2013

A Study of the Imitation, Collection and Usability Issues of Keystroke Biometrics

Chee Meng TEY
Singapore Management University, cmtey.2008@smu.edu.sg

A study of the imitation, collection and usability issues of keystroke biometrics

by

TEY Chee Meng

Submitted to School of Information Systems in partial fulfilment of the requirements for the Degree of Doctor of Philosophy in Information Systems

Dissertation Committee:

Debin GAO (Supervisor/Chair)
Assistant Professor of Information Systems
Singapore Management University

Xuhua DING (Co-Supervisor)
Associate Professor of Information Systems
Singapore Management University

Robert DENG Huijie
Professor of Information Systems
Singapore Management University

Jianying ZHOU
Senior Scientist
Institute for Infocomm Research

Singapore Management University

2013

Abstract

The majority of authentication systems used today involves passwords, where a user is required to remember and key in the correct password to login. Keystroke biometrics is an alternative approach whereby users are identified by one or more features such as (a) the timing between keystrokes, (b) how long users hold each key and (c) how hard users press each key. It is being assumed in prior research that the way one user types a password/word is different from the way another user types the same password and this characteristic remains stable over time. Existing literature however left open three questions that are material to the security and usability of keystroke biometrics.

The first concerns the uniqueness property. Keystroke biometrics is a form of behaviour biometrics. Behaviour can be conditioned by training. An important question is whether one person's typing pattern can be changed, through appropriate training to resemble that of another if the typing pattern of the latter is known. If the answer is positive, the false acceptance rate of keystroke biometrics would increase to unacceptable levels.

A natural second question which follows from the first, questions the extent to which typing patterns can be kept secret. Attackers who are able to log the keystrokes can create a model of the victim's typing pattern. We ask whether there exists more convenient and less intrusive ways to collect typing patterns. The availability of typing pattern raise both security and privacy concerns because it is a prerequisite for (a) achieving the best outcome in timing side channel attacks and (b) imitation attacks on keystroke biometrics. It can also be used to identify users.

The last question concerns the stability of typing patterns. Given that the environment, as well as the physical and mental conditions of users changes throughout the day, we ask whether collected typing patterns remain stable under different conditions. For example, to what extent do the typing patterns change with the mood,

muscle strain such as after exercise, posture, type of keyboard and even lighting? If typing patterns are not resistant to these conditions, it may result in a higher false rejection rate (FRR). High FRR leads to usability problems. Usability issues are an important aspect of security, because poor usability motivates users to find shortcuts that bypass the system. It also creates a pressure to tune the system in ways that lower security.

In this thesis, we answer all three questions. We found that by providing a novel feedback and training interface, it is possible for one person to imitate another through incremental adjustment of typing pattern. We show that even for targets whose typing patterns are only partially known, imitation training allows attackers to defeat one of the best anomaly detection engines using keystroke biometrics. For a group of 84 participants playing the role of attackers and 2 eight-character passwords of different difficulty, the false acceptance rate (FAR) of the easy and difficult password increases from 0.24 and 0.20 respectively before training, to 0.63 and 0.42 respectively after training. With full information, the FAR increases to 0.99 for both passwords for the 14 best attackers.

To explore the feasibility of collecting typing patterns, we focus on interactivity rich JavaScript applications. The Google Suggestions service used in Google Search is one example of an interactivity rich JavaScript application. We analysed the timing side channel of Google Suggestions by reverse engineering the communication model from obfuscated JavaScript code. The goal is to determine the extent to which an attacker can infer the typing pattern of a victim. From our experiments involving 11 participants, we found that for each keypair with at least 20 samples, the mean of the inter-keystroke timing can be determined with an error of less than 20%.

For the usability problem, we show that the FRR of keystroke biometrics changes for the worse under a range of common conditions such as background music, exercise and even game playing. In a user study involving 111 participants, the average penalties (increases) in FRR are 0.0360 and 0.0498, respectively, for

two different classifiers. We also show that not everyone is suitable for keystroke biometrics deployment, which is exacerbated by the susceptibility to external influences. For example, using a Monte Carlo simulation, we found that 30% of users would encounter an account lockout before their 50th authentication session (for a lockout policy of 3 attempts) if they are affected by external influences 50% of the time when authenticating.

Contents

List of Figures

List of Tables

Acknowledgements

This thesis would not have been possible without the support of many people. I am very grateful to my adviser, Debin Gao, who guided and helped me with resources, time, ideas and in many other ways. I am also grateful to my co-supervisor Xuhua Ding, committee members Robert Deng, and Jianying Zhou, for their guidance, comments and hard questions.

The school, as a community, has a stimulating culture of mutual help. In the course of my study, I have benefited from the help of Lau Hoong Chuin, Pang Hwee Hwa, Ma Dan, Li Yingjiu, Jiang Lingxiao, Lim Ee Peng, Payas Gupta, Han Jin, Yan Qiang, Kartik Muralidharan, Ong Chew Hong, Seow Pei Huan and many others. Steve Miller and Koh Noi Sian have been inspiring in sharing their philosophy about the PhD journey. Tan Swee Liang and Grace Cheong have been enlightening with their passion and knowledge in the art of teaching.

My colleagues in DSO National Laboratories supported and encouraged me in my PhD journey. Their ideas, knowledge sharing and criticism helped expand my understanding of both technical and philosophical issues.

A PhD journey is meaningful because life itself is meaningful. What is the meaning of life? It is family. For my family, I offer my greatest thanks and appreciation.

To my loving family.

Chapter 1

Introduction

Biometrics is the oldest form of authentication; people verify each other on telephone based on voice, humans recognise each other by face when they meet. There is a wide variety of candidate characteristics, including facial features, speech patterns, hand geometry [28], fingerprints, iris scans, DNA, typing patterns, signature geometry[1] and mouse dynamics. Biometrics can be classified into two major categories: 1) physiological biometric – a biometric that is based on a physical trait of an individual, e.g., facial features, hand geometry, fingerprints, iris scans, and DNA; and 2) behavioural biometric – a biometric that is based on the behavioural trait of an individual, e.g., speech patterns, typing patterns, signatures, and mouse dynamics.

