Winter 2014

Vulnerability analysis of cyber-behavioral biometric authentication

Abdul Serwadda
Louisiana Tech University

VULNERABILITY ANALYSIS OF CYBER-BEHAVIORAL BIOMETRIC AUTHENTICATION

by

Abdul Serwadda, B.Sc., PGDDCSE., M.S., M.S.

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

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

March 2014

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

July 30th 2013

Date

We hereby recommend that the dissertation prepared under our supervision by Abdul Serwadda

entitled

Vulnerability Analysis of Cyber-Behavioral Biometric Authentication

be accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

[illegible]

Supervisor of Dissertation Research

[illegible]

Head of Department

Computational Analysis and Modeling

Department

Recommendation concurred in:

[illegible]

[illegible]

[illegible]

[illegible]

Advisory Committee

Approved:

[illegible]

Director of Graduate Studies

[illegible]

Dean of the College

Approved:

[illegible]

Dean of the Graduate School

GS Form 13a
(6/07)

ABSTRACT

Research on cyber-behavioral biometric authentication has traditionally assumed naïve (or zero-effort) impostors who make no attempt to generate sophisticated forgeries of biometric samples. Given the plethora of adversarial technologies on the Internet, it is questionable as to whether the zero-effort threat model provides a realistic estimate of how these authentication systems would perform in the wake of adversity. To better evaluate the efficiency of these authentication systems, there is need for research on algorithmic attacks which simulate the state-of-the-art threats.

To tackle this problem, we took the case of keystroke and touch-based authentication and developed a new family of algorithmic attacks which leverage the intrinsic instability and variability exhibited by users' behavioral biometric patterns. For both fixed-text (or password-based) keystroke and continuous touch-based authentication, we: 1) Used a wide range of pattern analysis and statistical techniques to examine large repositories of biometrics data for weaknesses that could be exploited by adversaries to break these systems, 2) Designed algorithmic attacks whose mechanisms hinge around the discovered weaknesses, and 3) Rigorously analyzed the impact of the attacks on the best verification algorithms in the respective research domains.

When launched against three high performance password-based keystroke verification systems, our attacks increased the mean Equal Error Rates (EERs) of the systems by between 28.6% and 84.4% relative to the traditional zero-effort attack.

For the touch-based authentication system, the attacks performed even better, as they increased the system's mean EER by between 338.8% and 1535.6% depending on parameters such as the failure-to-enroll threshold and the type of touch gesture subjected to attack. For both keystroke and touch-based authentication, we found that there was a small proportion of users who saw considerably greater performance degradation than others as a result of the attack. There was also a sub-set of users who were completely immune to the attacks.

Our work exposes a previously unexplored weakness of keystroke and touch-based authentication and opens the door to the design of behavioral biometric systems which are resistant to statistical attacks.

DEDICATION

I dedicate this Dissertation to my mother, Ms. Betty Nagadya. Without doubt, I would never have gotten this far without her support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to several colleagues, friends and relatives, without whose support I would never have completed this work. First, I would like to thank my PhD advisor, Prof. Vir V. Phoha for his invaluable support and guidance during the entire four-year period for which I have pursued my Ph.D. Prof. Phoha introduced me to machine learning and its applications in cyber security and patiently educated me on this subject during the entire course of my Ph.D. I am also very grateful to Prof. Idris A. Rai, the person who first begun to polish me up as a researcher, after having noticed my raw research potential. Without Prof. Rai's insistence, I probably would never have made the decision to join the Ph.D. program.

I would like to thank my advisory committee members—Dr. Weizhong Dai, Dr. Jinko Kanno, Dr. Ratsko Selmic and Dr. Dexter Cahoy—for their input and support towards the completion of my dissertation. I am also very grateful to the funding sources which made my research possible. My research was supported by a Louisiana Board of Regents grant for the first three years, and a Defense Advanced Research Projects Agency (DARPA) Active Authentication grant during the final year.

Several graduate students in our research lab contributed towards my research in different ways; Zibo Wang co-authored several papers with me, was always a discussion partner on matters regarding research strategy, and participated in running some of my experiments for the last chapter. David Irakiza proof read every single paper that I wrote, and, along with Krystal Corbett, sacrificed a great deal of time exploring potential finger material for my robot. Ankunda Kiremire co-authored a paper with me and often gave insightful feedback on my work. Rajesh Kumar and Diksha Shukla joined our research lab at a time when I was translating my keystroke population statistics-based attack into a robotic population statistics-based attack and spent a big portion of their first two months in grad school helping me during data collection and the design of the robot.

Outside the school setup, several friends helped me maintain the momentum necessary to complete this work: Kawuki Tonny, Semuwemba James, Matovu Raymond, Mulumba John and Ssekalo Mahmud are just a few of the friends on this list. Finally, I would like to thank my mother Betty Nagadya, my grandmother Lubuzibwa Esereda, my father Muhammad Serwadda and my siblings and other relatives who include: Joanitor Nalwoga, Eric Lutwama, Stella Nabbosa, Tabitha Nalugwa, Rachel Kayaga, Abdul Kalema, Sherifa Nakalema, Amina Nakibuule, Samuel Sendi and Fred Serugunda for their endless encouragement during this process.

CHAPTER 1

INTRODUCTION

1.1 Overview

Biometrics — a set of measurements of either the human characteristics acquired naturally over time (behavioral biometrics), or the inherent physical traits of an individual (physiological biometrics) — have recently seen a lot of applications in user authentication [1, 2] and cryptographic key generation [3, 4]. The increased interest in biometrics has stemmed from a number of factors, key among which being the generally high entropy across a user population, and the elimination of the need for a subject to memorize a potentially complex secret.

While physiological biometrics (*e.g.*, fingerprints, iris patterns) are stable and highly unique for each user, behavioral biometrics (*e.g.*, keystroke dynamics, handwriting, touch gestures) tend to be imprecise, in some cases exhibiting considerable intra-user variability and overlap across users. This imprecision and variability prompts questions as to whether well orchestrated statistical attacks would not severely degrade the performance of authentication systems based on these modalities. Unfortunately, the majority of research in this field seems to disregard this threat, with most systems being evaluated under the assumption of a naïve attacker who is unable to pull off a sophisticated forgery. As a result, very little is known about the resistance of these systems to sophisticated forgeries.

In this work, we applied a wide range of pattern analysis techniques to discover attack vulnerabilities in keystroke and touch biometrics data, and then developed a new family of algorithmic attacks that exploit the discovered weaknesses to degrade the performance of the two categories of authentication systems. Keystroke authentication — the use of keyboard typing traits to identify/authenticate users — is categorized into two branches, namely, *fixed-text* authentication [5, 6, 7] and *continuous* authentication [8, 9]. Both types of authentication classify users based on the way in which they type the different characters making up a string, the only difference being that fixed-text authentication is based on short memorized strings (typically passwords), while continuous authentication is based on large chunks of text that users type while they freely interact with the computer. The keystroke attacks designed in this work are targeted against fixed-text keystroke authentication systems.

Touch-based authentication is a form of authentication in which touch patterns (such as swiping, zooming and clicking/tapping on a touch screen) are used to identify/authenticate users. Like keystroke authentication, touch-based authentication is also categorized into two branches: "entry point" authentication and continuous authentication. In continuous authentication users' touch gestures are monitored throughout a phone usage session [10, 11, 12]. In "entry point" authentication on the other hand, users are authenticated based on how they execute a certain (possibly secret) gesture at the entry point to an application or to the phone itself (i.e., login) [13, 14, 15]. This work focuses on continuous touch-based authentication.

For both keystroke and touch-based authentication, we first perform a statistical evaluation of biometric data collected from a large population of users, before using the observed statistical traits to design and launch statistical attacks on users' templates. In practice the keystroke statistical attack designed in this work would be launched with the aid of bots— a class of rogue applications that are now well understood to have the capacity to emulate keystrokes [16], based on programs like `xsendkeycode` [17] for the X Window system and APIs such as `SendInput` [18] for Microsoft Windows. Given a bot designed to mimic human typing, a motivated attacker who has access to a sizeable amount of users' typing data could extract representative model information from one population, and use it as input to the bot so as to attack users' keystroke profiles from any other population.

Since the dynamics of how a bot could generate and submit fake keystrokes at a verifier have been explored in recent literature [16], we do not implement a live bot in this study. Instead we evaluate a feature-level attack under the assumption of a password-keystroke co-authentication system for which the attacker has accessed the victim's password, possesses the required software tools, and is only left with the task of synthetically generating the keystroke sequence corresponding to the user's profile. Our assumption of a stolen password is not uncommon in security evaluations of biometrics systems "as it allows evaluators to better understand how much extra security the biometric adds to the strength of the password [19]". In fact, we argue that on the basis of the current prevalence of attacks launched to steal authentication data from central storage servers[1], it is not unlikely that a password-keystroke system could be faced with adversaries who already have knowledge of the victim's password.

To launch our statistical attack on a touch-based authentication system in practice, we used a robot to execute the statistically fine-tuned touch gestures on the screen. While there exist a wide range of robots that could be used for this purpose, we assume that it would be infeasible for the adversary to use a very expensive and sophisticated robot (that could cost thousands of dollars) for the sake of breaking the security of a stolen touch screen device (*e.g.*, phone). For this reason, we implemented our statistical attack using the standard Lego Mindstorms NXT robotic kit [21], a very cheap robot that can easily be programmed to perform swiping and clicking operations[2]. The simplicity of this robot convinces us that our attack could easily get embraced by adversaries if continuous touch-based authentication became widely deployed. Additionally, the fact that the attack is launched in the analog domain implies that it cannot easily be stopped by conventional software solutions like would be the case for malware-based attacks.

1.2 Dissertation Contributions

In this dissertation we analyze two large behavioral biometrics data repositories — one of which a keystroke dataset and the other a touch gestures dataset — for vulnerabilities that can be exploited by adversaries to attack the associated authentication systems. We then design a family of attacks that leverage these weaknesses

[1]One recent example of a large-scale attack is described in [20]

[2]Research on continuous touch-based authentication primarily revolves around three frequently occurring gestures — clicking (or tapping), swiping to move the screen vertically, and swiping to move the screen horizontally [10, 11])

to break keystroke and touch-based authentication systems. Our contributions are described in greater detail below.

1. We design an algorithmic attack mechanism that exploits the instability of users' keystroke features to degrade the performance of a keystroke biometric system. Relative to the *zero-effort* attacks typically used to test the performance of keystroke biometric systems, we show that our algorithmic attack increases the Equal Error Rates (EERs) of three high performance keystroke verifiers by between 28.6% and 84.4%. Our results confirm that *zero-effort* impostor testing can underestimate the threat faced by a keystroke verifier in practice, and demonstrate the need for the incorporation of algorithmic attacks in the standard impostor testing routine of keystroke verifiers.

2. We introduce the notion of robotic attacks against touch-based authentication. While we use a Lego robot to emphasize that these attacks could be implemented at minimal cost, our core contribution is not with regard to a particular robot type or algorithm (one could use a more sophisticated robot to launch a high precision attack). Rather, its the illustration that robots (in general) are a much more realistic performance evaluation tool (than the currently used methods) for the fast emerging field of touch-based authentication. Relative to the traditional *zero-effort* attack, our robotic attack increased the mean EERs of the verification algorithms by between 338.8% and 1535.6% depending on the *failure to enroll* threshold and the type of stroke used for classification. Further, we found that the impact of the attack could not be significantly reduced by a

failure to enroll policy which bars the "poor" performing users from enrolling onto the system.

3. Putting aside the performance of the attacks, our work, by virtue of being the first to analyze the statistical attributes of a large keystroke dataset assembled over several years, could serve as a reference benchmark for studies that continue to be built around small numbers of users. This problem of keystroke research being predominantly based on small datasets has prompted questions on how the results reported from these experiments generalize to large keystroke systems in practice [22], and has, among other issues always called for a large-scale study whose findings can give some insights into the properties of keystroke data at scale. Our observations on the Gaussianity, discriminability and mutual information of keystroke features should address this gap for a number of research areas within keystroke.

4. Although the small size of our touch biometrics dataset (relative to the keystroke dataset) limits the rigor of our analysis, we present some statistics expressing variables such as: the regions of the screen on which most swiping is done, the pressure exerted on the screen, and the area of the finger touching the screen. These empirical results should play a role in enabling the community to better understand the dynamics of users' touch behavior.

1.3 Definitions and Terminology

In this section we define the various terminologies that are central to the methodology used in this dissertation. Some of these terms are further described when they are first used in the dissertation.

False Reject Rate (FRR): The proportion of genuine authentication attempts that the authentication system classifies as impostor attempts. Elsewhere in literature, this term is also referred to as the False Alarm Rate.

False Accept Rate (FAR): The proportion of impostor authentication attempts that the authentication system classifies as genuine attempts. Elsewhere in literature, this term is also referred to as the Impostor Pass Rate.

Equal Error Rate (EER): The error rate at which the FAR and FRR are equal.

Detection-Error (DET) Tradeoff Curve: The plot of FAR versus FRR or vice versa. The EER can be computed with the aid of this curve.

Classifier (or Verifier): A program which assigns a new observation to a given class based on training carried out on observations whose class membership is known.

Biometric template: A stored record of a user's biometric features. During testing (or authentication), a new biometric sample provided by the user is compared with the stored template (using a classifier) so as to determine whether the new sample indeed belongs to the user in question.

Null hypothesis (H_o): A claim that is to be subjected to a statistical test. The *alternative hypothesis* is the hypothesis contrary to the null hypothesis. Rejection of the null hypothesis implies acceptance of the alternative hypothesis.

Level of significance, or critical value, α: The probability that the null hypothesis is rejected when it is in fact true. It is also referred to as the Type I error.

P value: The probability of obtaining a test statistic that is at least as extreme as the one that was actually observed, given that the null hypothesis is true

A touch stroke (or swipe): The path taken by the finger on the touch screen. These two are not standard terms, and could hence assume completely different meanings in other literature.

Cyber-behavioral biometrics: A class of biometric modalities in which users are identified based on how they interact with computing devices (*e.g.*, desktop computers, phones, *etc.*). Examples of cyber-behavioral biometric modalities include: keystroke dynamics, touch behavior, web usage patterns, *etc.*

Zero-effort attack: A method of testing the performance of a biometric authentication system that uses samples generated by one user (i.e., the user designated as the impostor) to attack the template built for another user (i.e., the user designated as the genuine user, or victim). This attack method simulates a scenario in which the attacker does not make any attempt to imitate the victim's biometric footprint.

1.4 Organization of the Dissertation

In Chapter 2, we discuss the various past works which relate to the statistical analysis and attacks designed in this paper. In Chapters 3, 4 and 5, we respectively discuss our data collection and feature extraction, statistical analysis and the attack on the keystroke system. In Chapter 6 we discuss our data collection and feature extraction, statistical analysis and the attack on the touch-based authentication system. Finally in chapter 7, we give our conclusions and some indications of future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Related Work and Motivation Behind Touch-based Authentication

Over the past few years, the popularity and usage of mobile devices (i.e., smart phones, tablets, *etc.*,) has grown exponentially [23]. One of the key factors for the proliferation of these devices—their portability relative to the desktop computer—also unfortunately manifests as a major weakness from the point of view of physical security. The ease with which these devices can be carried around in their owners' pockets and (or) briefcases is the same ease with which they can be misplaced or stolen by adversaries. Once in the hands of a sophisticated attacker, both the remotely accessible resources and stored data on these devices (*eg.,* passwords, social security numbers, bank details, private emails, company secrets, *etc.*,) could easily be compromised, potentially resulting into catastrophic consequences for businesses and (or) individuals.

Currently, the most widely employed defense against such threats is the PIN lock mechanism. However, this mechanism is incorrectly used by some users (eg., by setting very long timeouts [10]), completely disengaged by others [24], and susceptible to several attacks even when users engage it in accordance with the best practices [25, 26]. To augment the single line of defence offered by the PIN lock, researchers have recently studied the possibility of continuously authenticating users after the initial login phase is completed [27].

Among the continuous authentication approaches that have been explored, touch-based authentication has attracted a lot of attention given that it revolves around touch gestures that users execute during their routine operations on the phone [10, 11, 12]. Touch gestures arise naturally from operations such as scrolling, zooming and clicking, and can thus be used by an authentication application without requiring the user to pay attention to the authentication process.

In a recent Active Authentication (AA) research drive championed by the Defense Advanced Research Projects Agency (DARPA) [28], touch gestures have been identified as one of the candidate biometric modalities that could be built into a pilot multi-modal "biometric platform [28]" to be deployed in IT devices at the Department of Defense (DoD). With the American government actively joining the stake-holders interested in evaluating the potential of touch-based authentication, there is now little doubt that interest in this area of research is only bound to increase.

As already mentioned in Chapter 1, research on touch-based authentication is categorized into two groups: 1) authentication mechanisms in which touch gestures are used for authentication at an entry point (*e.g.*, at login), and, 2) authentication mechanisms in which touch gestures are extracted continuously as the user performs various tasks on the phone. The former category includes studies in which users touch behavior is analyzed based on a set of canonically defined gestures [13, 14] or gestures strictly captured at the unlock screen [15].

"Entry point" touch-based authentication has several operational dissimilarities with continuous touch-based authentication. Perhaps the most notable of these is the fact that the known geometry of the hand can be easily matched with the strictly defined structure of a gesture to ensure that only touch points associated with similar fingers (say, a thumb in the template and a thumb presented during testing) are compared during "entry point" authentication [13]. Such kinds of assumptions can not be made with continuous authentication where users freely interact with the phones, touching them with whatever fingers and in whatever way they find comfortable. Because the attacks designed in this work are targeted against continuous touch-based authentication systems, we delve deeper into past works which studied this type of authentication.

Using a dataset of 41 users, Frank *et al.* [10] obtained Equal Error Rates (EERs) of between 0 and 4% when a k-Nearest Neighbors (k-NN) classifier and a Support Vector Machine (SVM) were used to continuously authenticate users based on their touch gestures. The study was based on 30 features extracted while users swiped/scrolled (to move the screen vertically or horizontally) as they read text and browsed images. In [12], a digital sensor glove was shown to enhance the performance of a touch gesture-based continuous authentication system. Using a decision tree, Random Forest and Bayes Net classifier, the authors showed that the glove reduced the error rates seen during authentication. For instance, for the Bayes Net classifier, a False Accept Rate (FAR) of 11.96% and a False Reject Rate (FRR) of 8.53% respectively reduced to 2.14% and 1.63% when the glove was used. Similar improvements were noted for the two other classifiers.

More recently, Li *et al.* [11] evaluated the performance of a live implementation of a touch-based authentication system on a mobile phone. Leveraging a "hack into the lower layer of an Android system [11]", the system monitored touch gestures across all applications installed on the phone. Based on a group of 75 users who were allowed to freely interact with the phones for days, the SVM-based authentication system was shown to attain classification accuracies as high as 95%.

All three papers cited above employ a zero-effort testing routine in which the system's resistance to attack is gauged based on simplistic attacks in which samples from a subset of the population are used to attack samples drawn from a given user. It is on this front that this work advances the state-of-the-art, studying the impact that sophisticated adversaries could have on this type of authentication.

2.2 Related Work and Motivation Behind Keystroke Authentication

Unlike touch-based authentication which traces its roots to just a few years ago (following the emergence of touch screen devices), keystroke dynamics dates back several decades ago [29, 30]. Right from the earliest works on keystroke dynamics, the categorization between fixed-text and continuous keystroke dynamics was apparent: Forsen *et al.*'s work in 1977 [29] was based on a small group of users who typed each other's names (i.e., fixed-text) while Gaines *et al.*'s analysis in 1980 [30] was based on pages of text typed by the users (i.e., continuous or free text). Over the past few decades, both streams of keystroke dynamics have seen a tremendous amount of research (see detailed keystroke dynamics history in [31]), with fixed-text authentication being evaluated for its potential to add a second layer of defense to the password at login-time [2, 32, 5, 6]), and continuous authentication being evaluated as a means to repeatedly authenticate a user after the initial login is completed [8, 33, 34, 35].

With the vast majority of research on this topic assuming a zero-effort threat model, there are only a handful papers which directly relate to the algorithmic attack problem addressed in this work. Investigating synthetic attacks against keystroke systems, Khandakher *et al.* [36] present an attack called a *Snoop-Forge-Replay attack* in which an adversary snoops on a victim's typing session using a keylogger, uses the captured data to build a fake template for the user (with the aid of outlier filtering in some cases), and then replays the data to defeat the verification mechanism. The attack was shown to induce increments in EER (relative to the zero-effort baseline) of between 69.33 % and 2730.55 % depending on parameters such as the amount of text snooped, the outlier filtering policy and the classifier used for authentication.

The attacks in [37] and [7] build on the same idea used in [36] as they both use a keylogger to steal a user's typing latencies before using the stolen data as a source of input for an attack against the same user. The attack in [37] uses the captured latencies to train human impostors who later attack the victim's keystroke profile, while the attack in [7] uses these latencies as a basis for systematically morphing the victim's template into a weaker template that can easily succumb to attack. The major difference between these works and our research is that our generative algorithm uses general information on how a typical user would type a given string, and *does not depend on text snooped from the victim.* It is thus not surprising that the attacks in [36], [37] and [7] attain much higher success rates than our attack (i.e., relative to the zero-effort attack, the increments in EER were as high as 2730.55%, 395% and 305% in [36], [37] and [7] respectively compared to a maximum increment of 84.4% in our work). That said, our belief is that the relative ease of accessing general population keystroke statistics as compared to the intricacies of snooping on the typing session of an intended victim should make our model of attack more appealing to attackers in practice.

