M., Clark, G., Yang, Y., Sugrim, S., Modig, A., Lindqvist, J., … Roos, T. (2014). User-generated free-form gestures for authentication: Security and memorability. In Proceedings of the 12th annual international conference on mobile systems, applications, and services (pp. 176–189). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2594368.2594375` doi: 10.1145/2594368.2594375

Song, Y., Cho, G., Oh, S., Kim, H., & Huh, J. H. (2015). On the effectiveness of pattern lock strength meters: Measuring the strength of real world pattern locks. In Proceedings of the 33rd annual acm conference on human factors in computing systems (pp. 2343–2352). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2702123.2702365` doi: 10.1145/2702123.2702365

Steube, J. (2017). Hashcat. Retrieved from `https://hashcat.net/oclhashcat/`

Sugrim, S., Liu, C., & Lindqvist, J. (2019, September). Recruit until it fails: Exploring performance limits for identification systems. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol., 3(3), 104:1–104:26. Retrieved from `http://doi.acm.org/10.1145/3351262` doi: 10.1145/3351262

Sugrim, S., Liu, C., McLean, M., & Lindqvist, J. (2019). Robust performance metrics for authentication systems. In The network and distributed system security symposium 2019. Retrieved from `https://dx.doi.org/10.14722/ndss.2019.23351`

Tao, H., & Adams, C. (2008). Pass-go: A proposal to improve the usability of graphical passwords. International Journal of Network Security, 7(2), 273–292.

Thorpe, J., & van Oorschot, P. C. (2004). Graphical dictionaries and the memorable space of graphical passwords. In Proceedings of the 13th conference on usenix security symposium - volume 13 (pp. 10–10). Berkeley, CA, USA: USENIX Association. Retrieved from `http://dl.acm.org/citation.cfm?id=1251375.1251385`

Tian, J., Qu, C., Xu, W., & Wang, S. (2013). Kinwrite: Handwriting-based authentication using kinect. In 20th annual network & distributed system security symposium. Retrieved from `http://www.internetsociety.org/sites/default/files/10_2_0.pdf`

Uellenbeck, S., Dürmuth, M., Wolf, C., & Holz, T. (2013). Quantifying the security of graphical passwords: The case of android unlock patterns. In Proceedings of the 2013 acm sigsac conference on computer & communications security (pp. 161–172). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2508859.2516700` doi: 10.1145/2508859.2516700

Uludag, U., Pankanti, S., Prabhakar, S., & Jain, A. K. (2004, June). Biometric cryptosystems: issues and challenges. Proceedings of the IEEE, 92(6), 948-960. doi: 10.1109/JPROC.2004.827372

Ur, B., Segreti, S. M., Bauer, L., Christin, N., Cranor, L. F., Komanduri, S., … Shay, R. (2015). Measuring real-world accuracies and biases in modeling password guessability. In Proceedings of the 24th conference on usenix security symposium (usenix security '15) (pp. 463–481). Washington, D.C.: USENIX Association. Retrieved from `https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/ur` van Oorschot, P. C., & Thorpe, J. (2011, December). Exploiting predictability in click-based graphical passwords. Journal of Computer Security, 19(4), 669–702. Retrieved from `http://dx.doi.org/10.3233/JCS-2010-0411`

Vatavu, R.-D., Anthony, L., & Wobbrock, J. O. (2012). Gestures as point clouds: A $p recognizer for user interface prototypes. In Proceedings of the 14th acm international conference on multimodal interaction (pp. 273–280). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2388676.2388732` doi: 10.1145/2388676.2388732

Vatavu, R.-D., Vogel, D., Casiez, G., & Grisoni, L. (2011). Estimating the perceived difficulty of pen gestures. In Proceedings of the 13th ifip tc 13 international conference on human-computer interaction - volume part ii (pp. 89–106). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from `http://dx.doi.org/10.1007/978-3-642-23771-3_9` doi: 10.1007/978-3-642-23771-3_9