A central issue of biometrics security concerns the uniqueness of the biometrics feature. In the context of fingerprint biometrics for example, there had been cases where suspects were wrongly identified through fingerprints [31]. For keystroke biometrics, prior literature had shown that although typing patterns between individuals do overlap, and misidentification is possible as in fingerprinting, the error rates are low enough such that typing patterns can be considered unique to each individual [25, 39, 50, 43, 42].

Another important issue of biometrics security is verifying if the authentication

[1] Signature geometry encompasses not just the look of the signature, but also possibly the pen pressure, signature speed, etc.

data came directly from the owner. For example, in the case of fingerprint biometrics, if such verification is absent, arbitrary fingerprints may be forged. Geller et al. demonstrated how fingerprints can be forged in a forensic context [33]. Boatwright et al. cited an instance where gelatine created fingerprints were used to gain unauthorized access [28]. Likewise, for keystroke biometrics, if such verification is absent, an automated system may be used to deliver the desired typing pattern to the detection engine.

1.1 Uniqueness of typing patterns

In this thesis, we study and question the uniqueness property of keystroke biometrics in static mode (see Chapter 3). Keystroke biometrics deployment can be divided into static and continuous mode. For the former, the typing pattern is checked once typically during login. For the latter, the typing pattern is checked continuously, as the user is working on their documents.

We consider the scenario where attackers are shown the typing pattern of their victims and make a conscious attempt to imitate. Scenarios where the typing pattern may be known include

1. An attacker captures samples of the victim's password typing.

2. Information in the biometrics database was leaked.

If imitation is possible, the error rates of detection engines would become unacceptably high. This means keystroke dynamics would be unsuitable for use as a biometrics feature. The existing commercial security solutions using keystroke biometrics [14, 2, 19, 16, 13, 12, 7, 21, 3, 6, 9, 20, 8, 5] can therefore be attacked.

The majority of literature in this area focused on finding a detection algorithm that best separates the legitimate users from imposters. The work by Rundhaug et al. [45] is the most similar to our work (see Section 2.7). Based on 21 participants, they showed that the provision of feedback shortens the distance between

the attacker and victim's typing pattern by 9.7%. While differences can be reduced, Rundhaug et al. suggested that an attack remains difficult and proposed larger scale experiments to verify the feasibility of such attacks. In this thesis, we investigate whether it is possible to imitate someone else's keystroke typing if appropriate feedback is provided.

We propose a novel feedback interface Mimesis with the following design goals:

1. The information must be easy to understand with minimal cognitive load required. The latter is for the attackers to focus on their imitation task.

2. The interface should provide specific tips on particular aspects to improve on.

3. Both positive and negative feedback should be provided to the attacker so that she can repeatedly make minor adjustments to her typing pattern to imitate better.

The proposed experiments involve a group of 84 participants playing the role of attackers against one of the best keystroke biometrics based authentication systems by Araujo et al. [25] (based on the evaluation by Killourhy and Maxion [40]). We evaluate the effectiveness of Mimesis and investigate whether there exists individuals who can adjust their typing pattern to imitate someone else. We investigate this attack using two scenarios:

1. When the attacker only has an incomplete model of the victim's typing pattern, such as when only a limited number of victim typing samples are available to infer the model

2. When the attacker has complete information.

We found that by providing a novel feedback and training interface, it is possible for one person to imitate another through incremental adjustment of typing pattern. We show that even for targets whose typing patterns are only partially known, imitation training allows attackers to defeat one of the best anomaly detection engines

using keystroke biometrics. For a group of 84 participants playing the role of attackers and 2 eight-character passwords of different difficulty, the false acceptance rate (FAR) of the easy and difficult password increases from 0.24 and 0.20 respectively before training, to 0.63 and 0.42 respectively after training. With full information, the FAR increases to 0.99 for both passwords for the 14 best attackers. The details of this work is shown in Chapter 3.

1.2 Secrecy of typing patterns

In Chapter 4, we investigate the leakage of information about typing patterns from JavaScript applications. In particular, we explore whether it is possible to derive the continuous mode typing pattern of a victim from their web traffic. Rich and complex JavaScript applications provide sophisticated GUI updates and fast client- server communications that approach the capabilities of traditional desktop applications. Examples include Google Instant [1] and Google Suggestion [4] (see Figure 1.1) which allows the user to view the results/suggestions on–the–fly while typing search queries. To achieve such interactivity, the frontend JavaScript communicates using HTTP with the backend server very frequently in response to various events such as keypress. Table 1.1 shows the typical series of HTTP requests in Google Suggestions, when a user types in the search term 'secure'. If certain characteristics of the traffic is correlated with the content, then the improved user experience comes with a price in the form of side channel information leakages. Prior research by Chen et al. on encrypted packet sizes has demonstrated that such leakages exists [29].