In [16], it was shown that an authentication system (called *TUBA* [16]) using a Support Vector Machine (SVM) for keystroke verification was able to repel a form of statistical attack. Unfortunately, the authors did not publish the parameters used to set up the SVM verifier, as they stated that "our tuning approach was more along the lines of brute-force, and we thus do not show the final chosen parameters [16]". We do not run our attack against the verifier used in [16], since it is very difficult to make a meaningful comparison without a set of common parameter settings. However, we evaluate our attack against three state-of-the-art keystroke verification algorithms (details in Section 5.4) that have been demonstrated to be among the best for fixed text authentication, do not require sophisticated tuning of parameters, and whose implementation details we have clearly laid out for other researchers who may seek to evaluate their statistical attacks against the same set of verifiers.

Below, we discuss four major aspects that put our algorithmic attack apart from that implemented in [16]:

1. *Feature Distribution Assumptions*—The attack in [16] is built under the assumption that keystroke features follow a Gaussian distribution. However, as we show in this work (Section 4), this assumption is suspect since we find none of the

115 features extracted from our samples to be Gaussian. With Stefan *et al.* [16] not having tested their samples for Gaussianity, it is difficult to tell whether their conclusion about keystroke dynamics being resilient to synthetic forgeries could have been impacted by the Gaussianity assumption. Our attack does not make any assumptions about the keystroke feature distributions.

2. *Size of User-population and Duration of Data Collection*—Stefan *et al.* [16] estimated the parameters of their assumed Gaussian distribution, and implemented their attack based on data collected from a small group of 20 users. Even if the keystroke features had indeed been Gaussian, parameter estimates from a dataset of 20 users are unlikely to accurately represent the underlying distribution. Also, as we show in Section 5.5, the dip in mean system performance seen under our statistical attacks primarily originated from a small proportion of about 10% to 20 % of the full population who badly succumbed to the attack. With a small set of 20 users, the likelihood of capturing the full variety of typing traits naturally diminishes, and it is thus not so surprising that the work in [16] appears to have failed to reflect the impact of the "poor" users who negatively impacted the overall system performance in our experiments. The fact that the dataset in [16] was collected over a short period of time also made it hard to get a realistic view of the attack performance, since the inconsistency and long-term evolution of users' features that a statistical attack would typically be expected to exploit was obviously not reflected in the users' keystroke data samples.

3. *Attack Design*—The attack in [16] did not use any additional keystroke-feature properties (besides the assumed Gaussian behavior of keystroke features) to aid the feature space enumeration process. Our attack exploits the feature distributions, discriminability and dependencies between keystroke features to traverse the search space as intelligently as possible. We believe that the usage of a wider range of keystroke feature properties during attack design should make our statistical attack more rigorous than that implemented by Stefan *et al.* [16].

4. *Keystroke Features Used*— The attack in [16] was based on a relatively new feature-set[1], which despite extending our understanding of the different features that a keystroke system could use, left behind the question: How would the traditional keystroke features (described in Section 4) on which most proposed keystroke systems have been based, perform against synthetic forgeries ? Our work addresses this question by subjecting these (traditional) features to the attacks.

The works in [38, 39], despite being based on handwriting biometrics evaluate the performance of a wide range of synthetic attacks in an environment similar to ours. In particular, the *generative attack* whose input is either extracted from general population handwriting statistics or from text written by the victim in a context different from that of the exact word (or phrase) subject to attack, closely relates to our work by virtue of using an automated algorithmic approach against short

[1]From a 14-character string, they extract a 121 dimensional feature vector which they later reduce through Principal Component Analysis

strings of text comparable to the *password-like* strings used in this paper. For certain categories of users, this attack was shown to considerably outperform attacks launched by trained human forgers in a verification mechanism which included both human judges and an automated reference monitor.

These findings motivate the work in this paper, although it is noteworthy that the operational dissimilarities between keystroke dynamics and handwriting mean that our algorithmic framework can not directly derive from the approach in [38]. Another difference between ours and Ballard *et al.*'s [38] work, is that we do not extract any input from strings typed by the intended victim, since we focus on an attacker who only uses global typing traits to launch local attacks against individuals.

With regard to the keystroke statistical traits that we investigate to guide the design of our attack algorithm, Janakiraman *et al.* [40] also studied the discriminability of keystroke feature vectors as we do in this paper. However, that work was not based on *fixed text* (see detailed description of *fixed text* in Section 3.2) and used a small dataset built from samples collected from a group of 22 users. Meanwhile for the discriminability analysis performed by Balagani *et al.* [5], the authors used *fixed text* like in this paper, but again used a small population of 33 users and compared discriminability between heterogeneous and aggregate keystroke feature vectors, as opposed to a feature by feature evaluation which we use in this work. With all that said, the performance analysis methodology used in [5, 40] differs from the approach taken in this paper, since none of the two works subjected the observed differences difference in discriminative power seen across features to formal statistical tests of significance. The absence of formal statistical analysis particularly makes it hard to generalize the findings (such as the means and standard deviations reported in [40]) to keystroke dynamics in general [41].

CHAPTER 3

KEYSTROKE EXPERIMENTS

3.1 Overview

We collected several thousand typing samples in four 3-week events, spread over a 2-year period between 2009 and 2011. All typists were staff, students and faculty of Louisiana Tech University. Table 3.1 summarizes the details of the dataset used in our experiments.

Table 3.1: Summary of dataset details

Date	**Gender**		**First Language**		**Handedness**		
	Male	Female	English	Other	Left	Right	Ambidextrous
Oct 2009	589	412	-	-	-	-	-
Apr 2010	690	488	967	205	100	1051	22
Oct 2010	692	507	974	216	112	1052	21
Oct 2011	715	521	1007	221	117	1046	24

The missing values are because some details were not captured during the first phase of experiments when the project had just been initiated. Observe that the total number of users may appear inconsistent in some of the phases. For example, the sum of male and female typists in the April-2010 dataset is 1178, yet the sum of English and non English speakers for the same phase of experiments is 1172. The reason for the discrepancy is that some users opted not to fill certain fields of the questionnaire handed out before the typing exercise. The sum of female and male typists however reflects the actual total typing population for each phase.

3.2 Typing Samples

While a wide range of samples were collected during the study, this work focuses on samples which were collected as *fixed text.* With *fixed text,* a user types a memorized word or very short sentence that constitutes little or no cognitive load. Such text closely mimics basic authentication in which a password or short passphrase may be typed by a user at log-in time. The ideal way to collect such text is by having users type their actual passwords, for which they must have developed a natural typing pattern over time. Unfortunately this option has major security implications and is generally not used in keystroke research. Past works simulated *fixed text* entry in two ways– some studies used a common password string (typically having about 8-10 characters) across all users [6, 2], while others used a short sentence that could be easily memorized by users [5]. The usage of a sentence in the latter category of works was mainly motivated by the need to investigate how some modalities of *fixed text* keystroke authentication (*e.g.*, classification accuracies) depend on variables such as the identities of characters making up users' passwords (or passphrases), or authentication-string lengths such as in [5]. In such cases, the limited number of characters in a short password string would limit the scope of analysis.

Because our work involved investigations on a wide range of variables, we also used a sentence to simulate *fixed text* entry, and had users type the phrase "*I am an Undergraduate Student of Louisiana Tech University*". Our belief is that subsets of this phrase should give a plausible approximation of the nature of real passwords. Also, with most of the users in our study being undergraduate students of Louisiana Tech University, we figured that the words in this phrase would be familiar and easy to memorize.

To simulate password entry in our experiments, it was crucial that the character distribution in our *fixed text* reflected the character distribution seen in real passwords. Table 3.2 compares the frequencies of the different characters in our typing sample with those in three recently hacked password lists (details of the three lists — *i.e.*, the *Singles.org*, *Myspace.com* and *phpBB.com* password lists can be found in [20, 42]). The tabulated summary of the password character percentages is compiled from the statistics published in [42].

Table 3.2: Frequencies of the most common characters in passwords found on 3 recently hacked password lists compared with frequencies of the same characters in the phrase studied in this work.

Character	**Password List**			**Phrase Used in this Work**
	Singles.org	Myspace.com	phpBB.com	
e	8.84%	7.71%	8.95%	10.0%
a	8.13%	7.00%	8.79%	12.0%
o	6.01%	5.46%	6.32%	4.00%
s	5.60%	4.89%	5.93%	6.00%
i	5.42%	4.84%	5.24%	8.00%
n	5.18%	4.28%	5.32%	10.00%
r	5.08%	4.69%	6.13%	6.00%
t	3.78%	3.55%	4.78%	10.00%
u	2.54%	2.29%	3.26%	10.00%
TOTAL	**50.58%**	**44.71%**	**49.4%**	**76%**

For the password lists, the 3 tabulated values in each row represent the number of times a character appears on the list as a percentage of the total number of characters on the list. The value tabulated for our phrase is the number of times a character appears in the phrase as a percentage of the total number of characters in the phrase. Our percentages do not exactly match those in the lists, although it is clear that we captured many of the most frequently occurring characters on the password lists. Note that the very low frequency of special characters in the hacked password lists (full statistics can be found in [42]) supports the omission of these characters from our study.

During the typing sessions, users easily memorized the phrase, and quickly got into their regular typing rhythm after just a few trials. Whenever the system detected an error after the user had typed the full string, the parser prompted the user to re-enter the string afresh, like is done in regular password entry[1]. All typing was done using DELL QWERTY keyboards. Across the several typing phases, users provided 12 to 20 samples of this string during each typing session.

3.3 Keystroke Features and Pre-processing Method

3.3.1 Keystroke Features

For every key typed by a user, there are two associated time stamps—the time when the key is pressed, and the time when the key is released. Figure 3.1 illustrates these time stamps for the digraph HI, with P_H representing the time when the letter H is pressed, R_H representing the time when letter H is released, and the time stamps for the letter I defined similarly. For the digraph HI, three independent features can be derived from the raw time stamps. These are the Key Hold Time, KHT_H of letter

[1]The program used during the last phase of experiments did not have this parsing module. For data collected during that phase, we scanned the data after it was collected and eliminated a given typing instance of the passphrase if it had at least one error.

H, the Key Interval Time, KIT_{HI} between H and I, and the Key Hold Time, KHT_I of letter I. These features can respectively be calculated as: $KHT_H = R_H - P_H$, $KIT_{HI} = P_I - R_H$ and $KHT_I = R_I - P_I$, with KIT_{HI} assuming a negative value if I is pressed before H is released.

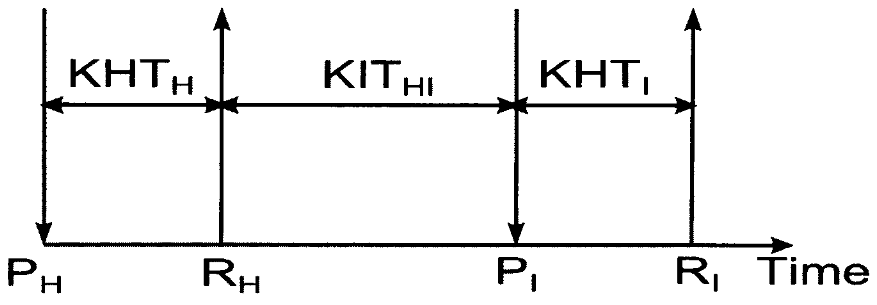


Figure 3.1: An illustration of the "atomic" features used in keystroke dynamics.

From these three "atomic" features, a number of other features can be derived to represent a user's typing pattern for the digraph HI. Examples of these features include the *up – up* time ($= R_I - R_H$ or $KHT_I + KIT_{HI}$), the *down – down* time ($= P_I - P_H$ or $KHT_H + KIT_{HI}$), and the total time required to type the full digraph ($= R_I - P_H$ or $KHT_H + KIT_{HI} + KHT_I$). For longer words, additional features can be derived from the "atomic" features to express the time interval between any sequence of adjacent characters within the word. For instance in a word made of 5 characters, n-graphs can be defined (*e.g.*, trigraphs, 4-graphs, 5-graphs [40, 43]) in addition to the set of features already described above. Because the bulk of keystroke features are just linear combinations of the "atomic" features (i.e., the $KHTs$ and $KITs$), most keystroke verification systems are designed to use these two features (for summary statistics of which studies have used which features refer to [5, 6]). In order to make a thorough analysis while avoiding duplications, this dissertation thus also focuses on the $KHTs$ and $KITs$. We believe that their properties will provide insights into the properties of the other kinds of features.

3.3.2 Outlier Filtering

Over the several samples collected from each user, uncharacteristically long pauses could occur at various points in the string and pollute the user's template if not filtered out. For each feature, we use the distance-based outlier detection method in [44] to filter out outliers from each user's samples before using the data for our statistical analysis. In this method, a point is considered an inlier if 68% of all the points are within 100 ms of it. These thresholds were fixed heuristically in [44], and have been found to perform well in a number of other works (see an example in [9]).

CHAPTER 4

KEYSTROKE FEATURE PROPERTIES

In this section we present the keystroke feature traits seen across our dataset. We discuss their implications to keystroke dynamics research and to the design of statistical attacks in particular. For all statistical tests performed, we report results based on a critical value of $\alpha = 5\%$.

4.1 Distribution of Key Hold and Inter-key Times

Observation#1: *All KHTs and KITs extracted from our fixed-phrase, "I am an undergraduate student of Louisiana Tech University", did not obey a Gaussian distribution*

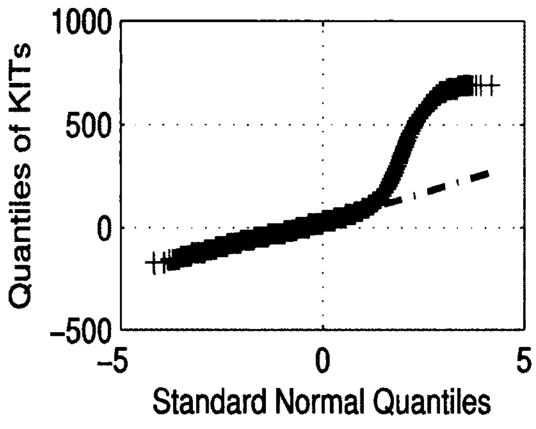
Evidence to Support Observation #1: We used the Lilliefors [45] and Cramer-von Mises [46] tests to formally check whether keystroke feature data follows the normal distribution. A modification of the Kolmogorov Smirnov (K-S) test [47], the Lilliefors test returns a more accurate P-value (than the K-S test) when the parameters of the hypothesized distribution are not completely specified during the test [45]. We included the Cramer-von Mises test in our hypothesis testing routine for the purpose of checking the result returned by the Lilliefors test, since different categories of normality tests may sometimes fail to agree on the distribution followed by a given dataset. For both normality tests and other statistical tests performed in this work, we used the *R* [48] statistical programming environment.

We ran the two normality tests for all features extracted from the fixed-phrase (57 KITs and 58 KHTs). For instance, to test whether the KITs of digraph *GR* followed a Gaussian distribution, we created a vector V_{GR} containing every user's latencies for digraph *GR*, from which we derived another vector V'_{GR}, containing 5000 latencies that were randomly selected from V_{GR}. The normality tests were performed on the vector V'_{GR}. For every feature extracted from the fixed-phrase, we repeated the process as done for digraph *GR*.

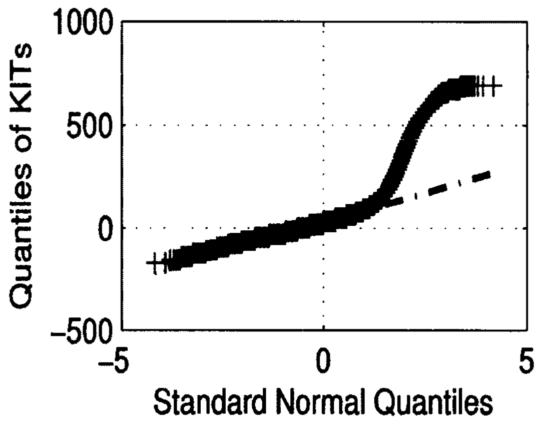
Our method of performing the hypothesis testing on random sub-samples (rather than the full population), has also been used in past work [49, 50], and is motivated by the fact that large datasets tend to have statistically inexact descriptions, which in turn makes it hard for a goodness-of-fit test to produce meaningful results if directly applied to the whole dataset [49]. For each normality test performed, our null hypothesis was that the latencies in the test vector came from the normal distribution. The alternative hypothesis was that the elements of the vector did not follow the normal distribution. For each vector tested, we rejected the null hypothesis for both the Lilliefors and Cramer-von Mises tests.

Note that while we perform a very large number of tests, we do not make any corrections (such as Bonferroni) on the critical values (of the Gaussianity tests and all other tests performed in the study) because each test checks a different hypothesis. For instance since we aim to study the distribution followed by each individual feature in our test-phrase, the test whether digraph *GR* followed a Gaussian distribution is distinct from the test whether digraph *LO* followed a Gaussian distribution. For this reason our result-reporting throughout the dissertation is centered around identifying and counting the number of features of a certain type for which a certain hypothesis holds or fails to hold.

Figure 4.1 shows the quantile-quantile(Q-Q) plots [51] for two features which illustrate the general trend seen across the dataset (See [52] for more visualizations of feature distributions.). A Q-Q plot compares the quantiles of one sample against the quantiles of another. If the samples come from the same distribution, the plot will be linear even if one distribution is shifted or re-scaled from the other. In our plots, keystroke feature data is compared with samples drawn from the Gaussian distribution. For instance the Q-Q plot in Figure 4.1a compares the KITs of digraph *SI* with data generated from the standard normal distribution.



(a) Q-Q plot for KITs of digraph SI



(b) Q-Q plot for the KHTs of letter T

Figure 4.1: Q-Q plots demonstrating the goodness-of-fit of the normal distribution for selected keystroke features.

The figure shows that the KITs of digraph *SI* (Figure 4.1a) depicted a significant positive skew as evidenced from the sharp departure from the straight (normal) line. On the other hand, the KHTs of letter T (Figure 4.1b) demonstrated a less pronounced departure from the normal distribution, although both features ultimately failed the normality test. We observed the trait depicted in Figures 4.1a and 4.1b across many features and concluded that some features were more *Gaussian-like* than others despite all features returning very low P values.

Impact of Observation #1 on Keystroke Research: The fact that all the features in our fixed-phrase fail the Gaussian test suggests that past studies such as [53, 16] which have built entirely on the Gaussian assumption across all features, could see improved results if features derived from the typing samples used in those works had been closely studied to determine which ones are more accurately modeled by the Gaussian distribution.

For the designer of a statistical attack, this non-Gaussian behavior explains why a simple generative model that uses the means and standard deviations as the central reference parameters during forgery generation may not always work well. In the attacks launched in this work, these findings prompt us to focus on a non-parametric attack-design, in which we explicitly work with individual feature histograms, without globally assuming Gaussian behavior of the keystroke features.

4.2 Keystroke Feature Discriminability

In this section we investigate the discriminative power of the different keystroke features extracted from our typing samples.

Observation#2: *In comparisons made between each KIT and each KHT extracted from our fixed phrase, we rejected the null hypothesis that a KHT was as discriminative as a KIT in favor of the alternative hypothesis that a KIT was more discriminative than a KHT in 98% of the comparisons. Also, we found that certain KITs had considerably higher discriminative power than the rest of the KITs, just like certain KHTs had considerably higher discriminative power than the rest of the KHTs*

Evidence to Support Observation#2: To study feature discriminability, we use the Bhattacharyya distance metric [40] to estimate the extent of overlap between the pdfs of users' features. Equation 4.1 shows the definition of the B-hattacharyya distance, D_B for the pdfs $u_i(x)$ and $u_j(x)$. A Bhattacharyya distance of 1 means that two pdfs overlap completely while a distance of 0 means the pdfs do not overlap at all. The more the overlap, the poorer the discriminability of the feature in question for the pair of users under study.

$$D_B = \int (u_i(x) * u_j(x))^{\frac{1}{2}} \, dx, \tag{4.1}$$

The first step in the D_B computation is to empirically estimate the pdfs, $u_i(x)$ and $u_j(x)$, representing the latencies of users i and j for a given feature. To this end we use the binning approach advocated by [40], in which the clock resolution used to time-stamp keystroke events is set as the bin size. With both $u_i(x)$ and $u_j(x)$ partitioned into bins, we multiply probabilities associated with corresponding bins, take the square root of each product, and then sum the results over all bins to obtain D_B between the two pdfs. This discrete implementation of Equation 4.1 was also used in [40].

For each of the 115 features in our dataset, we compute D_B as explained above, using a set of 3000 randomly selected user-pairs. Each user-pair (represented by the pdfs $u_i(x)$ and $u_j(x)$) results into a single value of D_B, which means that our computation produces a vector containing 3000 D_B values for each feature. Let D_{f_1} denote the vector of Bhattacharyya distances associated with the feature f_1, and D_{f_2} denote a similar vector for the feature f_2. Each of these vectors contains 3000 D_B values for the 3000 user-pairs. The vectors are such that the user-pair corresponding to the D_B value in the i^{th} position in vector D_{f_1} is the same user-pair corresponding to the i^{th} entry in vector D_{f_2}. The i^{th} element in the vector $\Delta D_f = D_{f_1} - D_{f_2}$ is thus the difference between the discriminabilities (Bhattacharyya distances) of features f_1 and f_2 for the i^{th} pair of users.