Weir, M., Aggarwal, S., Collins, M., & Stern, H. (2010). Testing metrics for password creation policies by attacking large sets of revealed passwords. In Proceedings of the 17th acm conference on computer and communications security (pp. 162–175). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/1866307.1866327` doi: 10.1145/1866307.1866327

Weir, M., Aggarwal, S., Medeiros, B. d., & Glodek, B. (2009a). Password cracking using probabilistic context-free grammars. In Proceedings of the 2009 30th ieee symposium on security and privacy (pp. 391–405). Washington, DC, USA: IEEE Computer Society. Retrieved from `http://dx.doi.org/10.1109/SP.2009.8` doi: 10.1109/SP.2009.8

Weir, M., Aggarwal, S., Medeiros, B. d., & Glodek, B. (2009b). Password cracking using probabilistic context-free grammars. In Proceedings of the 2009 30th ieee symposium on security and privacy (pp. 391–405). Washington, DC, USA: IEEE Computer Society. Retrieved from `http://dx.doi.org/10.1109/SP.2009.8` doi: 10.1109/SP.2009.8

Wheeler, D. L. (2016). zxcvbn: Low-budget password strength estimation. In Proceedings of the 25th conference on usenix security symposium (usenix security '16) (pp. 157–173). Austin, TX: USENIX Association. Retrieved from `https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/wheeler`

Wiedenbeck, S., Waters, J., Birget, J.-C., Brodskiy, A., & Memon, N. (2005, July). Passpoints: Design and longitudinal evaluation of a graphical password system. International Journal of Human-Computer Studies, 63(1-2), 102–127. Retrieved from `http://dx.doi.org/10.1016/j.ijhcs.2005.04.010` doi: 10.1016/j.ijhcs.2005.04.010

Wobbrock, J. O., Wilson, A. D., & Li, Y. (2007). Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In Proceedings of the 20th annual acm symposium on user interface software and technology (pp. 159–168). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/1294211.1294238` doi: 10.1145/1294211.1294238

Wu, J., Konrad, J., & Ishwar, P. (2013, May). Dynamic time warping for gesture-based user identification and authentication with kinect. In Acoustics, speech and signal processing, 2013 ieee international conference on (p. 2371-2375). Retrieved from `http://dx.doi.org/10.1109/ICASSP.2013.6638079` doi: 10.1109/ICASSP.2013.6638079

Yang, J., Li, Y., & Xie, M. (2015, March). Motionauth: Motion-based authentication for wrist worn smart devices. In Pervasive computing and communication workshops, 2015 ieee international conference on (p. 550-555). Retrieved from `http://dx.doi.org/10.1109/PERCOMW.2015.7134097` doi: 10.1109/PERCOMW.2015.7134097

Yang, Y., Clark, G. D., Lindqvist, J., & Oulasvirta, A. (2016). Free-form gesture authentication in the wild. In Proceedings of the sigchi conference on human factors in computing systems (pp. 3722–3735). New York, NY, USA: ACM. Retrieved from `http://dx.doi.org/10.1145/2858036.2858270` doi: 10.1145/2858036.2858270

Yeung, D.-Y., Chang, H., Xiong, Y., George, S., Kashi, R., Matsumoto, T., & Rigoll, G. (2004). Svc2004: First international signature verification competition. In D. Zhang & A. K. Jain (Eds.), Biometric authentication: First international conference, icba 2004, hong kong, china, july 15-17, 2004. proceedings (pp. 16–22). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from `http://dx.doi.org/10.1007/978-3-540-25948-0_3` doi: 10.1007/978-3-540-25948-0_3

Zheng, N., Bai, K., Huang, H., & Wang, H. (2014). You are how you touch: User verification on smartphones via tapping behaviors. In 2014 ieee 22nd international conference on network protocols (pp. 221–232). IEEE. Retrieved from `http://dx.doi.org/10.1109/ICNP.2014.43` doi: 10.1109/ICNP.2014.43