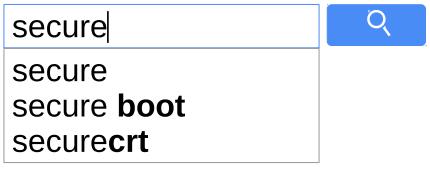


Figure 1.1: Suggestions for 'secure'

Chapter 4 studies the timing side channel of Google Suggestions. We hope a

(a)

```
GET /s?...&q=s&
GET /s?...&q=se&
GET /s?...&q=sec&
GET /s?...&q=secu&
GET /s?...&q=secur&
GET /s?...&q=secure&
```

(b)

```
GET /s?...&q=s&
GET /s?...&q=se&
GET /s?...&q=sev&
GET /s?...&q=se&
GET /s?...&q=sec&
GET /s?...&q=secu&
GET /s?...&q=secur&
GET /s?...&q=secure&
```

(c)

```
GET /s?...&q=s&
GET /s?...&q=se&
GET /s?...&q=sec&
    long pause
GET /s?...&q=secure&
```

(d)

```
GET /s?...&q=sec&
GET /s?...&q=secur&
GET /s?...&q=secure&
```

Table 1.1: Query scenarios: (a) Slow typing. (b) Correcting a typing error. (c) Typing only s, e, c, followed by choosing $secure$ from the suggestions. (d) Fast typing. In this thesis, we do not handle case (d).

detailed study of one application yields insights on the threats that may apply to the class of such applications as a whole. We want to know whether traditional threats apply and if new ones arise. Our work differs from prior work in 3 ways:

1. It is the first to analyze JavaScript timing side channels,
2. Our attack model includes inference of typing pattern, and
3. The correlation between keystroke and traffic timing is weaker compared to similar analysis on SSH binaries [47].

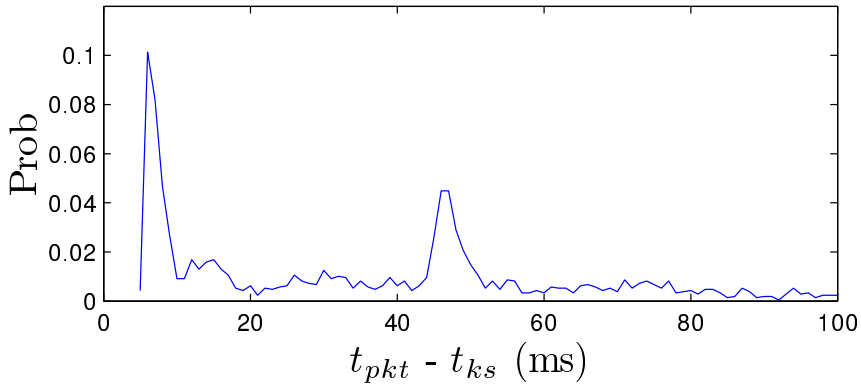


Figure 1.2: Initial investigation of $t_{pkt} - t_{ks}$. t_{ks} refers to the time the keystroke is typed, while t_{pkt} refers to the time the corresponding query appears on the network.

Correlation is weaker because JavaScript applications are far slower than native binary applications. The interaction with the browser via the Document Object

Model is more complex. The IO intensive tasks of JavaScript applications (such as UI updates) take longer than the computation intensive task of SSH. Lastly, JavaScript applications typically run in a single threaded co-operative multitasking execution model. Task execution is deferred when the scripting engine is busy with another task. Evidence of weak correlation can be seen in Figure 1.2, where there is a variable delay between keypress and Google Suggestions traffic. Also, in Figure 1.3, there is an obvious difference between the bell shaped inter-keystroke timing distribution compared to the exponential shaped inter-packet timing distribution.

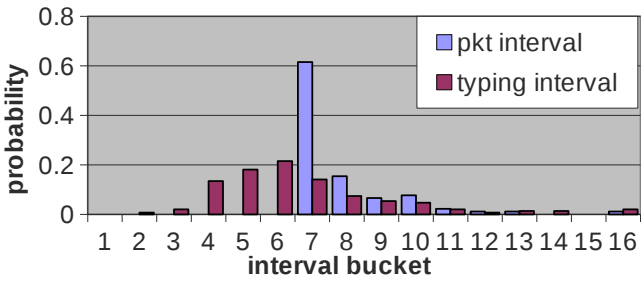


Figure 1.3: The difference between the probability distribution of the inter-keystroke timing and the corresponding inter-packet timing for the keypair 'in' of 1 participant.

Inference of typing pattern raises privacy concerns. Prior research showed that natural typing patterns are unique enough for user identification [25, 39, 43, 42]. Natural typing occurs when no attempts are made to consciously alter the typing. There are also security concerns because firstly, knowledge of typing patterns facilitates attacks on other timing side channels. This was demonstrated by Song et al. where personalized typing patterns gave the best outcome [47]. Secondly, once a victim's typing pattern is known, imitation attacks may become possible. Chapter 3 showed that imitation attacks on static mode keystroke biometrics systems are feasible. If the same is true of continuous mode, then leakage of the typing pattern results in a serious security problem.

In the following discussion, *keypair timing model* is the set of probability distributions (of timing differences) for all pairs of keystrokes; *packet-pair timing model*

is the set of probability distributions of the corresponding packets. We studied the communication model of Google Suggestions and derive a set of techniques to construct:

1. The keypair timing model from packets of unencrypted Google Suggestions traffic picked up by network sniffers or proxy logs;

2. The packet-pair timing model from the keypair timing model. The latter is useful when we need to reconstruct packets for Monte Carlo simulations.

To verify the effectiveness of these methods, we conducted a user study on 11 participants to collect their keystrokes and their timings. We found that for each keypair with at least 20 samples, the mean of the inter-keystroke timing can be determined with an error of less than 20%.

1.3 Stability of typing patterns

In Chapter 5, we investigate the extent to which static mode typing patterns remain stable under different environments as well as under different physical and emotional conditions. For example, computer gaming and exercising are 2 common activities. Gaming involves both muscle strain and emotional changes (due to winning and losing). Exercise also involves muscle strain and emotional change, but the emotional change is positive due to the release of mood enhancing endorphins. People's mood changes according to the type of video they watch. Muscle strain has a direct effect on the muscle control involved in typing, while prior studies [32] have also shown that people type faster when their mood is positive. However, what is not known is the quantitative extent to which the FRR (false rejection rate) of keystroke biometrics is affected. As computer gaming and exercise are common activities, a significant change in FRR may result in repeated logon failure.

More specifically, we want to evaluate:

1. What is the change in the FRR (false rejection rate) of 2 different classifiers before and after each of the above activities?