To determine whether the vector ΔD_f pointed to a significantly large difference in discriminability between the features f_1, and f_2, we used the Wilcoxon signed-rank test [54]. We zeroed on this test after finding that the differences-vectors across the population were far from Gaussian (based on observation of Q-Q plots and P values returned by Lilliefors normality test). The test generally "measures the tendency of one sample to contain values that are larger than those in another sample [55]", and is known to be robust when the testing distributions are non-normal [6]. Even where the parent populations are Gaussian, this test does not perform much worse than the t-test [56, 57].

A critical requirement of this test is that the vector ΔD_f should be symmetrically distributed around some median. Real-world data being rarely perfectly symmetric however, the test is often applied when data is approximately symmetric

[58], since the less restrictive alternative non-parametric test (i.e., the Sign test [59]), is generally considered less powerful[1] than the Wilcoxon signed-rank test. In this work we used the rules of thumb in [60] (in addition to visual inspection of histograms in some cases) to check for the symmetry of our ΔD_f vectors. According to these rules, a distribution is considered *approximately symmetric* if its skewness is between -0.5 and 0.5, *moderately skewed* if its skewness is between -1 and -0.5 or between 1 and 0.5, and *highly skewed* if the skewness is less than -1 or greater than 1.

When f_1 and f_2 were both KHTs, all ΔD_f vectors that we computed were *approximately symmetric.* Meanwhile in cases where both f_1 and f_2 were KITs, the vast majority ($\approx$ 95%) of ΔD_f vectors were *approximately symmetric*, with a few ($\approx$ 5%) being *moderately skewed.* Cases where f_1 was a KHT and f_2 was a KIT exhibited behavior that was in between the previous two cases. These results prompted us to conclude that the Wilcoxon signed-rank test was appropriate for our data. (See Appendix A for some symmetry results highlights).

The discriminability investigations conducted in this work were divided into two parts: In the first part, we sought to establish the extent to which the discriminative power of KITs generally compared to that of KHTs. As such, we made our computations in such a way that for each pair of features compared, f_1 was a KHT while f_2 was a KIT. The null hypothesis for the Wilcoxon signed-rank test performed on each of these pairs was that the vector $\Delta D_f = D_{f_1} - D_{f_2}$ followed a continuous symmetric distribution with zero median, which implied that the difference

[1]The lower power of this test relative to the Wilcoxon signed-rank test is mostly attributed to the fact that it uses limited information about the data, as it only takes into consideration the arithmetic signs of the elements in ΔD_f, and not their magnitudes [59].

ΔD_f between the Bhattacharyya distance-vectors of features f_1 and f_2 over the population was insignificant. The alternative hypothesis was that ΔD_f, came from a continuous symmetric distribution with median greater than zero, which implied that the KHT represented by f_1 had higher Bhattacharyya distances (and hence lower discriminability) than the KIT represented by f_2 over the population .

Since our typing sample contained 57 KITs and 58 KHTs, we made a total of 3306 (= 57 $\times$ 58) hypothesis tests since each KIT was tested against each KHT. We rejected the null hypothesis in favor of the alternative hypothesis in about 98% (3239 of 3306) of the tests, an indication that for the vast majority of tests, we could not find evidence to suggest that a KHT was as discriminative as a KIT. Figure 4.2 provides a visual perspective of how the discriminability of KHTs in our test-phrase compared to that of the KITs.

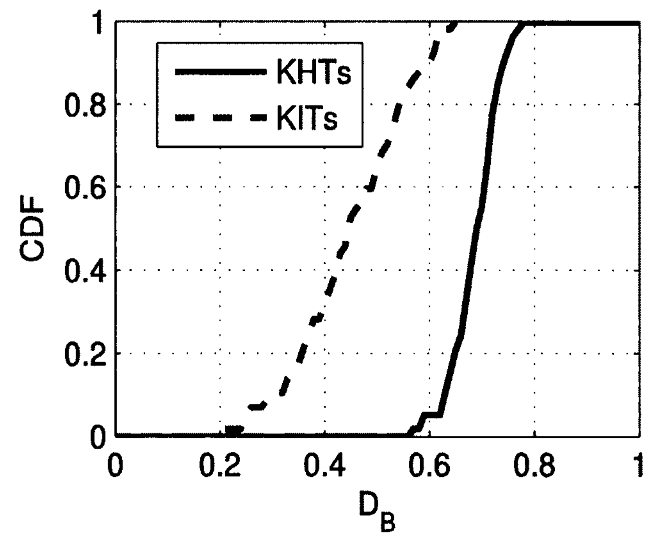


Figure 4.2: Comparing mean Battacharyya distances associated with KITs with those associated with KHTs in our typing samples

The figure compares the mean Bhattacharyya distances of the KITs side-by-side with those of the KHTs extracted from our typing samples over the population. Following the notation used throughout this section, the mean Bhattacharyya distance of a feature f_1 is computed by dividing the sum of the elements in vector D_{f_1} by 3000. This value will give some measure of how a feature such as f_1 discriminated between each pair of users over the 3000 user-pairs. For each of the 58 KHTs in our typing samples, we compute this mean value, and plot a CDF of the full array of mean values on Figure 4.2. We do the same for the 57 KITs in our sample.

As shown in Figure 4.2, KITs were more discriminative on average, as over 80% of them were associated with a mean Bhattacharyya distance of less than 0.6, while about the same percentage were associated with a mean Bhattacharyya distance of more than 0.6 for the case of the KHTs. These results support our findings from the hypothesis tests, as they provide confirmation that the KITs were more discriminative than the KHTs.

In our further investigations, we sought to establish how discriminability varied across KITs and across KHTs. This way, we should be able to determine if any KITs were considerably more (or less) discriminative than the rest of the KITs, and if any KHTs were considerably more (or less) discriminative than the rest of the KHTs. We thus performed a set of hypothesis tests in which both f_1 and f_2 were KITs or KHTs.

Due to space limitations we only present results from these tests for a few pairs of KHTs and KITs that are enough to support our conclusions on the disparity in the discriminability of different keystroke features. Based on P values that were approximately equal to zero, we rejected the hypotheses that[2]: 1) digraph UI was as discriminative as digraph $-O$, 2) digraph IS was as discriminative as digraph $-O$, 3) digraph DE was as discriminative as digraph OF, 4) letter S was as discriminative as letter M, and 5) letter N was as discriminative as letter I. Across the full dataset, we observed a number of features which were significantly much more discriminative than the others.

Impact of Observation #2 on Keystroke Research: Over the several decades of research on keystroke dynamics, very few papers (such as in [61, 16]) have applied feature selection during the keystroke template building process. With results in this section revealing certain features in our test-phrase to be significantly more discriminative than others, our work should motivate research on how feature selection could be employed to build users' profiles based on the most discriminative features. We believe that this direction of research could potentially improve the performance of keystroke verification systems.

Specific to the statistical attacks launched in this work, these findings on feature discriminability will be crucial for our feature-space enumeration strategy. Details of this strategy are discussed in Section 5.2 during our description of the attack.

[2]The hyphen represents the space character

4.3 Inter-Feature Dependencies

In this section we study the dependencies between the keystroke features.

Observation #3: *For a large number of feature-pairs in our fixed phrase, we found evidence to indicate that one feature depended on the other.*

Evidence to Support Observation #3: To study the dependencies exhibited by our keystroke data, we computed the mutual information between keystroke features. In contrast to measures such as the Pearson correlation which are sensitive to linear dependencies, mutual information measures correlation in general terms, and is sensitive to both linear and non-linear associations between variables. For two random variables X and Y, the mutual information $I(X;Y)$ is defined as:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} P_{XY}(x,y) log_2 \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)}, \tag{4.2}$$

where P_X and P_Y are respectively the probability mass functions (*pmf*) of X and Y, and P_{XY} is the *pmf* of the joint distribution between the two random variables. $I(X;Y)$ expresses the reduction in uncertainity of variable X given variable Y. In our experiments, the random variables X and Y correspond to two different keystroke features. Details of how we pre-process the keystroke feature data for $I(X;Y)$ computation follow:

Let $f_{i,j}$ denote the value of the j^{th} feature during the i^{th} typing attempt of a user, where $1 \le i \le n$, $1 \le j \le m$. The vector $V = \frac{1}{n} \cdot (\sum_{i=1}^{n} f_{i,1} \quad \sum_{i=1}^{n} f_{i,2} \quad \sum_{i=1}^{n} f_{i,3} \cdots \sum_{i=1}^{n} f_{i,m})$ is the feature-means vector for each user. For many keystroke verification algorithms, this vector is the main building block of a user's profile (*e.g.*, see [6] for a survey), and is the basis against which feature vectors extracted from later typing attempts are judged to match (or not match) the user's typing pattern. We use this vector to represent each user's typing pattern during our mutual information computations.

For each user we first compute the vector V, and then create a 2 dimensional matrix, M, whose every row is the vector V, computed for a different user. For a group of k users, M is a $k \times m$ matrix, for which the j^{th} column $(1 \leq j \leq m)$ is a vector, U_j, in which each element is the mean value of feature j for one of the k users. Our computation of $I(X;Y)$ between keystroke features will be based on pairs of the vectors U_j, since each of these vectors represents a single feature over the population.

For a pair of features identified by the indices $j = 1$ and $j = 2$, we first bin the associated vectors U_1 and U_2 (corresponding to X and Y in Equation 4.2) using the approach described in Section 4.2, before applying Equation 4.2 to calculate the mutual information between the two feature vectors. To determine whether the amount of mutual information between U_1 and U_2 is statistically significant, we perform the mutual information permutation test [62, 63], with the null hypothesis being that the expected mutual information between the two vectors is zero (i.e., that the two vectors are independent). The alternative hypothesis is that the two vectors are dependent on each other. Since our test phrase has 115 features (=57KITs and 58KHTs) there are $\binom{115}{2} = 6555$ possible feature pairings. We run this test for each of the 6555 feature-pairs.

Before presenting the results, we define what we term as *similar* features. We refer to multiple instances of a given digraph, (or onegraph) in a word as a set of *similar features.* For example, since the word *STUDENT* has two instances of the letter T, we refer to a feature-pair comprised of the KHTs of the two Ts as having *similar* features. We report results from the mutual information tests on such feature-pairs separately from those of the rest of the feature-pairs comprised of *dissimilar* features because high amounts of mutual information between similar features could give a deceptive view of how keystroke features depended on each other in general.

For tests run on feature-pairs comprising of *similar* features, we rejected the null hypothesis (i.e., the hypothesis that vectors in a pair were independent) in 85% of the tests, an indication that the way in which users typed a key gave a significant amount of information about how they typed the same key at different locations within the typing sample. Meanwhile for the tests run on feature-pairs containing *dissimilar* features, we rejected the null hypothesis (of independence) in over 40% of the cases, an indication that even some of the *dissimilar* features exhibited dependencies. Table 4.1 captures the inter-feature dependencies in terms of conditional probabilities of the form $P(f_1 < \alpha \mid f_2 < \beta)$.

Table 4.1: Conditional probabilities between KITs of selected digraphs.

Event	Probability
P($KIT_{AD} < 100 \mid KIT_{H-} < 100$)	0.50
P($KIT_{AD} < 100 \mid KIT_{AT} < 100$)	0.51
P($KIT_{AD} < 100 \mid KIT_{DU} < 100$)	0.51
P($KIT_{UN} < 150 \mid KIT_{RA} < 150$)	0.59
P($KIT_{-U} < 150 \mid KIT_{RA} < 150$)	0.30
P($KIT_{CH} < 150 \mid KIT_{RA} < 150$)	0.98

The features (f_1 and f_2), and thresholds (α and β) used in the table are chosen arbitrarily to give an example of how the extent of dependency between certain features could (or could not) aid statistical inferences about the features. The "-"

represents the space character. Take the case of digraph RA for instance. A user who typed the digraph RA in under 150ms was very likely to type digraphs CH in under 150ms, very unlikely to type digraph $-U$ in under 150ms, and moderately likely to type digraphs UN in under 150ms. This means that an adversary who knew how users were likely to type the digraph RA in our typing sample could (potentially) have lowered the search space for the digraphs CH, $U-$ and UN during an attack against a randomly selected user. Not all inter-feature probabilities (for different thresholds α,β) were that interesting however. For instance, a user who typed digraphs *DU*, *AT* or *H-* in under 100 ms had almost equal likelihood of typing digraph AD in over 100ms as in under 100 ms. In this case an adversary seeking to determine if a random user typed digraph AD in under 100 ms could not benefit much from the knowledge that the victim (or a typical user) typed the digraphs *DU*, *AT* or *H-* in under 100 ms.

Impact of Observation #3 on Keystroke Research: While biometric features are typically associated with dependencies and correlations [64], no previous work has investigated the extent of these dependencies in keystroke dynamics to the best of our knowledge. Our findings thus represent the first empirical evaluation of the dependencies exhibited by keystroke features, and should influence:— 1) Analytic work such as [5] in which assumptions regarding keystroke feature dependencies are used to aid investigations into various aspects of keystroke dynamics; and 2) The design of statistical attacks that build off of the inter-feature dependencies to break keystroke systems. The latter direction of research should in turn motivate work on defences against these types of attacks before they take root in real systems. Our attacks in this work actually also exploit the feature dependencies, as we use the conditional probabilities between features to make decisions on how to traverse the search space.

CHAPTER 5

THE KEYSTROKE STATISTICAL ATTACKS

In this section we discuss the underlying assumptions, implementation details and performance of the statistical attack on the fixed-text keystroke authentication system.

5.1 Assumptions and Attack Scenarios

Assumption #1: We assume that the adversary knows the victim's password and has access to keystroke forging software. We discussed these two issues and provided accompanying evidence during our discussion in Chapter 1. We thus do not re-emphasize them here. Perhaps the only point we have to add is that although different keystroke verifiers may be based on different features, the attacker does not have to know about these features, since the verification system will automatically parse the bot-injected samples for the right features in the same way it would for a human typist.

Assumption #2: We assume that the attacker will be able to access large amounts of keystroke data so as to extract the keystroke feature statistics needed to design the attack. One obvious option available to the adversary is to use accomplices to provide biometrics samples for the password in question. Ballard *et al.* [65] also cite this data collection option in their paper on synthetic attacks against handwritten signatures. Other possibilities include crowdsourcing with the aid of fake keylogging Web sites, directly extracting feature statistics from publicly accessible keystroke datasets. or even fooling unsuspecting users at a public Cyber Cafe so as to have them type a common pass code (matching the victim's authentication data) for access while a keylogger captures keystroke sequences.

Attack Scenarios: Our main attack scenario is the case of an adversary who uses a personal machine to attack other users via the Internet. If for instance Bob wants to launch an attack against a keystroke-protected Facebook account owned by Alice, Bob only has to make authentication attempts at the Facebook server using attack-software installed on his own machine. The attack is thus not affected by any host-based defences (such as the one in [66]) deployed at Alice's machine since the injection of synthetic keys is done at the attacker's own machine.

In high security applications such as online banking, the server may, in addition to Alice's password and keystroke signature also authenticate the IP address used by Alice to make her transactions. This means that Bob, seated at his own compromised computer may not be able to make successful authentication attempts against Alice's account. Our attack may hence only work in that case if Bob can have physical access to Alice's machine so as to compromise any defences and (or) install and launch the keystroke forging software. This attack scenario will be *much less likely* than the first, but cant be ruled out given a committed adversary.

5.2 The Attack Algorithm

Algorithm 1 shows how our attack extracts features from a user's password, and how it uses population data to generate feature values to be used for attack. For each distinct KHT and KIT in the victim's password, the algorithm bins the associated latencies over the population using a bin size defined by the value of the *binSize* parameter, and then returns the centers of the *h* highest frequency bins.

ALGORITHM 1: Generating feature outputs to be used for tree traversal

```
Input:  User's password string //E.g., ABAB;
Input:  KD population data corresponding to password string
Output: Matrix of feature outputs for each feature
KHTs [ ]← Distinct characters in password //E.g., KHTs=[A B] for string ABAB;
KITs [ ]← Distinct digraphs in password //E.g., KITs=[AB BA] for string ABAB;
Features [ ]←[KHTs    KITs ] // E.g., Features=[A B AB BA];
NumberOfFeatures←NumberOf(Features)// NumberOfFeatures=4 for string ABAB;
for i← 1 to NumberOfFeatures//Assume lowest array index is 1 for simplicity;
do
    BinnedFeature[i]←Binning(Features[i], binSize);
    //Use a bin size of binSize to bin the latencies of Features[i]. Return
    bins sorted in descending order of probability;
    for j← 1 to h do
        F[i,j]←SelectDominantBins(BinnedFeature[i], h);
        //Each pass through inner loop assigns center of $j^{th}$ bin of
        Feature[i] to the array location F[i,j];
        //E.g., For KHT of A (see Figure 5.1), we shall have $F[1,1] = A : V_1$,
        $F[1,2] = A : V_2$, $F[1,3] = A : V_3$;
Return F //Matrix containing a total of h outputs for each feature;
```

Take the password string *ABAB* for instance. The KHTs of *A* and *B* and the KITs of *AB* and *BA* are respectively the distinct KHTs and KITs extracted from this password string. Across the population, the algorithm extracts and bins the latencies corresponding to each of these four features, and then determines the centers of the *h* highest frequency bins in each of the four cases. We heuristically set the *binSize* parameter as 16 *ms* since this value gave us good results during our experiments. The function *Binning*() performs the binning process, while the function *SelectDominantBins*() returns the centers of the most frequent bins. We set the value of h as 3 in this work, implying that Algorithm 1 (or the function *SelectDominantBins*() in particular) returns three bin centers for each of the four features.

For a feature such as the KHT of A, we use the notation $A : V_1$, $A : V_2$ and $A : V_3$ to refer to the 3 bin centers returned by Algorithm 1. The first (i.e., $A : V_1$) corresponds to the highest frequency bin while the last (i.e., $A : V_3$) corresponds to the lowest frequency bin. During the search process, these three values per feature will account for only a small portion of the search space, but should be sufficient to illustrate the power of the attack.

Note that since the binning is explicitly built off of empirical data, rather than off of parametric density functions (say, under the Gaussian assumption), the highest probability bin will not necessarily be centered at the population mean. Additionally, for features having skewed or multi-modal distributions, a sequence of high probability bins may be located at the same side of the population mean, which would also not be the case under the Gaussian assumption.

We formulate the feature-space enumeration process as a tree traversal in which each tree level represents a feature. Figure 5.1 illustrates the structure of this tree for the hypothetical password $ABAB$. The figure is motivated by the attack tree used for the analysis of Biometrics Cryptographic Key Generators (BKGs) in [64],

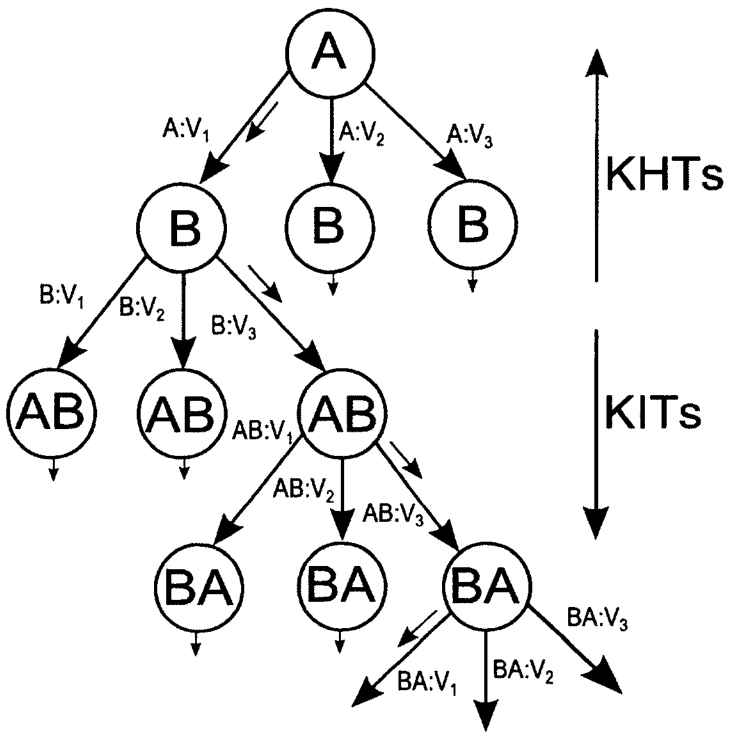


Figure 5.1: Enumerating the keystroke feature space for the hypothetical password *ABAB*.

although we use a completely different algorithmic framework tuned to exploit our observed statistical traits.