2. How long does it take for the FRR to go back to normal?

3. If during the recovery, there are some typing activities, does it delay the recovery?

4. Are there any difference on the FRR change between hardcore gamers and light gamers?

5. Is there any difference in the FRR when the success/failure of the logon attempt is made known to the user as compared to the case where it is not known? Existing evaluations of keystroke biometrics collected typing patterns without feedback. If the FRR is affected by the feedback, then existing methodologies need to be revised to take into account this factor.

We are also interested in whether environmental factors affect typing patterns. The identification of environments where typing patterns are affected allow the compilation of a black list where keystroke biometrics deployment is not advisable or may provide insights into mitigation measures. We are also interested in whether the provision of logon success/failure feedback provides any mitigation, and if yes, the extent of the mitigation. We are interested in the following factors:

1. Background rhythm/noise.

2. Low lighting (e.g. during blackout)

3. Time pressure: user needs to unlock keyboard under urgent circumstances

4. Minor injury: simulated by putting a plaster on the index finger

5. Standing

6. Change of keyboard (notebook keyboard to standard external keyboard, notebook keyboard to ergonomic external keyboard)

The collection methods in the existing literature differ in their attempts to account for the abovementioned factors. For example, Killourhy and Maxion collected 400 typing samples over 8 sessions, with a minimum interval of 1 day between each session so as to 'capture the day to day variation' [40], while Revett collected all samples in a single sitting [44]. However, even for the former case, it is not known how much of the variation is due to which external factor. In this thesis, we fill in the missing gap by providing a quantitative measure of the effects of various conditions on the typing pattern.

We show that the FRR of keystroke biometrics changes for the worse under a range of common conditions such as background music, exercise and even game playing. In a user study involving 111 participants, the average penalties (increases) in FRR are 0.0360 and 0.0498, respectively, for two different classifiers. We also show that not everyone is suitable for keystroke biometrics deployment, which is exacerbated by the susceptibility to external influences. For example, using a Monte Carlo simulation, we found that 30% of users would encounter an account lockout before their 50th authentication session (for a lockout policy of 3 attempts) if they are affected by external influences 50% of the time when authenticating.

1.4 Contributions

In this thesis, we argue that the use of keystroke dynamics as a form of biometrics is questionable because it is not unique, typing patterns can be easily leaked and are unstable over a range of external conditions.

More specifically, we found deficiencies in these 3 areas:

1. Uniqueness. It is possible for one person to imitate another through incremental adjustment of typing pattern. We show that even for targets whose typing patterns are only partially known, imitation training allows attackers to defeat one of the best anomaly detection engines using keystroke biometrics.

2. Secrecy. It is possible for a person's typing pattern to leak via the timing side channel of interactivity rich JavaScript application.

3. Stability. The FRR of keystroke biometrics changes for the worse under a range of common conditions such as background music, exercise and even game playing.

1.5 Structure of thesis

The rest of this thesis is organised as follows: Chapter 2 provides general background information on keystroke biometrics authentication systems as well as related works. Chapter 3 investigates the uniqueness property of keystroke biometrics. In Chapter 4, we investigate the leakage of information about typing patterns from JavaScript applications. In Chapter 5, we investigate the extent to which static mode typing patterns remain stable under different environments as well as under different physical and emotional conditions. We conclude the findings of this thesis in Chapter 6.

1.6 Source of publication

The material in Chapter 3 was originally published as a conference paper in the Network and Distributed System Security Symposium 2013 (NDSS Symposium 2013) [48]. The material in Chapter 4 was originally published as a conference paper in the 11th International Conference on Applied Cryptography and Network Security (ACNS 2013) [49]. The material in Chapter 5 was drawn from a project conducted at the School of Information Systems, Singapore Management University. The members of the project include Payas Gupta, Kartik Muralidharan, Debin Gao and the author.

Chapter 2

Background and related works

In this chapter, we describe the commonly followed procedures in the evaluation of a keystroke biometrics authentication system followed by the related works that are closely related to our investigations.

We first describe the collection of timing information in Section 2.1. This is followed by a description of how the timing information are organized into vectors for subsequent processing (see Section 2.2). Finally, in Section 2.3, we describe how classifiers used in keystroke biometrics compute anomaly scores and thresholds from the timing vectors.

2.1 Choice of timing information

Keystroke dynamics refer to information about the typing pattern. For example, pressing and releasing of a keystroke pair (k_a, k_b) results in 4 timings which are of interest to keystroke biometrics systems: (a) key-down time of k_a: $t^{\downarrow}_{k_a}$, (b) key-up time of k_a: $t^{\uparrow}_{k_a}$, (c) key-down time of k_b: $t^{\downarrow}_{k_b}$ and (d) key-up time of k_b: $t^{\uparrow}_{k_b}$

From these absolute time measurements, four relative timings can be derived:

- an inter-keystroke timing between k_a and k_b:
  $I_{k_a,k_b} = t^{\downarrow}_{k_b} - t^{\downarrow}_{k_a}$.
- hold timing of k_a: $H_{k_a} = t^{\uparrow}_{k_a} - t^{\downarrow}_{k_a}$

- hold timing of k_b: $H_{k_b} = t^{\uparrow}_{k_b} - t^{\downarrow}_{k_b}$
- a key up-down timing between k_a and k_b:
  $U_{k_a,k_b} = t^{\downarrow}_{k_b} - t^{\uparrow}_{k_a}$

Certain papers also include trigraphs [44]. Given 3 consecutive keys k_a, k_b and k_c, the trigraph T_{k_a,k_c} is equal to $t^{\downarrow}_{k_c} - t^{\downarrow}_{k_a}$.

Different anomaly detectors differ in the way timing information were collected and used. These differences can be categorized as follows:

1. Strings to collect. Certain papers collect timing information only for passwords or password-like strings [25, 40]. Others collect for both userids and passwords [44]. Yet others collect timing information for userids, passwords, first name and last name [35].

2. I, H, U and T. Different papers use different combinations. For example,

   (a) I, H and U [25]

   (b) only I [35, 25],

   (c) only H [25],

   (d) only U [25],

   (e) I and H [25],

   (f) H and U [30, 25],

   (g) I and U [25].

   (h) I, H and T [44],

3. Terminating carriage return. Some papers include the timing information of the terminating carriage return at the end of the password [40] while others ignore it [44].

In this thesis, due to differences in experimental design, Chapter 3 collects the timing information only for passwords and only uses I and H. On the other hand,

Chapter 5 collects timing information for both userids and passwords and uses I, H and T. Terminating carriage returns are ignored in this thesis.

2.2 Data vectorization

In the literature, timing information is typically stored as vectors. However, there are differences in the construction of these vectors. E.g. given n authentication samples, each of which yields l timing components, Haider et al. stored in each vector, the information for each timing component, resulting in l vectors of length n [34]. Araujo et al. on the other hand, stored in each vector the timing components of each sample, resulting in n vectors of length l. In this thesis, we follow the latter convention. Table 2.1 shows an example of the vectors collected when the string(s) collected in each sample yielded α inter-keystroke timings and β hold timings (such that $l = \alpha + \beta$). Each row corresponds to one vector. $t_{i,j}$ denotes the jth component of the ith vector.

Sample	Inter-Keystroke Time			Hold Time		
# 1	$t_{1,1}$	$\ldots$	$t_{1,\alpha}$	$t_{1,\alpha+1}$	$\ldots$	$t_{1,\alpha+\beta}$
# 2	$t_{2,1}$	$\ldots$	$t_{2,\alpha}$	$t_{2,\alpha+1}$	$\ldots$	$t_{2,\alpha+\beta}$
$\vdots$	$\vdots$					
# n	$t_{n,1}$	$\ldots$	$t_{n,\alpha}$	$t_{n,\alpha+1}$	$\ldots$	$t_{n,\alpha+\beta}$

Table 2.1: Example of data vectorization

The collected vectors are typically divided into 4 sets when evaluating a keystroke biometrics system. For each user of the system, one set of normal timing vectors from that user and one set of anomalous timing vectors are used for training. One additional set each of normal and anomalous timing vectors are used for testing. In an experimental setting, the set of anomalous timing vectors for each user is typically constructed from the normal timing vectors of all other users in the same authentication system.

2.3 Anomaly detector training and scoring

Once the training data set is collected, the next step is to train the anomaly detector. The purpose of training is to find parameters for the detector corresponding to the particular set of training data. Detectors differ in the choice of parameters. For example, in the papers of Joyce et al. and Cho et al. [35, 30] only the mean vector is needed, whereas Araujo et al. requires both a mean vector and an absolute deviation vector [25]. In comparison, Revett [44] proposed a bioinformatics based classifier which requires a set of bioinformatics related parameters. Once the parameters are determined, a detector can compute an anomaly score for each test vector. In this section, we describe the computation of parameters needed for:

1. A Euclidean distance based classifier.
2. A Manhattan (scaled) distance based classifier.
3. A bioinformatics based classifier.

2.3.1 Euclidean distance based classifier

The Euclidean classifier requires a mean vector $\bar{\boldsymbol{x}}$. Each element in $\bar{\boldsymbol{x}}$ is computed using the formula:

$$\bar{x}_j = \frac{\sum_{i=1}^{n} t_{i,j}}{n}$$

Given a test vector $\boldsymbol{y}$, the anomaly score s is computed using:

$$s = \sqrt{\sum_{j=1}^{l} (y_j - \bar{x}_j)^2}$$

2.3.2 Manhattan (scaled) distance based classifier

The Manhattan (scaled) distance classifier requires the mean vector as well as an absolute deviation vector $\bar{\boldsymbol{d}}$. Each element in $\bar{\boldsymbol{d}}$ is computed using the formula:

$$d_j = \frac{\sum_{i=1}^{n} |t_{i,j} - \bar{x}_j|}{n-1}$$

Given a test vector $\boldsymbol{y}$, the anomaly score s is computed using:

$$s = \sum_{j=1}^{l} \frac{|y_j - \bar{x}_j|}{d_j}$$

2.4 Bioinformatics-based classifier

The bioinformatics-based classifier [44] computes a motif vector using the following steps:

1. Computing a max vector $\boldsymbol{m}$

$$m_j = \max_i t_{i,j}$$

2. Mapping each vector in the training set $\boldsymbol{t}$ to a normalized vector $\boldsymbol{u}$ such that:

$$u_{i,j} = \frac{t_{i,j}}{m_{i,j}}$$

3. Mapping each normalized vector $\boldsymbol{u}$ to a bin vector $\boldsymbol{b}$ such that: $b_{i,j}$:

$$b_{i,j} = \begin{cases} 0 & \text{if } 0 < u_{i,j} \leq 0.05 \\ 1 & \text{if } 0.05 < u_{i,j} \leq 0.10 \\ \dots & \\ 19 & \text{if } 0.95 < u_{i,j} \leq 1.00 \end{cases}$$

4. Computing a position specific scoring matrix P with 20 rows and l columns from the set of bin vectors such that each element p_{kj} of P is given by:

$$p_{kj} = \frac{\text{count}_i(b_{i,j} = k - 1)}{n}$$

where $1 \leq k \leq 20$ and $\text{count}_i(b_{i,j} = k - 1)$ counts for all i the number of times $b_{i,j}$ equals to $k - 1$.