The enumeration process begins by setting the output value of the feature located at the root of the tree. Before delving into the criteria for determining which feature to be located at the root of the tree, lets assume that this feature is the KHT of A, for the case of the hypothetical password used for our illustrations. Of the three possible values that can be assumed by the KHT of A, the algorithm sets the largest (i.e., $A : V_1$, corresponding to the most frequent bin) as the output of feature A at this stage. The next output to be set is that of the KHT of B, located at the level just below the root node. Of the three possible values, $B : V_1$, $B : V_2$ and $B : V_3$ that can be assumed by the KHT of B, the algorithm then selects the one which maximizes the conditional probability, $P(B = B : V_n \mid A = A : V_1)$, for $n = 1, 2, 3$. This process continues down the tree, with the output at each tree level being based on the conditional probabilities between the possible outputs at the particular tree level, and the current output at the level just above.

The first *guessing attempt* is realized when the enumeration process first reaches the leaves of the tree. We define a *guessing attempt* as a set whose cardinality equals the number of levels in the feature enumeration tree, with every element in the set being an output of a feature from a different level of the tree. For our hypothetical example, the first guessing attempt could for instance take up the values in the set $G_1 = (A : V_1, B : V_3, AB : V_3, BA : V_1)$, where the four elements of the set respectively correspond to the outputs of the KHT of A, KHT of B, KIT of AB and KIT of BA. These four elements of G_1 are represented on the graph by the path indicated by the sequence of short arrows.

Each subsequent guessing attempt follows by modifying the output of a single feature in the current guessing attempt. These feature output modifications start from the features located at the leaves, and recurse up the tree, using conditional probabilities to guide decision making in the way already described.

For instance, assuming $P(BA = BA : V_2 \mid AB = AB : V_3) > P(BA = BA : V_3 \mid AB = AB : V_3)$, the second guessing attempt will be $G_2 = (A : V_1, B : V_3, AB : V_3, BA : V_2)$, while the third guessing attempt will be $G_3 = (A : V_1, B : V_3, AB : V_3, BA : V_3)$. Meanwhile, the next three guessing attempts will see the output of feature AB modified to a new value, and the output of feature BA again iterated through its three possible values in accordance with the earlier described conditional probability-based criteria.

Observe that between the first and third guessing attempts (G_1 through G_3), the first three feature outputs (i.e., $A : V_1, B : V_3, AB : V_3$) are unchanged, while the lowest feature (KIT of BA) sees three different outputs. A direct consequence of tree traversal techniques such as the one we employ, this trait means that a wrong output for the KHT of A, or KHT of B, or KIT of AB, will have a negative impact on all guesses G_1 through G_3, while a wrong output for the KIT of BA will only impact the individual associated guess. This problem generalists to all other guessing attempts, and is such that a wrong output for a feature located close to the root will result in a greater amount of fruitless feature space enumeration than a wrong output for a feature located at the leaves of the tree.

Since authentication systems impose limits on the number of permitted false authentication attempts, its crucial that the tree design minimizes the extent of fruitless search space enumeration. Our tree exploits information on the discriminability of features to handle this problem. Specifically, we ensure that the weaker (less discriminative) features are located towards the root of the tree while the powerful (more discriminative) features are located closer to the leaves of the tree. For two features f_1 and f_2, we locate the feature f_1 above the feature f_2 in the tree if the mean Battacharrya distance of f_1 across the population is greater than that of f_2. The method used to compute the mean Battacharrya distance of a feature was described in Section 4.2.

5.3 Attack Samples

The choice of string lengths used for the attacks was based on two factors. First, the summary statistics in [6], indicate that past fixed text keystroke studies have for the most part used strings ranging from 6 to 17 characters in length. Because we needed our results to be easily put in the context of past findings, we decided to attack this same range of string lengths. The second consideration behind our choice of string lengths was the need to attack strings whose lengths are representative of the password lengths being used today, as this would give a good reflection of the performance of a password-KD system under a statistical attack in the current Internet setting. With regard to this factor, we studied various recently hacked password lists from which we observed average password lengths of about 6.62 to 7.88 characters [20]. We took these average password lengths as some sort of password-length lower bound and attacked substrings of length 7, 9, 11, 13, 15, 17 and 20. This large number of password strings will help give a concrete view of how statistical attacks may perform over a wide range of string lengths.

5.4 Keystroke Verifiers

To evaluate the success of the synthetic attack, we used the Z-score [6], Scaled Manhattan [6] and Naïve Bayes [5] verification algorithms. The first two were among the best performers in a study that compared up to 14 different fixed text KD verifiers [6], while the third, despite not being part of the algorithms compared in [6], is very popular in machine learning literature, readily available in the Weka machine learning tool [67], and was recently shown in [5] to perform very well for the kind of fixed text heterogenous feature vectors used in this work. We implement the first two verifiers from scratch and use the Weka implementation of the Naïve Bayes classifier. A brief description of each of the three verifiers follows:

5.4.1 Z-score Classifier

Given feature vectors extracted from a user's keystroke data for a given string over several typing runs, this verifier computes the mean and standard deviation of each feature during the training phase. In the test phase, the absolute Z-score between each feature of the test vector, and the corresponding feature in the mean vector (created during the training phase) is computed. The anomaly score is a count of how many z-scores exceed 1.96. If a_i is the test value, and b_i and s_i are the mean and standard deviation of feature i as seen during training, the z-score is computed as $z = (|a_i - b_i|/s_i)$.

5.4.2 Scaled Manhattan Classifier

This verifier uses a Manhattan-distance computation in which each dimension is scaled by the average absolute deviation seen for each feature during the training phase. In the training phase, the mean and the mean absolute deviation for each of the features are computed. In the test phase, the anomaly score is computed as $\Sigma_{i=1}^{p}|a_i - b_i|/y_i$ where a_i and b_i carry the same meaning as in the z score classifier, and y_i is the average absolute deviation of feature i during training.

5.4.3 Naïve Bayes Classifier

During training, the verifier builds a user's model using maximum likelihood estimation. In the test phase, it makes a classification decision basing on the probability that a given set of latencies belongs to the user. Due to space limitations and the fact that the Naïve Bayes verifier is well studied in literature, the reader is referred to [44] for details on its mechanism and underlying assumptions.

5.5 Performance of the Attack

5.5.1 Overview:

To launch the attacks, we used templates from 110 users who participated in at least 3 of the data collection phases. For each of the 110 users, we used data from the first phase of our experiments for training[1], and then used 30 instances of the user's data from the other typing phases to attack the user's model so as to generate the user's genuine scores. To generate the impostor scores, we launched impostor attempts in 2 different ways. In the first approach, we used 50 randomly selected impostors from a pool of 1000 users to attack the template (or model) built for each user during training. Throughout the rest of the section, we shall refer to this attack as a *zero-effort attack* [9], since it simulates an impostor who makes no effort to imitate the genuine user's way of typing. In the second approach, we used 50 of the top 1000 guesses generated by our attack algorithm to conduct impostor

[1]Unlike the first two classifiers, the Naïve Bayes classifier requires instances of both the positive and negative classes during training. For each participant we used as many instances of the negative class as the participant had for the positive class during the first phase of our experiments.

attacks against each user. We shall interchangeably use the terms *algorithmic attack* and *statistical attack* to refer to this form of attack.

The performance analysis in this section will focus on the comparison between these two attacks, since our principal aim is to illustrate how our *algorithmic attack* compares to the well known *zero-effort attack* which is still the benchmark for the performance evaluation of keystroke dynamics systems. Central to our performance evaluation methodology is the Equal Error Rate (EER), a measure which represents the point at which a verifier's false-reject and false-accept errors are equal. The EER is often used to evaluate biometrics system performance [6, 9], and is such that a low EER is synonymous with good performance, while a high EER implies poor performance of a system. Some researchers prefer to express EER values on a scale running from 0 to 100 [7]), while others use a scale running from 0 to 1 [6]. In this work, we adopt the latter convention.

To calculate a user's EER for a given type of attack, we construct a Detection Error Tradeoff (DET) curve for the user, from which we determine the EER as the point at which the curve meets the line y=x (i.e., point at False Reject Rate (FRR) equals the False Acceptance Rate (FAR)). Figure 5.2 illustrates this procedure for a certain user for both the *zero-effort* and *algorithmic* attacks. The profile under attack was built for a 7-character string, and the algorithm used for verification was the Naïve Bayes algorithm. As indicated by the figure, this particular user's performance was negatively impacted by the *algorithmic attack*, given the high EER for this attack relative to the *zero-effort attack*. For each of the seven string lengths used in this study, we carry out this procedure for all 110 users and 3 verifiers, and eventually compute the mean EER and standard deviation of EERs over the population for the attacks against each string length and verifier.

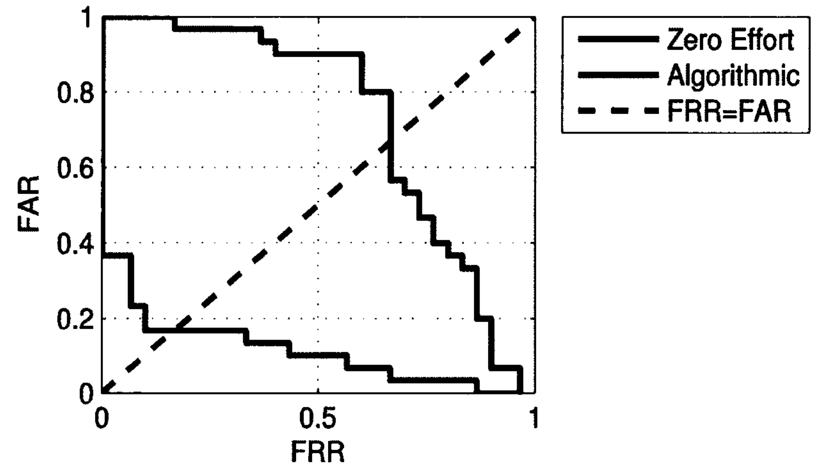


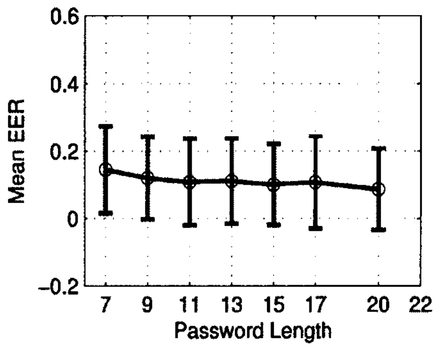
Figure 5.2: DET curves computed for the zero-effort and algorithmic attacks launched against one of the users in our experiments.

5.5.2 Global Impact of the Attack:

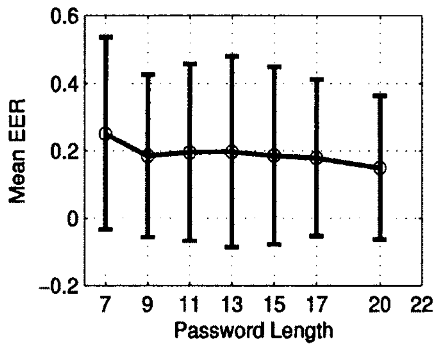
Below, we describe the major attributes of the of the attacks with regard to the general behavior seen across the population.

1. *Variance in EERs across the population:*— For all verifiers and string lengths, the algorithmic attack always caused a much higher variance in the average EERs than the zero-effort attack (see Figure 5.3)[2]. Because reliable systems are typically designed to have low variance in their performance metrics [68], this increment in the variance of EERs, irrespective of whether the mean EERs were affected or not, is the first indicator of why the algorithmic attack is a major threat relative to the zero-effort attack. Further investigations into the cause of this high variance (results not shown due to space limitations) revealed

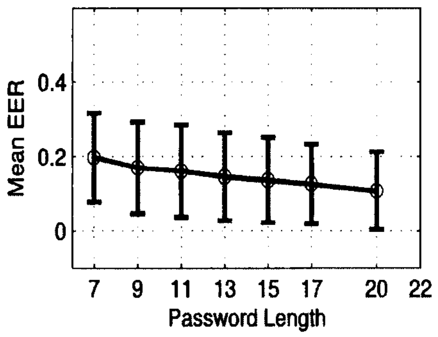
[2]The error-bars indicate a single standard deviation from above and below the mean EER. Also note that the EERs plotted on the graph have only been computed for password lengths of 7, 9, 11, 13, 15, 17 and 20 characters. The curve joining these points is only meant to show the general trend.



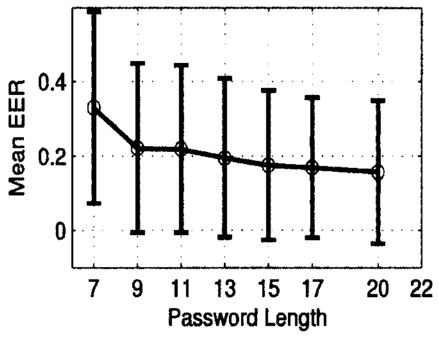
(a) Naïve Bayes classifier under zero-effort attack



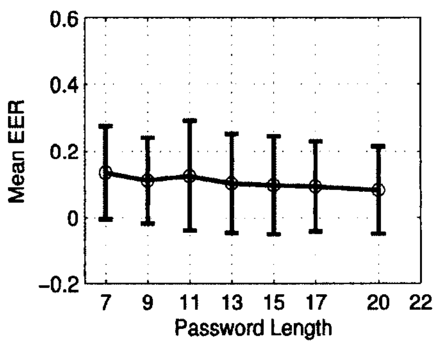
(b) Naïve Bayes classifier under algorithmic attack



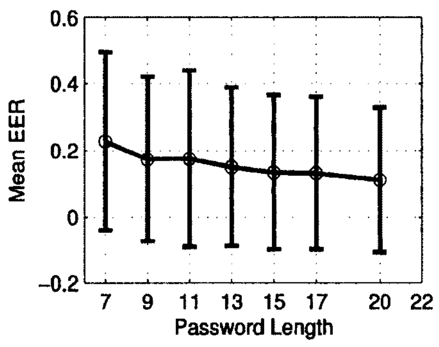
(c) Z-Score classifier under zero-effort attack



(d) Z-Score classifier under algorithmic attack



(e) Scaled Manhattan classifier under zero-effort attack



(f) Scaled Manhattan classifier under algorithmic attack

Figure 5.3: Mean performance of the Naïve Bayes, Z-score and Scaled Manhattan classifiers under the algorithmic and zero-effort attacks.

that under the algorithmic attack, there was a small proportion of the user-population which saw almost zero EERs and another small proportion which saw EERs almost close to 1. This kind of extreme behavior was not as pronounced under the zero-effort attack, and naturally accounted for the higher variance under the algorithmic attack. Regarding the cause of this high disparity in user behavior under the algorithmic attack, the population statistics-based nature of the attack indicates that the well performing users had their typing traits distinct from those of the general population, while the weak users' group had characteristics very similar to those of the population.

2. *Mean EERs across the population:*— For all verifiers and string lengths, the algorithmic attack caused higher mean EERs than the zero-effort attack. The long error bars associated with the algorithmic attack cause us to use a wide scale (that unfortunately seems to dim the clarity of these EER changes), however, Table 5.1 captures this behavior so well as it indicates that the EER increments ranged from 0.11 for the 7-character string and the Naïve Bayes verifier, to 0.03 for the 20-character string and the Scaled Manhattan classifier. While it may be tempting to write off these EER increments as trivial, they are quite high, as they constitute a large percentage of the EERs seen under the zero-effort attack. Observe for instance that the Naïve Bayes verifier sees increments of over 50% of the zero-effort EER, while the other two verifiers see increments of over 30% of the zero-effort EERs for all string lengths. As a final note on why these EER increments should represent a major threat in the sense of a (keystroke) biometrics system, the reader is referred to [6], where an EER

difference of just 0.075 separated the top ten verifiers in a study in which several verifiers were compared.

Table 5.1: EER increments caused by the algorithmic attack.

String Size	**KD Verification Algorithm**					
	Naïve Bayes		**Z-score**		**Scaled Manhattan**	
	Increase in mean EER	% Increase in mean EER	Increase in mean EER	% Increase in mean EER	Increase in mean EER	% Increase in mean EER
7	0.105	72.5	0.131	67.1	0.091	67.8
9	0.061	53.5	0.062	30.1	0.063	54.3
11	0.093	80.4	0.063	35.9	0.052	39.8
13	0.081	76.7	0.054	33.8	0.045	45.6
15	0.076	84.4	0.038	28.6	0.041	37.7
17	0.068	66.7	0.037	33.1	0.039	40.2
20	0.062	73.2	0.048	44.5	0.033	35.5

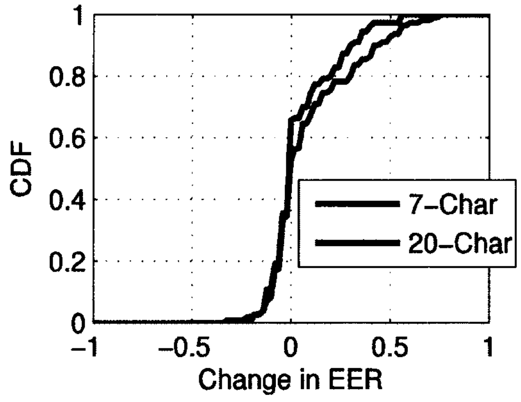
3. *Impact of string length:*— As the string lengths increased, the increments in mean EER caused by the algorithmic attack for the most part saw a monotonic decrement (save for 2 cases). The observed reduction in the impact of the attack suggests that free text keystroke systems, by virtue of using long blocks of text for verification could see much lower, or even no increment in EER, under the kind of algorithmic attacks implemented in this paper.

5.5.3 Effect of the Attack on the Performance of Individual Users:

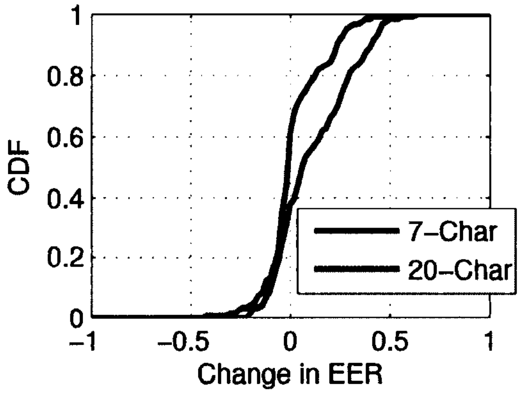
Having compared the mean system performance under the two attacks, we proceeded to investigate the extent to which the algorithmic attack improved or worsened each user's performance relative to the zero-effort attack. For this analysis, we subtracted each users EER under the zero-effort attack from that under the algorithmic attack and plotted a CDF of these differences.

Figure 5.4 shows the full distribution of the EER changes for the 7 and 20 character strings over the population of users subjected to the two attacks. We focus on the two extreme string lengths because they reflect and (or) bound the general behavior exhibited across all string lengths. Highlights from this figure are discussed below:

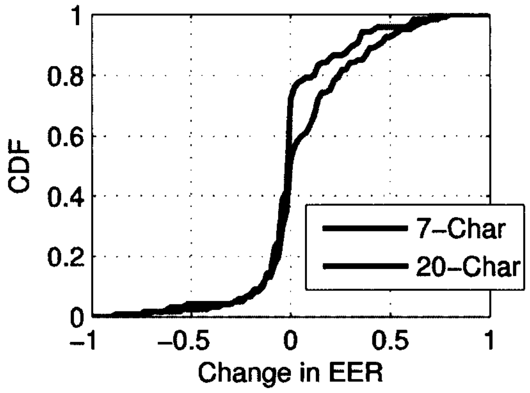
1. *EER Variations of Individual Users:*—The EERs of different users were impacted markedly differently by the attacks. While some users saw performance improvement (reduced EERs) under the algorithmic attack, others saw considerably large increments in their EERs. For instance, under the Naïve Bayes verifier (Figure 5.4a), about 40% of the population saw improved EERs (EER differences less than zero) under the algorithmic attack for both the 7 and 20 character strings, while about another 20% for the 7 character string, and over 15% for the 20-character string saw no change at all in their EERs. Meanwhile, about 5% of the population saw EER increments greater than 0.5 for the 7 character string, while an even smaller number saw a similar increment for the 20-character string. This variation in users' behavior is seen across all verifiers and string lengths used for our study, and again points to the earlier mentioned trait of certain user clusters being very similar to the population, while others are very dissimilar to it. Additionally, the fact that a small group of users seem to be responsible for the increment in mean system EER suggests that a targeted solution for the small group of poor users could be employed to control the attack.



(a) CDF of the changes in EER caused by the algorithmic attack against the Naïve Bayes verifier for each of the 110 users



(b) CDF of the changes in EER caused by the algorithmic attack against the Z-score verifier for each of the 110 users



(c) CDF of the changes in EER caused by the algorithmic attack against the Scaled Manhattan verifier for each of the 110 users

Figure 5.4: Effect of the algorithmic attack relative to the zero-effort attack for individual users.

2. *Impact of String Length:*—Figure 5.4 shows that the proportion of users whose change in EER exceeded zero was higher for the 7 character string than for the 20 character string across all verifiers. This observation explains the monotonic decrease in mean EER with increased string length that we highlighted earlier in the discussion since the small proportion of users with increased EERs for the long strings should naturally result into lower mean EERs for such strings.

CHAPTER 6

ROBOTIC IMITATIONS OF TOUCH GESTURES

6.1 Overview

Touch-based authentication—now widely studied for its potential to serve as a second layer of defense to the PIN lock mechanism on mobile devices—has traditionally been evaluated under the assumption of naïve (zero-effort) adversaries. The zero-effort threat model, although well understood not to be representative of the state-of-the-art threats [69], is for several reasons fronted by researchers as being able to sufficiently capture the threat that a touch-based authentication system would face in practice. For instance in one of the recent papers on touch biometrics, Frank *et al.* [10] make the following arguments to rule out the need for stringent penetration testing of their system:

.... we can hardly imagine someone learning the touch behavior of 30 features, such as pressure, distribution of acceleration, etc., just by looking over the shoulder [10].