5. Computing a motif vector $\boldsymbol{f}$ such that:

$$f_j = \begin{cases} k & \text{if } \exists k : p_{kj} > 0.8 \\ -1 & \text{otherwise} \end{cases}$$

6. Counting the number of positive elements in $\boldsymbol{f}$. If it is less than 22, a new max vector $\boldsymbol{m}^*$ is computed such that:

$$m_j^* = \begin{cases} m_j \times 1.1 & \text{if } f_j < 0 \\ m_j & \text{otherwise} \end{cases}$$

Steps 2 to 6 are then repeated, substituting $\boldsymbol{m}^*$ for $\boldsymbol{m}$, until the number of positive elements in $\boldsymbol{f}$ is at least 22. The last computed value of $\boldsymbol{m}$ is saved.

The values 19 (step 3), 20 (step 4), 0.8 (step 5) and 22 (step 6) are tuneable parameters chosen for our experiment in Chapter 5 because they gave reasonable classifier performance. After obtaining the motif vector, to compute the anomaly score s for a test vector $\boldsymbol{y}$, a bin vector $\boldsymbol{y}^*$ is obtained following steps 2 and 3. The score s is computed by comparing the differences between $\boldsymbol{y}^*$ and $\boldsymbol{f}$ element by

element as follows:

$$s = \sum_{j=1}^{l} D(y_j^*, f_j) \text{ where } D(\alpha, \beta) = \begin{cases} 0 & \text{if } \beta = -1 \\ 1 & \text{if } \beta \geq 0,\ \alpha = \beta \\ -1 & \text{if } \beta \geq 0,\ \alpha \neq \beta \end{cases}$$

2.5 Computation of threshold

Once an anomaly score is computed, a decision needs to be made to accept or reject the test sample. This requires a threshold parameter. For both the Euclidean and Manhattan (scaled) distance based classifiers, scores less than the threshold (close to the mean vector) are accepted. For the bioinformatics classifier, scores greater than the threshold (better match with motif vector) are accepted. In either case, the threshold defines a multidimensional boundary separating the acceptance space from the rejection space.

Setting a strict threshold means that less anomalous vectors are wrongly classified as normal. The percentage of such vectors is known as the false acceptance rate (FAR). A strict threshold, however, also means that more normal vectors are wrongly classified as anomalous. The percentage of such vectors is known as the false rejection rate (FRR). A lenient threshold on the other hand has the opposite effect: FAR increases (worse) but FRR decreases (better).

In the context of keystroke biometric based authentication, submission of a test vector that is classified as anomalous means that the authentication attempt is rejected. Conversely, if the test vector is classified as normal, the authentication attempt is accepted. The selection of the threshold therefore involves a trade-off between FAR and FRR. A common way to set the threshold is to chose it such that the FAR and FRR are equal. The value of the FAR (or FRR) at such a threshold is known as the equal error rate (EER). Once the threshold is computed, an anomaly detector becomes ready for the classification task. Typically, a set of test vectors

containing both normal and anomalous vector are used to evaluate the effectiveness of the detector.

2.6 Visualising classifier shapes

In Chapter 5, there is a need to explain changes in the FRR due to changes in typing pattern. Visualising classifier shapes provides an intuitive way to understand those changes. In this section, we describe the 2-d shapes of the classifiers used in that chapter, namely Manhattan (scaled) distance classifier and the bioinformatics based classifier.

A 2-d classical Manhattan distance boundary would have the shape of a (45° rotated) square and centred at the mean vector. That is because at the boundary, the sum of the absolute x and y distance from the mean equals to the threshold. The 2-d Manhattan (scaled) distance classifier is a rhombus centred at the mean but scaled in each dimension by the corresponding component of the absolute deviation vector. Figure 2.1 shows the shape of the Manhattan (scaled) boundary for a 2-dimensional problem.

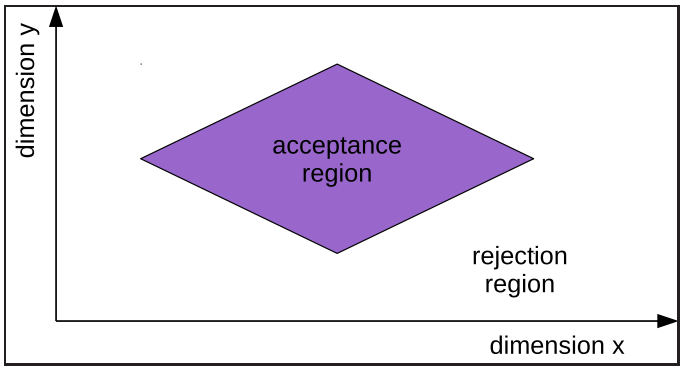


Figure 2.1: Graphical view of Manhattan (scaled) classifier

For the 2-d bioinformatics based classifier, the basic shape consists of 2 overlapping rectangles that form a thick cross (see Figure 2.2). Each rectangle corresponds to a bin encoded in the motif vector. Each bin is defined by an upper and a lower limit. If an element of the test vector is within the limits, it would be mapped to the

same bin as the motif vector, thereby adding 1 to the anomaly score. Otherwise, 1 would be subtracted from the anomaly score. The shape of the 2-d bioinformatics-based classifier boundary depends on the threshold. If the threshold is less than 0, the shape is the union of both rectangles.

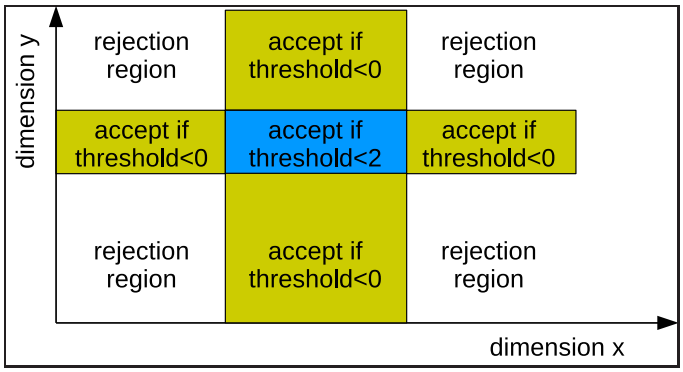


Figure 2.2: Graphical view of bioinformatics based classifier

On the other hand, if the threshold is between 0 and 2, then both elements of the test vector need to be mapped to the correct bin before the test vector is accepted. The shape is thereby the intersection of both rectangles, resulting in another rectangle. Figure 2.2 shows the shape of the bioinformatics-based classifier boundary. Note that the bin sizes for each dimension can be different due to the tuning process. Also, unlike the Manhattan (scaled) classifier, the mean of the training vectors need not necessarily be located in the middle of the intersection.