...A more successful but more involved attack would be to place a malware application on the user's device. This malware could learn and report the touch pattern if the details of how to compute the features are known to

the attacker... However, we argue that a user with malware on the device has already lost the race against the attacker [10].

These arguments—echoed in many other papers—are sound without doubt. Notably though, the notion that these two attacks (malware and a form of shoulder surfing) represent the full spectrum of threats that the system could face is for several reasons debatable. In this chapter we demonstrate that: 1) a simple robotic/mechanical device (as opposed to malware) can very effectively degrade the performance of a touch-based authentication system, and, 2) publicly accessible touch biometrics data (such as the data at [70]) can be leveraged to drive the attacks even where detailed information about the intended victim's swiping behavior is not available.

The chapter covers our data collection experiments, attack design and results of the robotic attack.

6.2 Data and Features Used for our Investigations

6.2.1 Data Collection Process

We conducted two data collection experiments using two Android applications that captured the way in which users touched the mobile phone screen. The gestures that users typically perform on a touch screen include zooming (in and out), clicking (tapping), swiping to switch between screens (i.e., horizontal swiping) and swiping to move a page up and down (vertical swiping). The tap gesture does not hold enough information to strongly separate between a large group of users [10], while the zoom gesture does not occur frequently enough to guarantee that a continuous authentication application will always have enough data to make classification decisions [10]. For these reasons, most work on continuous touch-based authentication hinges around the two swipe gestures[1]. We focus on these two gestures in this work.

For each of a set of points on a touch stroke registered on the screen during swiping, the applications recorded the: 1) x and y coordinates, 2) time at which the finger touched the point in question, 3) area occluded between the finger and the screen, 4) pressure exerted on the screen and 5) orientation of the phone (portrait or landscape). The two Android applications basically simulated how users read text and view images on the touch screen. Based on a short paragraph of text or an image, users had to answer several questions by selecting one of two to four alternative answers that we provided per question. On reading a question, each user would scroll/swipe back to the image or block of text containing clues to the solution, before scrolling/swiping towards the answer section where the user would select one of the choices provided.

Both applications were based on the same idea, although each application was based on a different set of questions/images. In the first phase of experiments (Session I), users interacted with one application. They then returned on another day at their convenience to interact with the second application (i.e., during Session II). All participants used the same brand of phone—the Google Nexus S running Android version 4.0—so as to avoid bias in our findings that might be caused by differences in the way in which different phones extract information from touch gestures.

[1]Li *et al.* [11] used the tap gesture in conjunction with the two swipe gestures. However they found that it had very poor discriminability.

6.2.2 Feature Extraction and Preprocessing

Before extracting features from the data, we performed an outlier filtering step to eliminate very short strokes since these likely originated from click events (or taps), as opposed to swiping (scrolling). Frank *et al.* [10] performed a similar step on users' strokes before proceeding with the classification process. Having removed outliers, we extracted 28 features from each stroke. There is currently no universal feature-set that researchers use to represent a distinct stroke. For example, Frank *et al.* [10] defined 30 features and discarded 3 of them after feature analysis, Li *et al.* [11] defined 13 features (or 14 features if the x and y coordinates of a point are considered as distinct features) and discarded 4 of them after feature analysis while Feng *et al.* [12] used 53 features. For this work we used 28 features that we believe best summarize the statistical attributes of a touch stroke. A description of how we computed these features follows:

Using the pressure and area readings at different points along a stroke, we respectively built a pressure vector, P, and an area vector, A, to represent the pressure and area associated with the stroke. We computed the velocity between every pair of consecutive points along a stroke, and used these values to generate the velocity vector, V. Finally, for every pair of points in V, we computed the acceleration, and generated an acceleration vector, A'.

For each of the four vectors A, P, V and A', we computed five measures to summarize a user's mean behavior, variability in behavior and extreme behavior along a stroke. These were: 1) lower quartile, 2) second quartile, 3) third quartile, 4) mean, and, 5) standard deviation. This gave a total of 20 (=4$\times$ 5) features per stroke. The last 8 features making up a vector representing a stroke were: the x and y coordinates of the starting points, the x and y coordinates of the end points, the distance between the end and starting points of a stroke, the time taken to complete the stroke, the tangent of the angle between the line joining the end-points of a stroke and the horizontal, and the sum of distances between every pair of adjacent points on a stroke.

6.3 Attack Design

6.3.1 General Assumptions

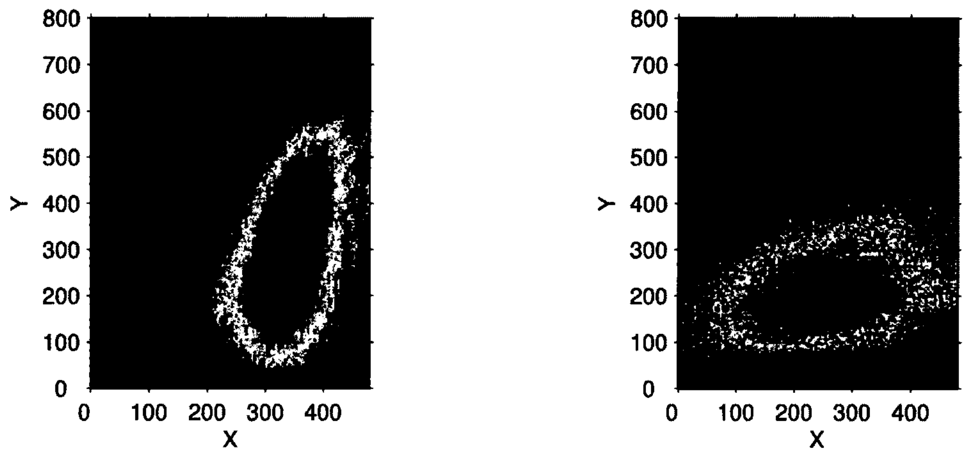
We assume an adversary who gets physical access to a phone for which touch-based continuous authentication is the only active layer of defence. In practice, this scenario may arise for an attacker who : 1) breaks the PIN lock mechanism (*e.g.*, using methods such as those in [25, 26]), or, 2) finds a phone in which the PIN lock has been disabled temporarily (*e.g.*, a user who sets a very long timeout for the PIN lock), or, 3) finds a phone in which the PIN lock has been completely disabled by the owner [10]. To be able to determine the amount of extra security that touch-based continuous authentication adds to the standard PIN lock in the worst case, we believe that these assumption must necessarily be made. Also see [71], for an investigation in which a similar assumption (i.e., that the adversary has access to the victim's password) was made in order to enable rigorous evaluation of the security of Randomized Biometric Templates (RBTs).

In the attack itself, the attacker will seek to view private information on the phone (*e.g.*, emails, pictures, *etc.*,) without triggering the anomaly detection mechanism. The attack thus basically proceeds by scrolling/swiping through documents on the phone. In practice we believe that the attacker could even assist the robot during certain operations (*e.g.*, occasionally clicking at a challenging location), since the anomaly detector will most likely not be sensitive enough to detect a few anomalous clicks. Next we discuss the underlying statistical observations that drive the attack, and the details of the mechanical and algorithmic design of the robot.

6.3.2 How do People Swipe on the Phone?

To design the attacks, we first examine the way in which people swipe in general. How random is swiping behavior across a population ? Are there certain distinct traits that manifest frequently across a large number of users? This section provides answers to these and related questions. Due to space limitations, we only present results on the pressure exerted on the screen, the area between the finger and the screen and the region of the phone at which most swiping is done. Other measures such as the time interval between consecutive swipes, the velocity of the finger and the length of a stroke, to mention but a few, are left out here but will be used in the attack design.

Location of Swiping Activity: Figure D.12 shows the density of touch strokes captured at different positions of the phone screen during the first phase of experiments. The dark blue color corresponds to regions which saw very little or no swiping/scrolling activity while a high intensity of red corresponds to regions which



(a) Spatial distribution of swiping activity during vertical swiping.

(b) Spatial distribution of swiping activity during horizontal swiping.

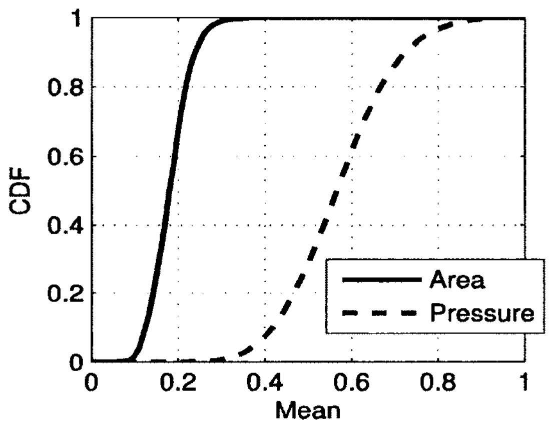
Figure 6.1: Color map showing the spatial distribution of touch strokes on the phone screen.

saw a lot of swiping. The phone was being used in portrait mode when the strokes were generated. Note that the coordinate system used on the figures is different from that used by the Android system. Observe (Figure D.12a) that the vast majority of vertical strokes generated by our user population originated from points having X values in the neighborhood of 300 units, and terminated at a position with an X value of close to 400 units (and vice versa). Notably, this region of high activity comprises less than 50% of the screen display. The heart of the red region (which tends towards black) occupies an even a much smaller portion of the screen. Similar traits (see Figure D.12b) were seen with the horizontal swiping.

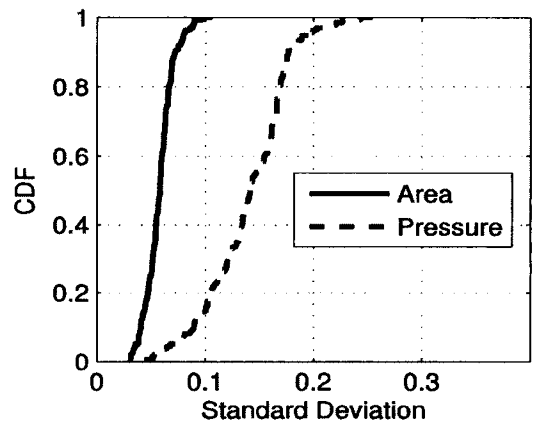
Based on evidence provided through this plot, an adversary with access to general population statistics could potentially significantly narrow down the scope of features such as: 1) the x coordinate of the start point of a stroke, 2) the y coordinate of the start point of a stroke, 3) the x coordinate of the end point of a stroke, 3) the y coordinate of the end point of a stroke, and, 4) the direction of the end-to end line, among other features. These features represent a good proportion of the features used to characterize users' touch gestures in past research (such as in [10, 11]), and will also be used in this study.

Regarding the cause of the clustering tendency, our conjecture is that the high density of strokes on the right side of the screen (i.e., taking the case of vertical swiping for instance) was likely because the majority of users are right handed, tending to hold the phone in the right hand and swiping with the thumb, or holding the phone in the left hand and swiping using one of the fingers on the right hand. In any of these two scenarios, a user is very likely to swipe in the manner reflected in the figure. We do not rule out the possibility that certain highly specialized applications could depict variations from the pattern shown in the figure. In this case we argue that a committed attacker who has interest in breaking into such an application could easily make research on the swiping trends for such an application.

Finger Area and Pressure on the Screen: Figure 6.2a shows the distribution of the mean area touched by the finger and the mean pressure exerted on the screen across a subset of our full user population. To plot the figures, we computed each user's mean area (and mean pressure) and plotted the results on the CDF. Observe that over 80% of the population had a mean area of between 0.1 and 0.25 and that about 50% of the population had mean pressure values of between 0.4 and 0.6. These user proportions already suggest that a large number of users could be clustered around a narrow band of values (for both pressure and area). To get a more concrete insight into the possible clustering of users' profiles, we studied the variability seen by users for each of these two variables. Particularly, we computed the standard deviation of the mean area and mean pressure exhibited by each of the users represented in Figure 6.2a, and then plotted these values on a CDF (Figure 6.2b).



(a) Distribution of mean pressure and area across the population.



(b) Distribution of standard deviation of pressure and area across the population.

Figure 6.2: CDFs expressing the mean and variability of area and pressure across the population.

Taking the case of pressure for instance, the figure shows that about 40% of the population had a standard deviation of over 0.15. Assuming users' pressure values follow a Gaussian distribution, a user with a standard deviation of 0.15 could see her/his biometric pattern fall on a band having a width of up to 0.6 units (i.e., 2 standard deviations on either side of the mean). Given such a wide span, an input selected from the earlier mentioned clustered regions (Figure 6.2a) could have a good chance of falling within such a user's feature range.

Similar observations made for the other features (*e.g.*, velocity, length of strokes, start point of stroke, *etc.*) further prompted us to believe that generic information from the population could possibly enable us to implement a lethal attack on a touch-based authentication system.

6.3.3 Mechanical and Algorithmic Design of Robot

Fabrication of the "Finger": We had three main design considerations regarding the object to be used to touch the screen. These were: 1) the object had to be able to register touch events on the capacitative screen, 2) it had to easily match the finger surface area as needed, 3) it had to be soft to avoid damaging the screen, and 4) it had to be made from cheap, domestically accessible materials. The fourth point rules out technologies such as prosthetics [72] that despite guaranteeing artificial fingers that match many of the properties of a human finger, would make the attack implementation expensive, and likely defeat our aim of demonstrating how easily the attack can be launched based on materials that are cheaply available off the shelf.

To address all four points we fabricated the finger surface from play-doh [73], a malleable compound that children use to model different kinds of play-objects. Although play-doh on its own was (to our surprise) able to register touch points on the screen, we housed it inside a touch screen glove (see [74]) in order to have more close control of the "finger" area touching the screen. The small Play-doh lump was fastened to the extreme end of a blunted steel nail to ensure firm contact between the "finger" and the phone screen.

In all experiments we set our "finger" area to be approximately 0.15 units, which was the mean value we observed across our user population. The area setting itself was manual—we iteratively molded the play-doh and tested it on the phone (during preliminary experiments) until the area value stabilized at around 0.15 units. When the motors pushed the play-doh against the touch screen during scrolling, the play-doh, owing to its softness, would see some amount of deformation. These small variations in play-doh touch surface area did not affect our attack that much, since human fingers also see variations in area along the path of a stroke.

Perhaps one interesting observation worth noting here is that when we connected a battery (AA type) to the play-doh during the attack, the area registered on the phone screen increased. A possible reason for this is that the effective area of contact between the screen and the phone is not only dependent on the physical area of contact between the two, but, it also depends on the extent of electrical contact between them. We leveraged this property to introduce changes in the "finger" area during the experiments.

Robot Components and Attack Algorithm: The basic idea behind our attack is for the finger to stroke the screen in such a way that matches the average user's behavior. Based on population statistics therefore, the finger has to be set to: 1) move at a certain average speed, 2) move in some general direction at a certain region of the phone, and 3) exert a certain average pressure on the phone, to match the average user. Compared to some of the tasks that Lego robots have been programmed to do in the literature (see [75] for an example), designing a Lego to attain the above four targets is obviously a much more straightforward problem. We stress that the focus of our work is not to push at the boundaries of Lego design, but rather to illustrate that a very cheap robot running a very simple algorithm (that could easily be implemented by a novice attacker) is a much more rigorous penetration testing tool for touch-based authentication systems than the current state-of-the-art methods. While we take some steps to minimize the likelihood that the attack could be very easily thwarted, the question of the most sophisticated robotic design that could be used for this kind of attack is not of interest in this work. In fact, depending on the value of the resource to be retrieved from the phone, the attacker could potentially use a more sophisticated robot (*e.g.*, the NAO [76]) and hence a different design philosophy. We leave all such investigations for future research after highlighting the impact of a robotic attack based on the very bare minimum resources. Figure 6.3 depicts the robot design.

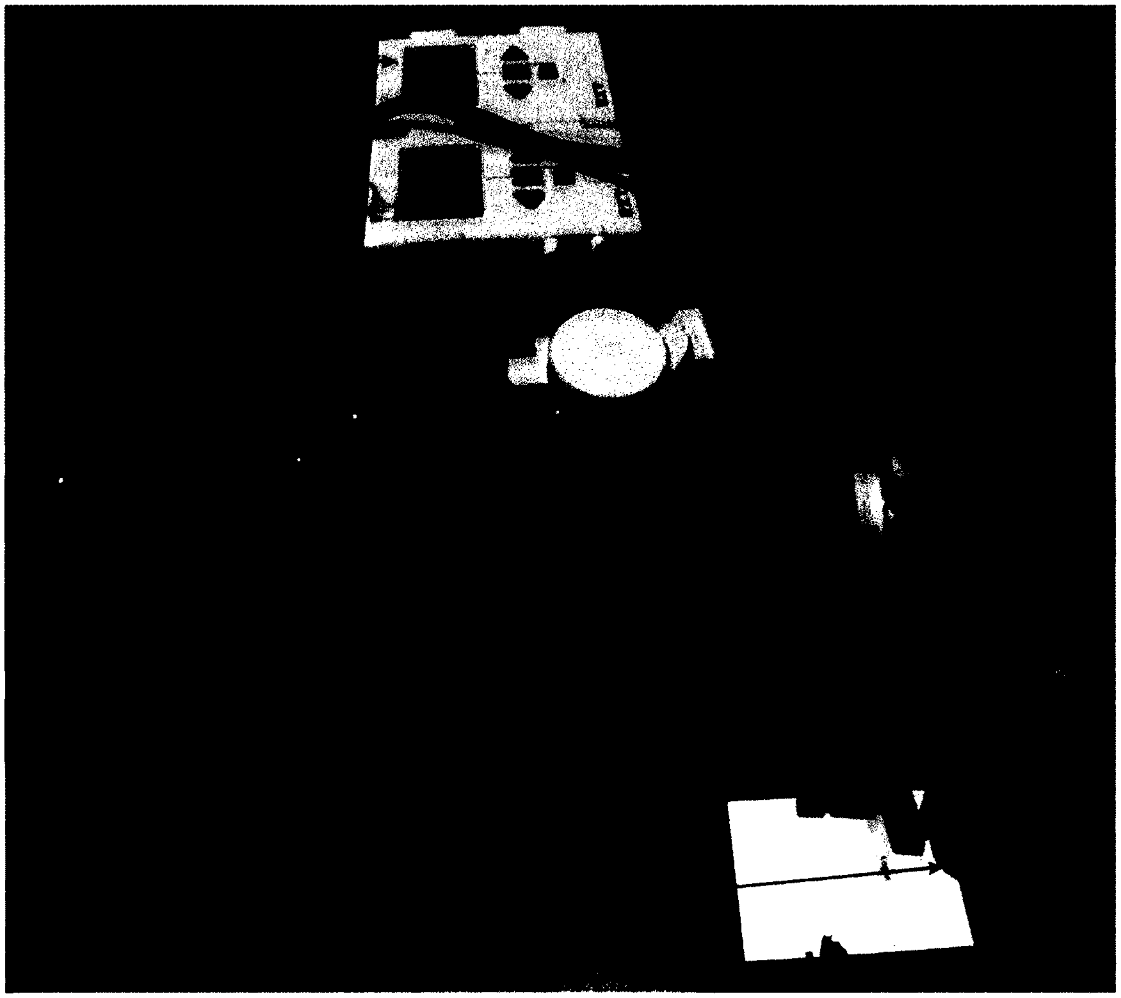
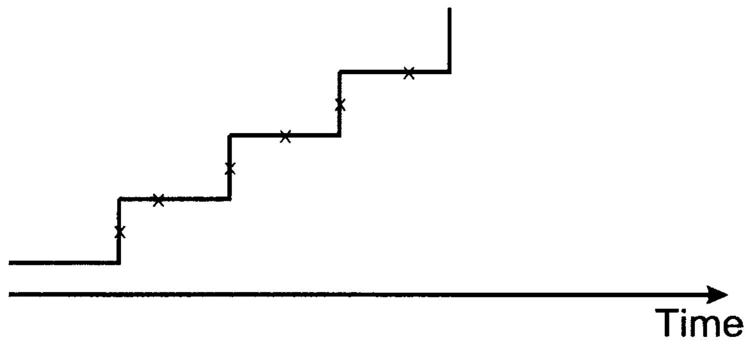


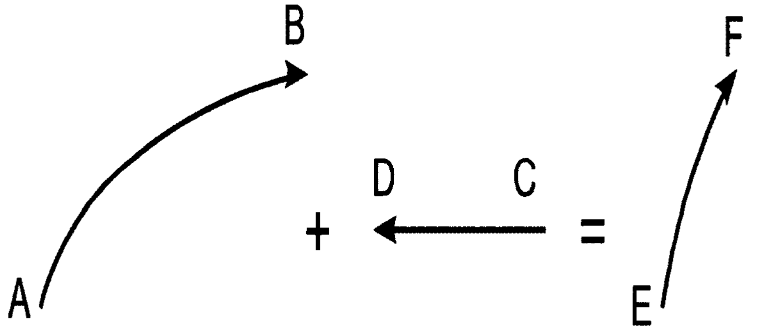
Figure 6.3: Mechanical design of the robot.