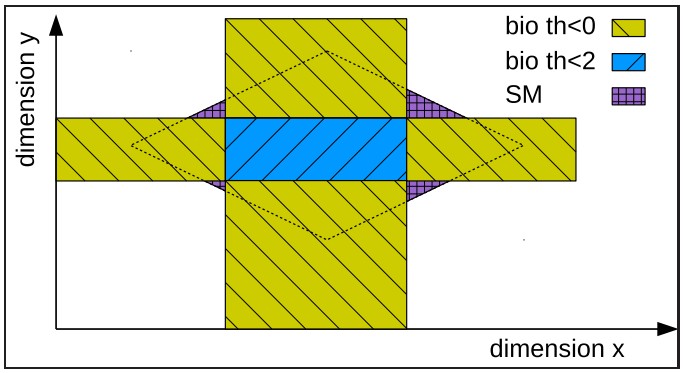


Figure 2.3: Comparison of classifier shapes

Figure 2.3 shows a possible arrangement of 2 classifiers that are trained on the same training set. Note that the centres of both shapes are not necessarily aligned.

Any changes in typing pattern can be visualized as a shift on the diagram. Depending on the direction and magnitude of the shift, the resulting sample may be accepted by one classifier and rejected by the other. The Manhattan (scaled) classifier is comparatively more vulnerable to shifts along the direction of the axes, while the bioinformatics based classifier is more vulnerable to shifts at 45° to the axes.

2.7 Related Work

In this section, we review some specific works in the literature which are closely related to the investigations in this thesis.

The work by Rundhaug et al. [45] involves imitation and is similar to ours in terms of intent and general approach. They provided imitation feedback to a team of attackers in three ways: a simple accept/reject feedback, a distance score feedback and full feedback, where the attackers are able to examine each component of their timing vectors. For each attacker, three different passwords were used, each paired with a feedback mechanism. Each attacker goes through 3 imitation sessions, one for each password. For the full feedback, a graph with 3 lines was plotted. The first line plots the victim's mean vector plus 1 standard deviation, serving as an upper boundary. The second line plots the victim's mean vector minus 1 standard deviation, serving as the lower boundary. The third line plots the attacker's timing vector. Each attacker tries to modify their typing pattern to fit their timing line within the upper and lower boundaries of the victims. The author concluded that the change in anomaly scores, before and after training, is statistically significant and imitation is therefore possible. However, they also concluded that imitation is difficult and 'can indicate that keystroke dynamics is a very secure authentication method when combined with a password'.

Cho et al. [30] and Hwang et al. [46] explored the use of artificial rhythms and cues to improve the quality of typing samples. These include:

1. Breaking up a password into multiple segments and inserting pauses in be-

tween the segments when typing.

2. The use of tune, chant or rooting to guide the password typing.

3. Minimizing the hold time in a "staccato" style [52].

4. Maximizing the hold time in a "legato" style [51].

5. Maximizing the inter-keystroke time in a "slow tempo" style.

With such cues, users produced timing vectors that are more consistent and unique.

Side channels attacks on Web applications had been studied by Chen et al. [29]. They analysed the work flow of several Web applications and modelled the flow using state machines. They showed that there is low entropy in the input to the Web applications that were investigated. Therefore, given the state machine models and packet sizes of Web traffic, it is possible for attackers to infer the content of submitted input.

Song et al [47] studied the packet size and timing side channel of SSH [53]. They found that SSH packets reveals the size of the plaintext data and the inter-keystroke timing. For the latter, Song et al. found that inter-keystroke timings can be modelled individually as normal distributions. The SSH side channel was found to leak approximately 1 bit of information per keypair. In their experiments, this translated to a reduction in search space by a factor of 50. They also found that using a personalised model of the victim's typing pattern gave better results.

Khanna et al. investigated the inference of emotion from the typing pattern [37]. They conducted a study where participants were asked to record their emotional state. Various typing features were also recorded and correlated with the emotional state. They showed that typing speed decreases under negative emotions and vice versa for positive emotions.

We are not aware of any large scale formal study that investigates the effect of environmental factors and external conditions on typing pattern. There were

however suggestions of a correlation. For example, a survey by Banerjee et al. suggested that changes in posture may influence the typing pattern [26].

Chapter 3

Can typing pattern be imitated?

3.1 Introduction

In this chapter, we question the uniqueness property of keystroke biometrics. We consider the scenario where attackers are shown the typing pattern of their victims and make a conscious attempt to imitate. If imitation is possible, the error rates of detection engines would become unacceptably high. This means keystroke dynamics would be unsuitable for use as a biometrics feature.

The rest of this chapter is organized as follows. In Section 3.2, we state the design considerations for our experiments. We provide the details of the experimental setup in Section 3.3. We describe the details of Mimesis, the interface used to provide feedback in Section 3.4, followed by an evaluation in Section 3.5. We discuss the characteristics of the attacker and the limitations of our methodology in Section 3.6. Finally, we summarise this chapter in Section 3.7.

3.2 Experimental design considerations

This chapter questions whether it is possible for one person to imitate another's typing pattern. Our approach is to provide feedback, such that the imitator can incrementally adjust her typing pattern to be closer to her target's. We also want to investigate the factors affecting the effectiveness of imitation. In this section, we

explain our experimental design considerations.