The main components used to build the robot are: 2 NXT Intelligent bricks, 3 motors, 4 gears, 4 wheels and the "finger". One of the Intelligent bricks serves as the CPU of the system while the second one helps prevent the robot from toppling over due to weight imbalances (also see Figure C.1, Appendix C). Motor C moves the pen (or robot finger) on and off the phone screen while motors A and B move the pen along the surface of the screen. The shape of the touch stroke is determined by the movement of motors A and B which run concurrently. Motor A drives the robot along a straight path while motor B, via a set of gears (see gear setup in Figure C.2, Appendix C) drives the framework supporting the robot "finger" along a circular path centered about the axis labeled R. To ensure that the robot does not deviate from the (approximately) straight path and get out of position (away from the phone region of interest), we connected it to a beam whose movement was restricted between a pair of rails firmly screwed on the board.

Based on observations on our dataset, most users' strokes deviate very slightly from a straight path. Because our attack seeks to mimic general user behavior (as opposed to the traits of an individual user), our robot was designed to generate near-straight strokes, with just a slight amount of curved behavior. Bar any effects arising out of the mechanical interactions between robot components, our mechanical and algorithmic design of the robot seeks to generate a straight stroke, with a very slight amount of curvature at a section of the stroke. Figure 6.4 (see Figure 6.4b in particular) illustrates the philosophy behind the curved section of our stroke. The combination of a motor driving the pen along a curved trajectory (i.e., the arc labeled AB) and a motor driving the pen along a straight line (i.e., path labeled CD) results into the curved stroke section such as EF.



(a) Design Philosophy #1: To generate a stroke, two motors run one after the other (i.e., serially). One motor makes a horizontal displacement, the other makes a vertical displacement. The × represents the points on the physical path which are sampled by the Android system to represent a stroke. The low sampling rate used by the Android OS (an average of 15 ms per sample in our experiments) guarantees that the system is blind to the true shape of the stroke since a few points on each segment are registered by the system.



(b) Design Philosophy #2: To generate a stroke, two motors run concurrently. The final trajectory (*e.g.*, EF) is a result of the superposition of the two motor movements (*e.g.*, AB and CD).

Figure 6.4: Philosophy behind the curved behavior of a touch stroke.

In our earlier design (see Figure 6.4a and (or) [77]), we used a different philosophy for the generation of a stroke. In that design, the mechanical and algorithmic implementation of the attack were such that the two motors controlling the shape of the stroke run sequentially at right angles to each other to form the zig-zag (or close to zig-zag) pattern that was our source of curvature. The coarse touch stroke sampling rate ($\approx$ 15 ms on average) on our Google Nexus phones ensured that the Android system was "blind" to the precise shape of the stroke. While the design was sufficient to illustrate the impact of the attack for our set of inputs, a generalized robotic attack based on that design could (in theory and possibly in practice) be thwarted in several ways. For example, an increment in the rate at which the Android system samples touch points on the screen would potentially expose the zig-zag shape of the stroke. Because humans don't typically generate zig-zag strokes, detection of the zig-zag pattern would be a key indicator for the possible occurrence of an attack. Alternatively a sensor (such as an accelerometer or gyroscope) could be used to detect repeated instances of a vibration pattern resulting from the zig-zag movement of the robotic finger executing several strokes.

Depending on the exact attack setup, exploiting these theoretical weaknesses in practice may not necessarily be be trivial, especially given that we used a small number of very short vertical and horizontal steps to create the saw-tooth shape. We however still move to eliminate this theoretical weakness by using the design represented by Figure 6.4b for this dissertation. Note though that the final stroke will not always be as smooth as trajectory EF due to the mechanical dynamics of the robot components.

It is noteworthy that owing to mechanical factors (*e.g.*, vibrations, friction, variations in the shape and size of the play-doh, *etc.*), the strokes produced by either approach can have notable variations from the behavior stipulated by the algorithm. With meticulous mechanical design of the robot however (*e.g.*, through careful selection of members, balancing weights, firmly securing vibrating elements, *etc.*), this behavior can be minimized. We implemented the attack using the Lego NXT

Mindstorms IDE [21] due to its straightforward support for motors with dissimilar inputs running in parallel.

Figure 6.5 depicts a graphical model of how the motors generate a single stroke. Motors B and C are connected in series with each other (i.e., they don't run at the same time), and their channel connected in parallel with that of Motor A. At the extreme ends of a stroke, the algorithm is such that the "finger" (see Motor C) is not in contact with the phone because the breaking (and in some cases starting) of the robot is associated with momentum effects which may cause a distorted pattern on the screen. The "finger" only moves towards the phone screen (see downward arrow – Motor C) after Motor A (which carries the full weight of the robot and hence the greatest momentum effects) has been in motion for sometime. At the end of a stroke, the "finger" begins to leave the screen before Motor A comes to a stop.

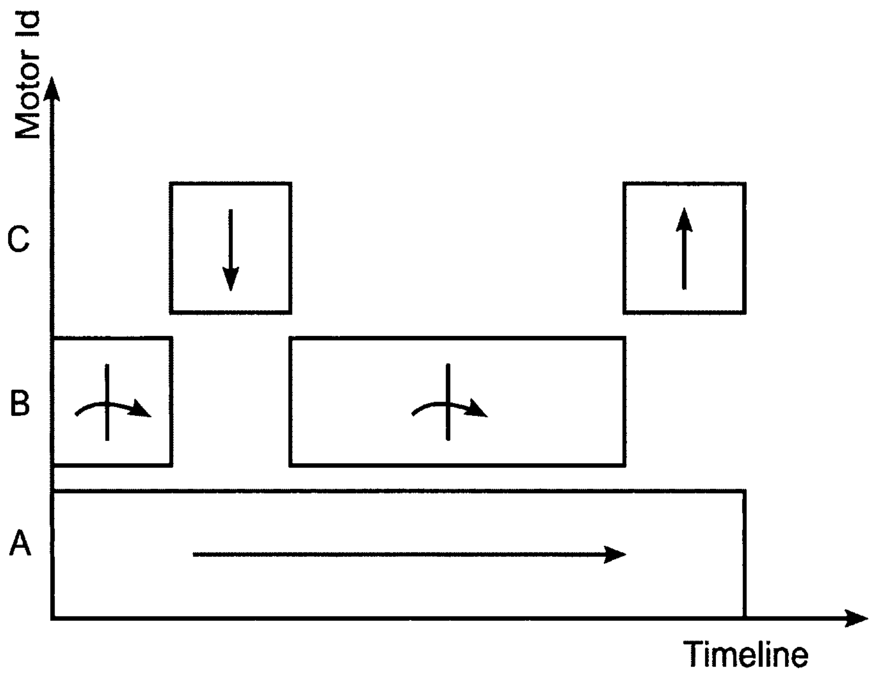


Figure 6.5: Model for Generation of a Stroke.

The exact duration for which the "finger" is in contact with the phone depends on the initial position of the "finger" relative to the phone. This is in fact one of the manual aspects of the attack, as we adjusted the position of the phone through trial and error (*e.g.*, by putting thin material under it) until the pressure that the "finger" exerted on the screen during swiping reached our required average[2]. In practice the attacker can use a phone (other than the one to be attacked) to guide these initial settings.

At the end of each stroke, we allowed a pause of about 1s in order for any existing vibrations to die out before the next stroke was executed. After a number of strokes, the robot may drift away from the region of interest since one end of the rails was left open to allow for randomness in the exact length of a stroke. To address this, we stopped the swiping, and then physically placed the robot back at the region of interest. For each of the blocks represented in Figure 6.5, the motors are given *power* and *speed* inputs (see Table 6.1).

Table 6.1: Parameter settings used for the robotic attack.

Motor Id	Power	Time
A	70	0.7
B_1	12	0.2
B_2	12	0.1
C_1	25	0.2
C_2	70	0.2

The time parameters are in seconds while the power parameters are a function of the voltage applied to the motor. These parameters are in essence just a general

[2]The Google Nexus S has a slight bulge at one end, so it does not sit perfectly horizontally if rested on a flat surface. We tried to compensate for this imbalance by raising one side of the phone more than the other.

guide — a slight change in the experimental conditions could call for changes in these inputs. B_1 is the first block representing Motor B while B_2 is the second block representing Motor B. The same notation is used for Motor C. The power input for the upward motion of Motor C is greater than that for its downward motion because the motor has to support the "finger" and its attached mechanism during the upward movement. For the downward movement, the weight of the "finger" and its parts assist the motor instead.

To generate a horizontal stroke, we positioned the phone such that the finger started around the point with coordinates (363, 541) and moved towards the point having the coordinates (145, 588)[3]. For the vertical strokes, the phone was positioned such that the start coordinates were approximately (320,613) and the direction of the finger being towards the point (352,400). These values were the means/averages observed over a portion of the full user population. Unlike in [77] where we explicitly generated noise to cause variations in the different features across different strokes, we rely on noise arising out of the mechanical interactions between robot elements as our sole source of randomness. The fact that the whole robotic framework moves the entire length of the stroke introduces a great amount of noise (*e.g.*, due to vibrations of the members). We find this noise to be sufficient to ensure that the robotic strokes are not exactly similar to each other.

We set the *power* and *time* inputs of Motor C to the values tabulated so as to get pressure outputs of between 0.4 and 0.6 on the phone screen. Depending on factors such as the area of the finger and the initial position of the finger relative to

[3]In practice it only moves towards some point in the neighborhood of the point in question.

the phone, one may have to set different values for these parameters in order to get pressure outputs in this range. Like in the case of the finger area (see Section 6.3.3.), we not only iteratively set the pressure during initial experiments, but also depended on a connected battery to increase the pressure to the required range. To ease the task of setting the various attack parameters, we enabled pointer locations (under developer options) so as to view the strokes and their associated raw feature outputs on the screen during the fine-tuning phase.

6.4 Attack Performance Evaluation

6.4.1 Verification Algorithms

We demonstrate the impact of the attack using a Support Vector Machine (SVM) [78] and the k-Nearest Neighbors (k-NN) classifier [79]. We select these two verification algorithms because they have recently been shown in [11] and [10] to perform very well for continuous touch-based authentication. We briefly describe the mechanisms of operation of the two algorithms below:

Support Vector Machine: An SVM is a binary classifier which uses a hyperplane to separate two data classes in such a way that the margin between the two classes is maximized. The margin is the distance between the hyperplane and the boundary observations which are also referred to as support vectors. For classes that are not linearly separable in a given feature space, it is sometimes necessary to map the original data points to a higher dimensional space with the aid of a kernel function. We used the Gaussian radial-basis function as our kernel, like was done in

[10]. During classification, we set a given user's data as the positive class, and a set of samples randomly selected from the other users as the negative class.

k-Nearest Neighbors: During training, this classifier does not have to extract any model from the data—it only stores the feature vectors from the different classes (in our case two classes). Given a new observation that is to be assigned a class label, the k-NN classifier assigns it to the class A if the majority of the k closest training vectors to the new observation belong to the class A. Different researchers use different measures to represent the distance between the training vectors and a test observation. In this work we use the Eucledian distance metric since it was also used in [10]. Like we did for the SVM, during training, we set a given user's data as the positive class (genuine class), and a set of randomly selected samples from the rest of the population as the negative class (impostor class).

For both the k-NN and SVM, we used WEKA [67] via its Java API to implement the classification system. We used k=9 for the k-NN classifier since this value gave us the best performance. For all other parameters across the two classifiers, we used the WEKA defaults.

6.4.2 Training and Testing Methodology

Training and Zero-effort Testing: Training was done based on data collected during *Session I* while zero-effort testing was done based on data collected during *Session II.* For each user, we distinguished between portrait and landscape strokes, and further distinguished between horizontal and vertical strokes for each of the two phone orientations. This way, each user had four reference templates. The reason for separating between these four types of strokes was because certain features change depending on the type of stroke and the way in which the phone is held when the stroke is executed. For example, for the typical user, a horizontal stroke executed in portrait mode will very likely have different start and end-points (among other features) from a horizontal stroke executed in landscape mode. Owing to the mismatch between features, a classification mechanism that does not distinguish between these two types of strokes will likely perform unreliably.

In practice we believe that a touch-based authentication application should use all four types of reference templates since users can switch between stroke types depending on the type and organization of content they read on the phone. Regardless of whether a user is biased towards a certain type of stroke, the system should be able to accurately perform classification during those times when the user executes the other kinds of strokes.

For each of the four categories of strokes, we only performed our analysis for those users who executed at least 80 strokes during *Session I.* For the portrait strokes we had 106 and 118 users who met this 80 strokes requirement for the horizontal and vertical strokes respectively. For the landscape strokes, we had 41 and 50 users who met the requirement for the horizontal and vertical strokes respectively. For training, we used 80 strokes executed by the user in question (i.e., genuine or positive class) and 5 strokes from each of the other users (i.e., the impostor or negative class) for each of the four categories of strokes.

To establish a baseline against which to measure the impact of the robotic attack, we carried out zero-effort testing for each user. In these tests, to launch an *impostor attack* against a given user's template, we used 10 strokes from each of the other users. To carry out a *genuine attack* against a given user's template, we used all the strokes captured from that particular user during *Session II.* Because a user will every now and then execute a stroke which is very distinct from the rest of her strokes, we used a block of strokes, rather than a single stroke to make authentication decisions.

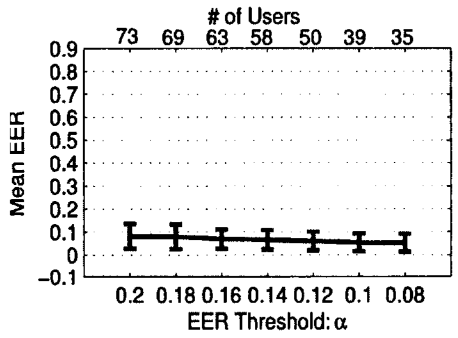
Each legitimate or impostor authentication attempt was based on a single vector derived from 10 consecutive feature vectors (or strokes). The single authentication vector was computed such that its elements were the component-wise means of the 10 vectors contained in a sliding window. From the results obtained from these tests, we generated four Detection Error Tradeoff (DET) curves [80] for each user, one for each kind of swiping. From each of these curves, we determined the Equal Error Rate. *Robotic Testing* The robotic testing process was the same as that described in Section 6.4.2, except that the *impostor attack* was based on samples generated by the robot. We will refer to the impostor attack in this case as the *robotic impostor attack.* We used 600 strokes generated by the robot to carry out this attack against each user. Like we did in the zero-effort tests, we again generated two DET curves for each user, and calculated the EER from each of the curves.

6.4.3 Attack Results

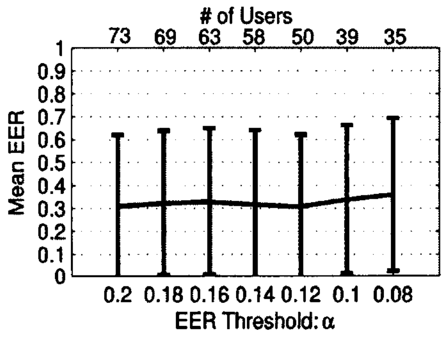
The Failure to Enroll Policy: To rigorously evaluate the impact of the attack, we employed a "*failure to enroll*" policy in which we only enrolled users whose mean EERs across the two verifiers at baseline were less than a certain EER threshold

(α)[4]. Our attack performance evaluation was done at values of α ranging from 0.2 to 0.08. We chose an upper bound of α=0.2 because we believe that a user with an EER higher than that would probably not use the technology anyway. For the lower bound we decided to use α=0.08 because the number of users able to enroll on the system became too small for values of α less than that.

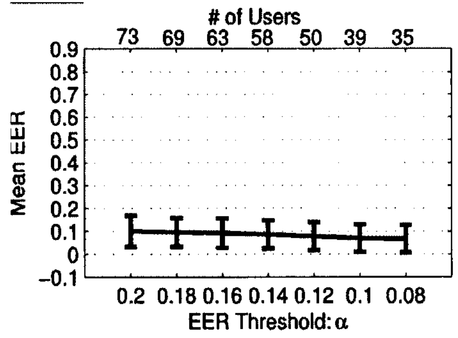
Mean Impact of the Attack: Figures 6.6 and 6.7 respectively summarize the effect of the attack on the mean and standard deviation of the classifier EERs.



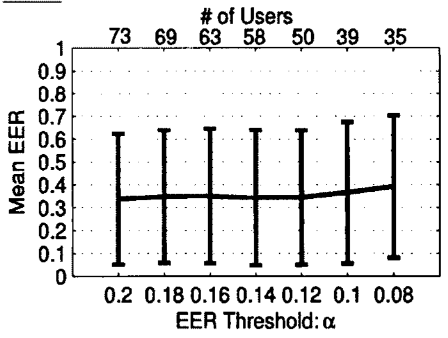
(a) Mean and standard deviation of EERs obtained for the SVM verifier <u>*before*</u> the robotic attack.



(b) Mean and standard deviation of EERs obtained for the SVM verifier <u>*after*</u> the robotic attack.



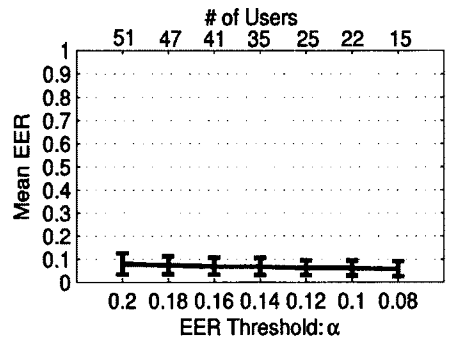
(c) Mean and standard deviation of EERs obtained for the k-NN verifier <u>*before*</u> the robotic attack.



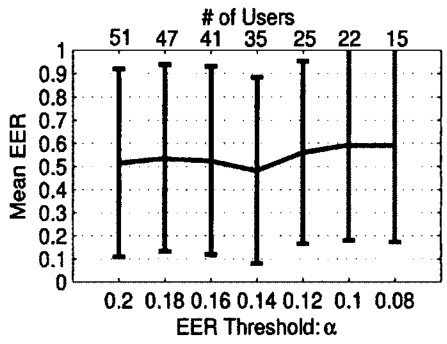
(d) Mean and standard deviation of EERs obtained for the k-NN verifier <u>*after*</u> the robotic attack.

Figure 6.6: Impact of the robotic attack on the classification of the <u>*horizontal*</u> strokes generated in portrait mode.

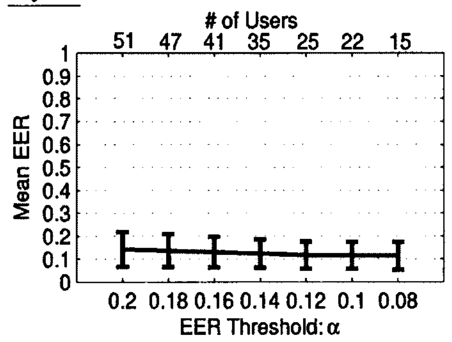
[4]For each user we computed the EER seen with each of the k-NN and SVM verifiers at baseline and found the mean of these two values. It is this mean value that we compared with α in order to make "*failure to enroll*" decisions.



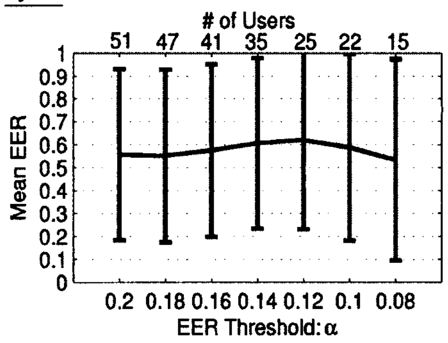
(a) Mean and standard deviation of EERs obtained for the SVM verifier *before* the robotic attack.



(b) Mean and standard deviation of EERs obtained for the SVM verifier *after* the robotic attack.



(c) Mean and standard deviation of EERs obtained for the k-NN verifier *before* the robotic attack.



(d) Mean and standard deviation of EERs obtained for the k-NN verifier *after* the robotic attack.

Figure 6.7: Impact of the robotic attack on the classification of the *vertical* strokes generated in portrait mode.

For different values of α, we computed the mean EER and standard deviation of the EERs across the population before and after the robotic impostor attack. Figures 6.6 and 6.7 respectively summarize these results for the horizontal and vertical touch strokes. The bottom (horizontal) axis shows the different EER thresholds (α), while the top (horizontal) axis shows the number of users who were able to enroll onto the system at each value of α. Before the robotic attack was launched we obtained EERs of between 0.13 and 0.035 (see plots on the left side of Figures 6.6 and 6.7). These EERs are higher than the EERs reported in [10], but comparable to those reported in [12] during the sub-set of experiments in which the users did not wear a digital sensor glove. With our baseline EERs[5] (i.e., EERs before attack) being comparable to the EERs reported in the literature, we proceeded to evaluate the impact of the robotic attack.

Observe (Figures 6.6b, 6.6d, 6.7b, 6.7d) that for both the vertical and horizontal strokes, the attack drastically increased both the mean EERs and the standard deviation of the EERs. The high mean EERs indicate that users begin to see very high False Reject Rates (FRRs), while impostors see equally high False Acceptance Rates (FARs). Also, the high variance in EERs implies that system performance becomes very unreliable/unpredictable as a result of the attack. It is noteworthy that the heightened EERs and standard deviations persist for both verification algorithms even when the system is used only by the best performing users (i.e., α=0.08). This implies that a defence mechanism centered around barring the poor users from enrolling onto the system would not thwart the attack. Table 6.2 gives a more precise view of the impact of the attack on the mean EERs.

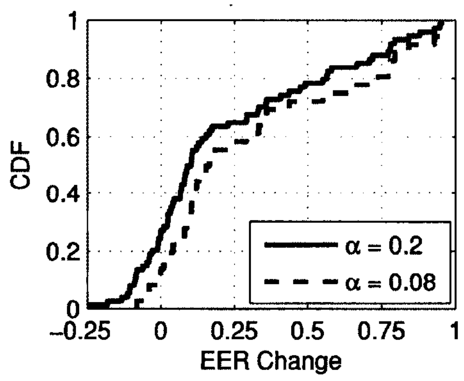
Table 6.2: Percentage increment in mean EER due to the robotic impostor attack on the portrait strokes.

α	SVM		k-NN	
	Horizontal	Vertical	Horizontal	Vertical
0.2	377.3%	638.6%	333.8%	382.9%
0.18	407.6%	734.9%	375.7%	405.6%
0.16	479.8%	752.9%	401.2%	475.4%
0.14	486.9%	702.8%	436.2%	583.1%
0.12	522.2%	1021.7%	509.7%	779.4%
0.1	695.4%	1175.8%	691.9%	827.0%
0.08	799.5%	1535.6%	803.3%	863.2%

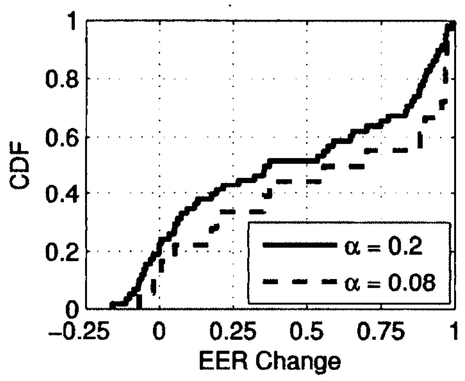
[5]See our work in [81] for an in-depth analysis of the baseline EERs of various algorithms.

The table shows the percentage change in mean system EER seen by each verification algorithm as a result of the attack. Regardless of the verification algorithm or *failure to enroll* threshold, the percentage change in mean EER is beyond 300% in all cases, and over 1500% in the most extreme case. These results confirm why the robotic attack would significantly degrade the performance of a touch-based authentication system.

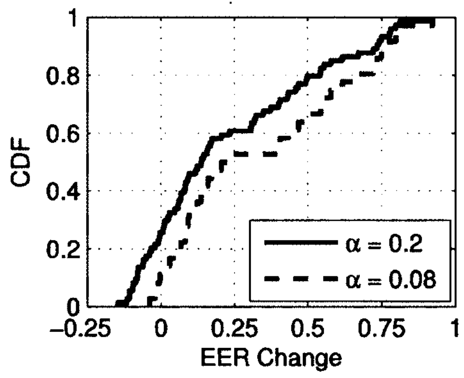
Impact of the Attack on each User: Figure 6.8 summarizes the impact of the attack on each user's verification performance.



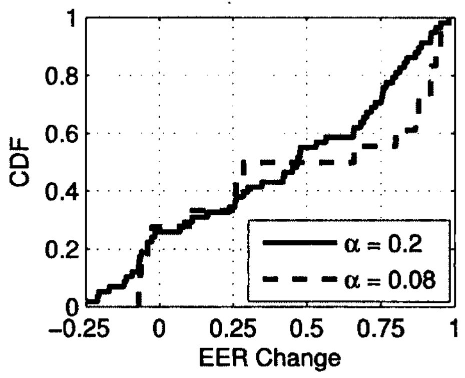
(a) SVM performance with the horizontal strokes.



(b) SVM performance with the with vertical strokes.



(c) k-NN performance with the with horizontal strokes.



(d) k-NN performance with the with vertical strokes.

Figure 6.8: Impact of the attack on each user's portrait strokes.

For each user, we subtracted the EER seen under the zero-effort attack from that seen under the robotic attack and then plotted the CDFs of these changes in EER for each of the two extreme failure-to-enroll thresholds. For this analysis we only present results for α=0.2 and α=0.08 since the other values of α did not give us any new insights. The plot reveals two salient features:

1. There was a proportion of users (in some cases up to 30% of the population) whose EER changes were negative. For these kinds of users, the robotic attack actually performed worse (i.e., caused lower EERs) than the zero-effort attack. Since our attack was designed based on data gleaned from the population, this trend suggests that there is a proportion of users (say, 30% of the population) whose touch gesture biometric footprint is very distinct from that of the majority of the users.
2. There was a proportion of users who had EER changes that were extremely high (close to 1). These types of users likely had their touch biometric patterns very similar to the mean values observed over the population.

These two features to some extent explain the high variance seen in Figures 6.6b, 6.6d, 6.7b and 6.7d, since a combination of users seeing decrements in EER and others seeing very drastic increments in EER must have resulted into a population having very high variability in EER relative to the variability seen before the robotic attack. Results obtained with the phone held in landscape mode are left out because they did not provide any noteworthy new insights.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this Dissertation we conducted a rigorous pattern analysis on keystroke and touch biometrics data and leveraged the observed traits to design a new family of attacks that break keystroke and touch authentication systems. We evaluated the impact of our algorithmic attack on the best verification algorithms in the keystroke and touch authentication fields and compared the performance to that seen with the traditional zero-effort attacks.

When subjected to zero-effort impostor attacks, the EERs of the keystroke verification algorithms were between 0.2 and 0.08 for a set of password strings whose lengths ranged between 7 and 20 characters. This range of EERs was comparable to the zero-effort EERs reported in the benchmark study in [6]. When we launched our algorithmic attack, the EERs of the three verifiers increased by between 28.6% and 84.4%, relative to the zero-effort EERs. Also, we found that the shorter passwords were more vulnerable to the attacks, and that a small proportion of the user-population accounted for most of the performance degradation caused by the algorithmic attack.

For the touch-based authentication system, the mean EERs of the verification algorithms were between 0.08 and 0.035 under the zero-effort attack. This range of

EERs was comparable to the EERs reported in the literature [12]. When we launched our robotic attack, the mean EERs of the verification algorithms increased by between 333.8% and 1535.6% depending on the failure-to-enroll threshold and type of touch stroke subjected to attack. Like was observed with the keystroke attacks, a subset of the population resisted the attack, while another subset of users badly succumbed to the attack.

In general, the results from our research indicate that in comparison to the zero-effort attacks typically used to test keystroke and touch authentication systems, our algorithmic attacks were considerably much more lethal. The kinds of synthetic attacks presented in this work rest on two premises: 1) The large amounts of keystroke and touch biometrics data required to design the attack can be easily accessed by committed adversaries, and, 2) The software tools and cheap easily programmable robots required to implement the attacks are within the reach of adversaries. From evidence cited in this work, these are realistic assumptions, implying that a keystroke or touch-based authentication system would have a decent chance of being subjected to such a kind of attack in practice.

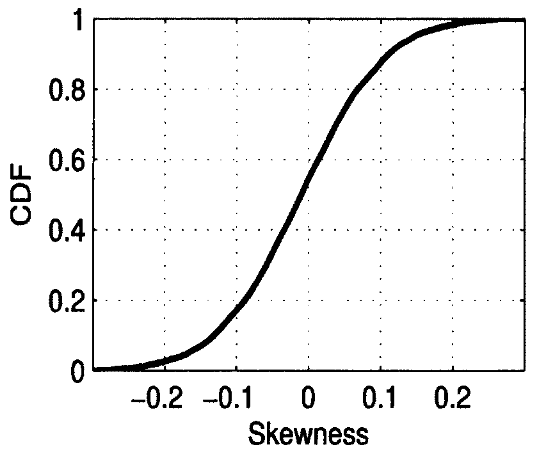
There are several aspects of our attacks that might need further research. First, like most past studies in this area (see works cited in Chapter 2), our touch biometrics data collection was based on a group of users who used a small number of specialized applications (two applications in our case). In practice, people use a wide range of applications, some of which are designed for tasks which could prompt "touch signatures" (*e.g.*, with regard to regions of the phone that people touch) that are very different from those seen with our applications. It would be interesting to determine how the attack scales to a large number of applications.

Another area worthy of investigation is whether a touch stroke could be decomposed into a set of features that are more resilient to this kind of attack than our features. Because touch-based authentication does not yet have a standard set of features universally used by all researchers, we defined a set of 28 features that captured the key statistical attributes exhibited along a stroke. The underlying philosophy behind our feature definitions is not so different from that of the features used in past work, however, this does not guarantee that all feature-sets will succumb to the attack in exactly the same way. It is thus interesting to determine how much less or how much more the other features are affected by the attack. Similar kinds of questions can be raised about our keystroke attacks — *e.g.*, with regard to the variety of keyboards used during data collection, the variety of texts analyzed and the question of how the findings relate to free-text keystroke authentication. Investigations into the effects of changes in these variables would greatly increase the community's understanding of the extent of the threat posed by these types of attacks.

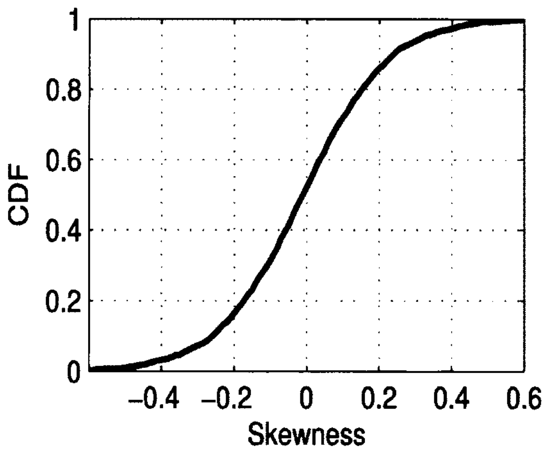
The above open research problems notwithstanding, our attacks highlight previously unknown threats to keystroke and touch-based authentication. Our findings do not only call for more stringent performance evaluation of keystroke and touch-based authentication systems, but should also motivate research into technologies which could defend against the larger family of robotic and software attacks, two instances of which have been demonstrated in this dissertation.

APPENDIX A

CHECKING FOR THE SKEWNESS REQUIREMENT OF THE WILCOXON SIGNED-RANK TEST



(a) Distribution of skewness values of ΔD_f vectors when both f_1 and f_2 were KHTs.



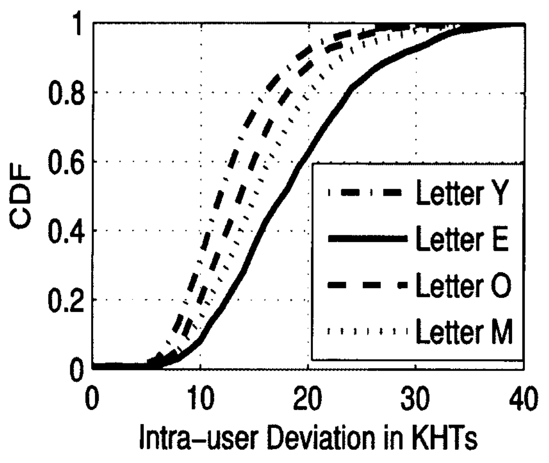
(b) Distribution of skewness values of ΔD_f vectors when both f_1 and f_2 were KITs.

Figure A.1: Skewness of ΔD_f vectors.

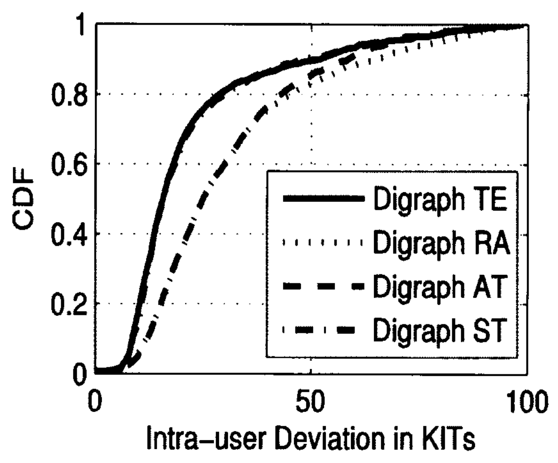
In Figure A.1a, the ΔD_f vectors are computed such that the features f_1 and f_2 forming a pair are both KHTs. For each feature-pair, the associated ΔD_f vector is computed based on a set of 3000 randomly selected user-pairs. Since our test-phrase contains 58 KHTs, there are $\binom{58}{2} = 1653$ possible feature-pairs that can be formed out of the set of KHTs. We compute a ΔD_f vector for each of these feature-pairs, and calculate the skewness value of this ΔD_f vector. Figure A.1a is a CDF of the full array of skewness values obtained across all 1653 KHT feature-pairs. With both f_1 and f_2 being KITs, we repeat the procedure to generate the CDF in Figure A.1b. Results on the two plots generally supported the use of the Wilcoxon signed-rank test, as all KHT feature-pairs had a ΔD_f vector with skewness between -0.3 and 0.3, and about 95% of KIT feature-pairs had a ΔD_f vector with skewness between -0.5 and 0.5.

APPENDIX B

INTRA-USER VARIATIONS FOR SELECTED KHTS AND KITS



(a) CDF of intra-user standard deviations of KHTs over the full population.



(b) CDF of intra-user standard deviations of KITs over the full population.

Figure B.1: Intra-user variability of KHTs and KITs.

For each of the 8 features used to plot the two graphs, we calculate the standard deviation exhibited by each user in our population and then generate a CDF of the standard deviations. In the context of a statistical attack, high intra-user variability for a given feature indicates that the user's profile for the feature in question could be matched by a wide range of guesses. For clarity of the plots, standard deviations exceeding 40 ms and 100 ms for the KHTs (Figure B.1a) and KITs (Figures B.1b) respectively, were filtered off before generating the CDFs, since a very small proportion of users had standard deviations exceeding these thresholds. The 8 selected features show the general trend observed across the dataset. Observe that different features did not necessarily have similarly shaped distributions, an indication of why the adversary would benefit from a feature-by-feature understanding of the statistical traits exhibited by keystroke data associated with the password in question.

APPENDIX C

LEGO CONSTRUCTION

The NXT brick at the far end right provides a balancing moment that prevents the robot from toppling over due to the combined weight of the motor and finger-support mechanism on the other side of the robot.

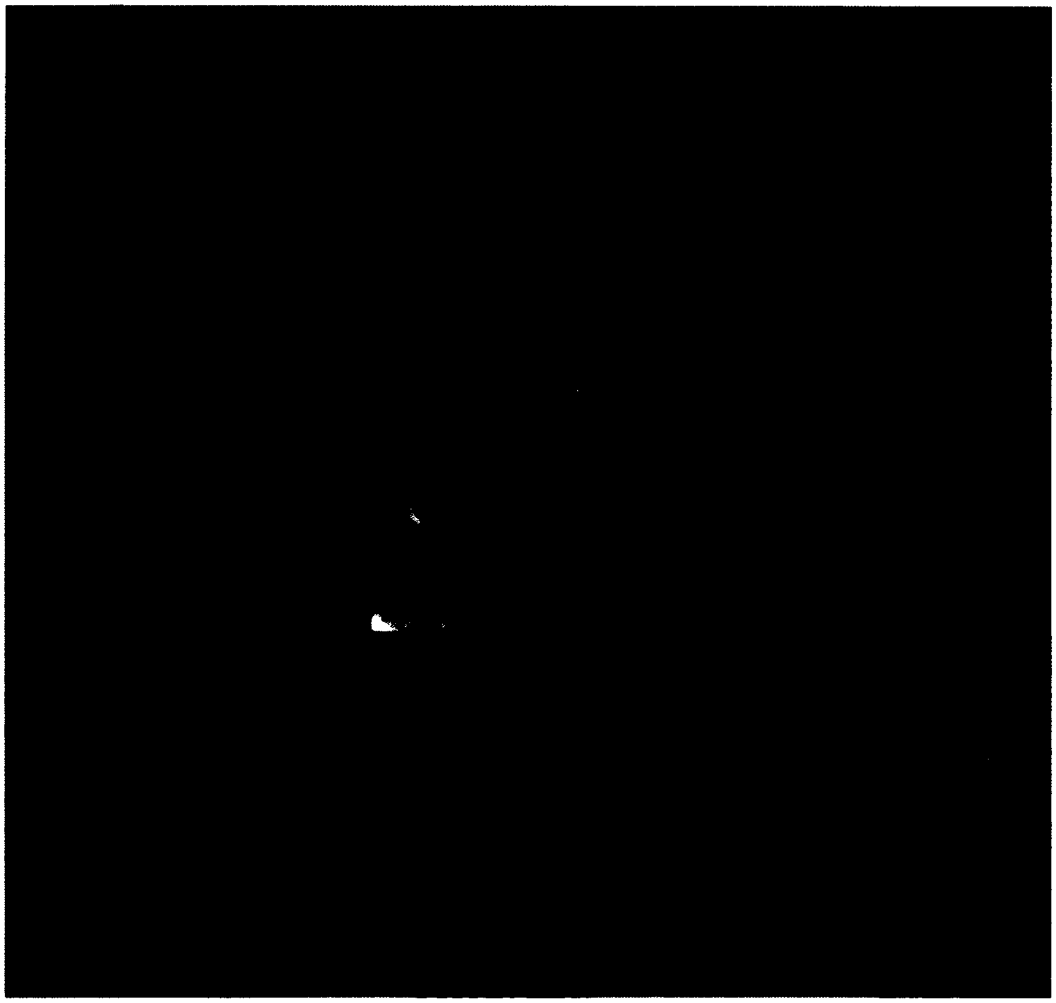


Figure C.1: Aerial view of the robot.

The gear-pairs are selected in such a way to produce a low speed and high torque to drive the "finger" and its support-mechanism.

Figure C.2: Gear mechanism driving the robot "finger" along a circular arc.

APPENDIX D

HUMAN USE IRB APPROVAL DOCUMENTS

D.1 HUC 1086

LOUISIANA TECH
UNIVERSITY

MEMORANDUM

OFFICE OF UNIVERSITY RESEARCH

TO:	Mr. Abdul Serwadda and Dr. Phoha
FROM:	Barbara Talbot, University Research
SUBJECT:	HUMAN USE COMMITTEE REVIEW
DATE:	April 11, 2013

In order to facilitate your project, an EXPEDITED REVIEW has been done for your proposed study entitled:

"Characterizing Mobile Phone Users Based on Typing Patterns Touch Gestures and Body Movement"

HUC 1086

The proposed study's revised procedures were found to provide reasonable and adequate safeguards against possible risks involving human subjects. The information to be collected may be personal in nature or implication. Therefore, diligent care needs to be taken to protect the privacy of the participants and to assure that the data are kept confidential. Informed consent is a critical part of the research process. The subjects must be informed that their participation is voluntary. It is important that consent materials be presented in a language understandable to every participant. If you have participants in your study whose first language is not English, be sure that informed consent materials are adequately explained or translated. Since your reviewed project appears to do no damage to the participants, the Human Use Committee grants approval of the involvement of human subjects as outlined.

Projects should be renewed annually. ***This approval was finalized on April 11, 2013 and this project will need to receive a continuation review by the IRB if the project, including data analysis, continues beyond April 11, 2014.*** Any discrepancies in procedure or changes that have been made including approved changes should be noted in the review application. Projects involving NIH funds require annual education training to be documented. For more information regarding this, contact the Office of University Research.

You are requested to maintain written records of your procedures, data collected, and subjects involved. These records will need to be available upon request during the conduct of the study and retained by the university for three years after the conclusion of the study. If changes occur in recruiting of subjects, informed consent process or in your research protocol, or if unanticipated problems should arise it is the Researchers responsibility to notify the Office of Research or IRB in writing. The project should be discontinued until modifications can be reviewed and approved.

If you have any questions, please contact Dr. Mary Livingston at 257-2292 or 257-5066.

D.2 HUC 416

LOUISIANA TECH
UNIVERSITY

OFFICE OF UNIVERSITY RESEARCH

MEMORANDUM

TO: Dr. Vir Phoha

FROM: Dr. Les Guice, V.P. for Research & Development

SUBJECT: Human Use Committee Review

DATE: November 19, 2012

RE: Approved Continuation of Study HUC 416 with Attached Amendments

TITLE: **"Studies Related to the use of Keystroke Dynamics as a Biometric"**

HUC- 416 Adding Amendment Dated October 30, 2012

The above referenced study has been approved as of November 19, 2012 as a continuation of the original study that received approval on September 7, 2008. **This project will need to receive a continuation review by the IRB if the project, including collecting or analyzing data, continues beyond November 19, 2013.** Any discrepancies in procedure or changes that have been made including approved changes should be noted in the review application. Projects involving NIH funds require annual education training to be documented. For more information regarding this, contact the Office of University Research.

You are requested to maintain written records of your procedures, data collected, and subjects involved. These records will need to be available upon request during the conduct of the study and retained by the university for three years after the conclusion of the study. If changes occur in recruiting of subjects, informed consent process or in your research protocol, or if unanticipated problems should arise it is the Researchers responsibility to notify the Office of Research or IRB in writing. The project should be discontinued until modifications can be reviewed and approved.

If you have any questions, please contact Dr. Mary Livingston at 257-4315.

LOUISIANA TECH
UNIVERSITY

OFFICE OF UNIVERSITY RESEARCH

MEMORANDUM

TO: Dr. Vir Phoha

FROM: Barbara Talbot, University Research

SUBJECT: Human Use Committee Review

DATE: March 1, 2010

RE: Approved Continuation of Study HUC 416
Changing Number of Subjects from 500 to 2000

TITLE: **"Studies Related to the use of Keystroke Dynamics as a Biometric"**

# HUC- 416

The above referenced study has been approved as of March 1, 2011 as a continuation of the original study that received approval on September 7, 2008. **This project will need to receive a continuation review by the IRB if the project, including collecting or analyzing data, continues beyond March 1, 2012.** Any discrepancies in procedure or changes that have been made including approved changes should be noted in the review application. Projects involving NIH funds require annual education training to be documented. For more information regarding this, contact the Office of University Research.

You are requested to maintain written records of your procedures, data collected, and subjects involved. These records will need to be available upon request during the conduct of the study and retained by the university for three years after the conclusion of the study. If changes occur in recruiting of subjects, informed consent process or in your research protocol, or if unanticipated problems should arise it is the Researchers responsibility to notify the Office of Research or IRB in writing. The project should be discontinued until modifications can be reviewed and approved.

If you have any questions, please contact Dr. Mary Livingston at 257-4315.

LOUISIANA TECH
U N I V E R S I T Y

OFFICE OF UNIVERSITY RESEARCH

MEMORANDUM

TO:	Dr. Vir Phoha
FROM:	Barbara Talbot, University Research
SUBJECT:	Human Use Committee Review
DATE:	September 28, 2009
RE:	Approved Continuation of Study HUC 416
TITLE:	**"Studies Related to the use of Keystroke Dynamics as a Biometric"**

# HUC- 416

The above referenced study has been approved as of September 16, 2009 as a continuation of the original study that received approval on September 7, 2008. **This project will need to receive a continuation review by the IRB if the project, including collecting or analyzing data, continues beyond September 16, 2010.** Any discrepancies in procedure or changes that have been made including approved changes should be noted in the review application. Projects involving NIH funds require annual education training to be documented. For more information regarding this, contact the Office of University Research.

You are requested to maintain written records of your procedures, data collected, and subjects involved. These records will need to be available upon request during the conduct of the study and retained by the university for three years after the conclusion of the study. If changes occur in recruiting of subjects, informed consent process or in your research protocol, or if unanticipated problems should arise it is the Researchers responsibility to notify the Office of Research or IRB in writing. The project should be discontinued until modifications can be reviewed and approved.

If you have any questions, please contact Dr. Mary Livingston at 257-4315.

LOUISIANA TECH
UNIVERSITY

OFFICE OF UNIVERSITY RESEARCH

MEMORANDUM

TO: Dr. Vir Phoha

FROM: Barbara Talbot, University Research

SUBJECT: HUMAN USE COMMITTEE REVIEW

DATE: September 16, 2008

In order to facilitate your project, an EXPEDITED REVIEW has been done for your proposed study entitled:

"Studies Related to the use of Keystroke Dynamics as a Biometric"

# HUC-416

The proposed study's revised procedures were found to provide reasonable and adequate safeguards against possible risks involving human subjects. The information to be collected may be personal in nature or implication. Therefore, diligent care needs to be taken to protect the privacy of the participants and to assure that the data are kept confidential. Informed consent is a critical part of the research process. The subjects must be informed that their participation is voluntary. It is important that consent materials be presented in a language understandable to every participant. If you have participants in your study whose first language is not English, be sure that informed consent materials are adequately explained or translated. Since your reviewed project appears to do no damage to the participants, the Human Use Committee grants approval of the involvement of human subjects as outlined.

Projects should be renewed annually. ***This approval was finalized on September 4, 2008 and this project will need to receive a continuation review by the IRB if the project, including data analysis, continues beyond September 4, 2009.*** Any discrepancies in procedure or changes that have been made including approved changes should be noted in the review application. Projects involving NIH funds require annual education training to be documented. For more information regarding this, contact the Office of University Research.

You are requested to maintain written records of your procedures, data collected, and subjects involved. These records will need to be available upon request during the conduct of the study and retained by the university for three years after the conclusion of the study. If changes occur in recruiting of subjects, informed consent process or in your research protocol, or if unanticipated problems should arise it is the Researchers responsibility to notify the Office of Research or IRB in writing. The project should be discontinued until modifications can be reviewed and approved.

If you have any questions, please contact Dr. Mary Livingston at 257-4315.

LOUISIANA TECH
UNIVERSITY

OFFICE OF UNIVERSITY RESEARCH

MEMORANDUM

TO: Dr. Vir Phoha

FROM: Barbara Talbot, University Research

SUBJECT: HUMAN USE COMMITTEE REVIEW

DATE: September 17, 2007

In order to facilitate your project, an EXPEDITED REVIEW has been done for your proposed study entitled:

"Studies Related to the use of Keystroke
Dynamics as a Biometric

# HUC-416

The proposed study's revised procedures were found to provide reasonable and adequate safeguards against possible risks involving human subjects. The information to be collected may be personal in nature or implication. Therefore, diligent care needs to be taken to protect the privacy of the participants and to assure that the data are kept confidential. Informed consent is a critical part of the research process. The subjects must be informed that their participation is voluntary. It is important that consent materials be presented in a language understandable to every participant. If you have participants in your study whose first language is not English, be sure that informed consent materials are adequately explained or translated. Since your reviewed project appears to do no damage to the participants, the Human Use Committee grants approval of the involvement of human subjects as outlined.

Projects should be renewed annually. ***This approval was finalized on September 7, 2007 and this project will need to receive a continuation review by the IRB if the project, including data analysis, continues beyond September 7, 2008.*** Any discrepancies in procedure or changes that have been made including approved changes should be noted in the review application. Projects involving NIH funds require annual education training to be documented. For more information regarding this, contact the Office of University Research.

You are requested to maintain written records of your procedures, data collected, and subjects involved. These records will need to be available upon request during the conduct of the study and retained by the university for three years after the conclusion of the study. If changes occur in recruiting of subjects, informed consent process or in your research protocol, or if unanticipated problems should arise it is the Researchers responsibility to notify the Office of Research or IRB in writing. The project should be discontinued until modifications can be reviewed and approved.

If you have any questions, please contact Dr. Mary Livingston at 257-4315.

BIBLIOGRAPHY

[1] Y. Sheng, V. Phoha, and S. Rovnyak, "A parallel decision tree-based method for user authentication based on keystroke patterns," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 35, no. 4, pp. 826 –833, aug. 2005.

[2] F. Monrose, M. K. Reiter, and S. Wetzel, "Password hardening based on keystroke dynamics," in *International Journal of Information Security.* ACM Press, 1999, pp. 73–82.

[3] F. Monrose, M. K. Reiter, Q. Li, D. P. Lopresti, and C. Shih, "Toward speech-generated cryptographic keys on resource-constrained devices," in *Proceedings of the 11th USENIX Security Symposium.* Berkeley, CA, USA: USENIX Association, 2002, pp. 283–296. [Online]. Available: http://dl.acm.org/citation.cfm?id=647253.720284

[4] L. Ballard, S. Kamara, and M. K. Reiter, "The practical subtleties of biometric key generation," in *In Proceedings of the 17 th Annual USENIX Security Symposium*, 2008, pp. 61–74.

[5] K. S. Balagani, V. V. Phoha, A. Ray, and S. Phoha, "On the discriminability of keystroke feature vectors used in fixed text keystroke authentication," *Pattern Recognition Letters*, Feb. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2011.02.014

[6] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *DSN*, 2009, pp. 125–134.

[7] Z. Wang, A. Serwadda, K. Balagani, and V. Phoha, "Transforming animals in a cyber-behavioral biometric menagerie with frog-boiling attacks," in *Fifth IEEE International Conference on Biometrics: Theory Applications and Systems (BTAS),*, sept. 2012.

[8] D. Gunetti and C. Picardi, "Keystroke analysis of free text." *ACM Trans. Inf. Syst. Secur.*, vol. 8, no. 3, pp. 312–347, 2005.

[9] R. Khandaker, K. Balagani, and V. Phoha, "Making impostor pass rates meaningless: A case of snoop-forge-replay attack on continuous cyber-behavioural verification with keystrokes," in *IEEE Computer Society and IEEE Biometrics Council Workshop on Biometrics*, ser. BIOM, 2011.

[10] F. Mario, B. Ralf, M. Eugene, M. Ivan, and S. Dawn, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 136–148, 2013.

[11] L. Li, X. Zhao, and G. Xue, "Unobservable reauthentication for smart phones," in *Proceedings of the 20th Network and Distributed System Security Symposium*, ser. NDSS'13. Reston, VA: Internet Society, 2013.

[12] F. Tao, L. Ziyi, C. Bogdan, B. Daining, and S. Weidong, "Continuous mobile authentication using touchscreen gestures," in *Proceedings of the 12th IEEE Conference on Technologies for Homeland Security*, ser. HST'12, 2012.

[13] N. Sae-Bae, K. Ahmed, K. Isbister, and N. Memon, "Biometric-rich gestures: a novel approach to authentication on multi-touch devices," in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 977–986. [Online]. Available: http://doi.acm.org/10.1145/2208516.2208543

[14] N. Sae-Bae, N. Memon, and K. Isbister, "Investigating multi-touch gestures as a novel biometric modality," in *Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*, 2012, pp. 156–161.

[15] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch me once and i know it's you!: implicit authentication based on touch screen patterns," in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 987–996. [Online]. Available: http://doi.acm.org/10.1145/2208516.2208544

[16] D. Stefan, X. Shu, and D. D. Yao, "Robustness of Keystroke-Dynamics based biometrics against synthetic forgeries," *Computers & Security*, Oct. 2011.

[17] S. Aivils, "Ubuntu Manuals," http://manpages.ubuntu.com/manpages/gutsy/man8/xsendkeycode.8.html, March, 2005.

[18] "Microsoft developer network," msdn.microsoft.com, last accessed in Sept, 2012.

[19] L. Ballard, S. Kamara, F. Monrose, and M. K. Reiter, "Towards practical biometric key generation with randomized biometric templates," in *Proceedings of the 15th ACM conference on Computer and communications security.* New York, NY, USA: ACM, 2008, pp. 235–244.

[20] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "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*, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 162–175.

[21] "Lego mindstorms," http://mindstorms.lego.com/en-us/default.aspx, last accessed in April, 2013.

[22] A. Peacock, X. Ke, and M. Wilkerson, "Typing patterns: A key to user identification," *IEEE Security and Privacy*, vol. 2, pp. 40–47, 2004.

[23] "Gartner says worldwide pc, tablet and mobile phone combined shipments to reach 2.4 billion units in 2013," http://www.gartner.com/newsroom/id/2408515, last accessed in April, 2013.

[24] O. Riva, C. Qin, K. Strauss, and D. Lymberopoulos, "Progressive authentication: deciding when to authenticate on mobile phones," in *Proceedings of the 21st USENIX conference on Security symposium*, ser. Security'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 15–15. [Online]. Available: http://dl.acm.org/citation.cfm?id=2362793.2362808

[25] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *Proceedings of the 4th USENIX conference on Offensive technologies*, ser. WOOT'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–7. [Online]. Available: http://dl.acm.org/citation.cfm?id=1925004.1925009

[26] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "Accessory: password inference using accelerometers on smartphones," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '12. New York, NY, USA: ACM, 2012, pp. 9:1–9:6. [Online]. Available: http://doi.acm.org/10.1145/2162081.2162095

[27] M. Jakobsson, E. Shi, P. Golle, and R. Chow, "Implicit authentication for mobile devices," in *Proceedings of the 4th USENIX conference on Hot topics in security*, ser. HotSec'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 9–9. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855628.1855637

[28] "Darpa-baa-13-16 active authentication (aa) phase 2," https://www.fbo.gov/index?s=opportunity&mode=form&id=aa99ff477192956bd706165bda4ff7c4&tab=core&_cview=1, last accessed in April, 2013.

[29] F. George, N. Mark, and S. Raymond, "Personal attributes authentication techniques," Pattern Analysis and Recognition Corp Rome NY, East Lansing, Michigan, Tech. Rep. ADA047645, February 1977.

[30] S. Gaines, W. Lisowski, J. Press, and N. Shapiro, "Authentication by keystroke timing," RAND Corporation, East Lansing, Michigan, Tech. Rep. R-2526-NSF, February 1980.

[31] K. Killourhy, "A scientific understanding of keystroke dynamics," Ph.D. dissertation, Carnegie Mellon University, 2012.

[32] C. R. Romain Giot, Mohamad El-Abed, "Keystroke dynamics with low constraints svm based passphrase enrollment," in *IEEE Third International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, 2009, pp. 1–6.

[33] R. Zack, C. Tappert, and S.-H. Cha, "Performance of a long-text-input keystroke biometric authentication system using an improved k-nearest-neighbor classification method," in *IEEE Fourth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, 2010, pp. 1–6.

[34] M. Hossain, K. Balagani, and V. Phoha, "New impostor score based rejection methods for continuous keystroke verification with weak templates," in *IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, 2012, pp. 251–258.

[35] A. Serwadda, Z. Wang, P. Koch, S. Govindarajan, R. Pokala, A. Goodkind, D.-G. Brizan, A. Rosenberg, V. Phoha, and K. Balagani, "Scan-based evaluation of continuous keystroke authentication systems," *IT Professional*, vol. 15, no. 4, pp. 20–23, 2013.

[36] K. A. Rahman, K. S. Balagani, and V. V. Phoha, "Snoop-forge-replay attacks on continuous verification with keystrokes," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 528–541, 2013.

[37] T. C. Meng, P. Gupta, and D. Gao, "I can be you: Questioning the use of keystroke dynamics as a biometric," in *NDSS,2013*, Feb 2013.

[38] L. Ballard, D. Lopresti, S. Member, and F. Monrose, "Forgery quality and its implications for behavioral biometric security," *IEEE Transactions on Systems, Man and Cybernetics (Special Edition)*, 2007.

[39] L. Ballard, D. Lopresti, and F. Monrose, "Evaluating the security of handwriting biometrics," in *In The 10 th International Workshop on the Foundations of Handwriting Recognition*, 2006, pp. 461–466.

[40] T. Sim and R. Janakiraman, "Are Digraphs Good for Free-Text Keystroke Dynamics?" in *IEEE Conference on Computer Vision and Pattern Recognition*, jun 2007, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2007.383393

[41] K. Killourhy and R. Maxion, "Should security researchers experiment more and draw more inferences?" in *4th Workshop on Security Experimentation and Test (CSET-11)*, 2011.

[42] J. Ruska, "Most Common Passwords lists from 3 Databases," http://blog.jimmyr.com/Password_analysis_of_databases_that_were_hacked_28_2009.php, Feb, 2009.

[43] F. Bergadano, D. Gunetti, and C. Picardi, "User authentication through keystroke dynamics," *ACM Transactions on Information and System Security*, vol. 5, pp. 367–397, November 2002.

[44] J. Shrijit, "Naive bayes and similarity based methods for identifying computer users using keystroke patterns," Ph.D. dissertation, Louisiana Tech University, 2009.

[45] H. Lilliefors, "On the Kolmogorov-Smirnov test for normality with mean and variance unknown," *Journal of the American Statistical Association*, 1967.

[46] M. A. Stephens, "Edf statistics for goodness-of-fit: Part 1," Office of Naval Research, USA, Tech. Rep., Jan 1972.

[47] F. J. Massey, "The Kolmogorov-Smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[48] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2008, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org

[49] K.-C. Lan and J. Heidemann, "Rapid model parameterization from traffic measurements," *ACM Transactions on Modeling and Computer Simulation*, vol. 12, no. 3, pp. 201–229, Jul. 2002.

[50] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '98/PERFORMANCE '98. New York, NY, USA: ACM, 1998, pp. 151–160.

[51] M. Wilk and R. Gnanadesikan, "Probability Plotting Methods for the Analysis of Data," *Biometrica Trust*, vol. 55, no. 1, pp. 1–17, March 1968.

[52] A. Serwadda and V. V. Phoha, "Examining a large keystroke biometrics dataset for statistical-attack openings," *ACM Transactions on Information and System Security.*, vol. 16, no. 2, pp. 8:1–8:30, sep 2013.

[53] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on ssh," in *Proceedings of the 10th conference on USENIX Security Symposium - Volume 10*. Berkeley, CA, USA: USENIX Association, 2001, pp. 25–25. [Online]. Available: http://portal.acm.org/citation.cfm?id=1267612.1267637

[54] R. D. Gibbons, *Nonparametric Statistical Inference*, 2nd ed. M. Dekker, 1985.

[55] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, ser. VLDB '04. VLDB Endowment, 2004, pp. 180–191. [Online]. Available: http://dl.acm.org/citation.cfm?id=1316689.1316707

[56] N. A. Weiss, *Introductory Statistics*, 5th ed. Addison Wesley, 1999.

[57] M. K. McDougall and G. D. Rayner, "Robustness to non-normality of various tests for the one-sample location problem," *JAMDS*, vol. 8, no. 4, pp. 235–246, 2004.

[58] P. D. Doane and E. L. Seward, *Applied Statistics in Business and Economics*, 1st ed. McGraw-Hill, 2007.

[59] E. Whitley and J. Ball, "Statistics review 6: Nonparametric methods." *Critical care (London, England)*, vol. 6, no. 6, pp. 509–513, Dec. 2002. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/12493072

[60] M. G. Bulmer, *Principles of Statistics*, 2nd ed. Dover, 1979.

[61] E. Yu and S. Cho, "Ga-svm wrapper approach for feature subset selection in keystroke dynamics identity verification," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 3, july 2003, pp. 2253 – 2257 vol.3.

[62] M. Scutari and A. Brogini, "Bayesian network structure learning with permutation tests," in *Statistics for Complex Problems: the Multivariate Permutation Approach and Related Topics*, Mar. 2011.

[63] D. Franois, V. Wertz, and M. Verleysen, "The permutation test for feature selection by mutual information," in *European Symposium on Artificial Neural Networks*, ser. ESANN 2006, 2006, pp. 239–244.

[64] L. Ballard, "Robust techniques for evaluating biometric cryptographic key generators," Ph.D. dissertation, John Hopkins University, 2008.

[65] L. Ballard, F. Monrose, and D. Lopresti, "Biometric authentication revisited: understanding the impact of wolves in sheep's clothing," in *Proceedings of the 15th conference on USENIX Security Symposium - Volume 15*. Berkeley, CA, USA: USENIX Association, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267336.1267339

[66] R. Gummadi, H. Balakrishnan, P. Maniatis, and S. Ratnasamy, "Not-a-bot (nab): Improving service availability in the face of botnet attacks," in *NSDI 2009*, Boston, MA, April 2009.

[67] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.

[68] M. Marseguerra, E. Zio, L. Podofillini, and D. W. Coit, "Optimal design of reliable network systems in presence of uncertainty," *IEEE Transactions on Reliability*, vol. 54, no. 2, pp. 243–253, 2005.

[69] L. Ballard, D. Lopresti, and F. Monrose, "Forgery quality and its implications for behavioral biometric security," *Trans. Sys. Man Cyber. Part B*, vol. 37, no. 5, pp. 1107–1118, Oct. 2007. [Online]. Available: http://dx.doi.org/10.1109/TSMCB.2007.903539

[70] "Touchalytics," http://www.mariofrank.net/touchalytics/, last accessed in April, 2013.

[71] L. Ballard, S. Kamara, F. Monrose, and M. K. Reiter, "Towards practical biometric key generation with randomized biometric templates," in *Proceedings of the 15th ACM conference on Computer and communications security*, ser. CCS '08. New York, NY, USA: ACM, 2008, pp. 235–244. [Online]. Available: http://doi.acm.org/10.1145/1455770.1455801

[72] J.-J. Cabibihan, "Patient-specific prosthetic fingers by remote collaboration - a case study," *CoRR*, vol. abs/1105.1028, 2011.

[73] T. Walsh, *Timeless Toys: Classic Toys and the Playmakers Who Created Them.* McMeel Publishing, 2005.

[74] "A gloves. original touch screen gloves," http://www.amazon.com/Agloves-Original-Touchscreen-Gloves-Texting/dp/B005GXMM5W, last accessed in April, 2013.

[75] "Lego robot solves sudoku puzzles," http://news.cnet.com/8301-17938_105-10318953-1.html, last accessed in Sept, 2013.

[76] "Robots lab," http://www.robotslab.com/#gsc.tab=0, last accessed in Sept, 2013.

[77] A. Serwadda and V. V. Phoha, "When kids' toys breach mobile phone security," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 599–610. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516659

[78] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: http://dx.doi.org/10.1023/A:1022627411411

[79] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theor.*, vol. 13, no. 1, pp. 21–27, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1967.1053964

[80] A. F. Martin, G. R. Doddington, T. Kamm, M. Ordowski, and M. A. Przybocki, "The det curve in assessment of detection task performance," in *Fifth European Conference on Speech Communication and Technology*, ser. EUROSPEECH '97. ISCA, 1997.

[81] A. Serwadda, V. V. Phoha, and Z. Wang, "Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms," in *IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, 2013, pp. 1–8.