3.2.1 Choice of detector and its features

We chose the Manhattan (scaled) anomaly detector by Araujo et al. [25] (the best out of 14 anomaly detectors evaluated by Killourhy and Maxion [40]). In Chapter 2, we provided a brief description of its computation of anomaly score. Araujo et al. conducted 7 experiments based on different combinations of inter-keystroke timing (I), hold timing (H) and key up-down timing (U). U refers to the time between releasing the previous key and pressing the next key. I and H are always positive. U can sometimes be negative, because it is possible that the next key is pressed before the previous key is released. Although only 2 timings are independent[1], the best performing combination used all three timings (choice VII [25]). In our study however, we chose to use a combination of only I and H (choice V [25]), which had a FAR and FRR of 5.59% and 1.27% respectively, compared to 1.89% and 1.45% for choice VII [25].

Our reasons for excluding U are: firstly, including U increases the amount of feedback information to show in the feedback interface by about 50%. Our concern is that this may overwhelm the participants. Secondly, I and H timings are rather intuitive and participants should have little issue understanding it. U on the other hand is less intuitive and can even be negative. By excluding it, we avoid the possibility of under-performance due to poor understanding of this parameter.

3.2.2 Attack scenarios

As a prerequisite for imitation, an attacker must know the typing pattern of her victim. When designing the experiments, we considered 2 possible scenarios whereby the typing pattern may be obtained. In the first scenario, the attacker is able to extract the victim pattern from a compromised biometrics database. From the at-

[1]The inter-keystroke timing, hold timing and key up-down timing are related: $I = H + U$. Hence, any one of the three can be calculated if the other 2 are known.

tacker's point of view, this is the optimum scenario, because it allows her to build an exact replica of the detector with the victim's parameters for her training needs. In the second scenario, the attacker may be able to capture samples of the victim's keystrokes as she is authenticating (e.g. by installing a key-logger). If the attacker is able to capture a large number of samples, she would be able to get a good approximation of the victim parameters.

A question however arises when the attacker is only able to capture a relatively small number of samples. For our chosen detector, there are 2 parameters which are important to the attacker: the mean vector and the absolute deviation vector. It is possible that only one such parameter can be estimated with a small number of samples. An investigation into imitation effectiveness should therefore include an analysis of the extent to which both vectors can be approximated. If only one vector can be approximated, it is useful to measure the imitation effectiveness under such a scenario. We refer to this as the partial information scenario.

3.2.3 Motivating the participants

We consider motivation as a key factor that decides the outcome of our experiments. For that, we gave special considerations in three aspects. Firstly, the feedback interface must be designed to sustain the participant's interest. Secondly, good imitators should be rewarded for their extra efforts in the form of a performance bonus. Lastly, the duration of the experiments must strike a balance between (1) pushing the participants to try hard enough and (2) not setting it so long that it bores the participants.

Given that there will be multiple experiments, we decided that the first imitation experiment should have a fixed duration of about 30-45 minutes. For subsequent experiments, we assumed that we can identify and select those with high motivation. For these participants, the experiment is designed to be target based. That is, they will be given targets and associated rewards. There is no duration constraint: they

can leave anytime or ask for more time.

3.2.4 Basis for comparison of results

In each experiment, we need to determine how much each attacker has improved and more importantly, their chance of success for the next try if they are sent to attack a system in an actual scenario. If we used all the data in each experiment to determine the improvement, there will be a problem of underestimation in 2 aspects. Firstly, each attacker is likely to spend a good part of her time exploring and fine tuning her keystrokes. The data during this period reflects the trials and errors of each attacker's learning process, but not the outcome of the learning. Secondly, for the fixed duration experiments, boredom may set in after some point. The participant may be just "clocking" their time without trying hard.

We therefore decide that for all experiments, feedback shall include a history of their last 20 tries. Comparison of results across different experiments is also be based on the same set of data. We name the best 20 consecutive tries of each experiment the b20 data set. Our justification for the choice of 20 is that if an attacker achieves a certain target for 20 tries, even if she has only a 50% chance of repeating the feat for the next 20 tries, the probability of success for the next try is given by $\sqrt[20]{0.5} = 0.97$. It means that an attacker who has been trained before based on the b20 targets has a significant chance of success in a real life scenario.

3.2.5 Choice of password

One common problem with password based authentication systems is the prevalence of weak passwords. For example, 'password' is the top password choice. Peacock et al. considered keystroke dynamics as an effective low cost countermeasure [43]. The argument is that even if attackers guess the weak password, they cannot imitate the typing pattern. However, weak passwords tend to be easy to type. If attackers can imitate better as the password weakens, then the effectiveness of keystroke

biometrics in mitigating weak passwords is lesser than previously assumed.

For this reason, we decide to have 2 groups of participants. One group practises based on an easy password chosen to minimise finger movements on a standard US keyboard. The other group practices based on a relatively harder password chosen to maximise finger movements and therefore difficulty of typing. We also added a criterion that it must contain mixed case alphabets, at least 1 number and at least 1 punctuation to ensure compliance with the requirements of a strong password. We chose the weak and strong password as 'serndele' and 'ths.ouR2' respectively. Having two groups allows us to evaluate the effect of password typing difficulty on imitation.

3.3 Experimental setup

In this section, we describe our experimental setup[2] given the considerations in Section 3.2. We divide our investigation into 4 experiments, e1, e2, e3a and e3b. In e1, we collect the keystroke dynamics for each participant; e2 and e3a involves imitation training with only the mean vector given (the latter is a repeat of the former, but with one week interval in between); and e3b studies the effectiveness of using Manhattan (scaled) distance as the feedback. Figure 3.1 shows the experimental structure and demographics.

Timing information was collected using JavaScript, by monitoring key-down and key-up events. Although JavaScript timing measurements have a granularity of millisecond (via the Date object), the actual timing granularity is affected by the operating system. For example, Windows XP based machines have a scheduling tick quantum of approximately 16 ms. This implies that the JavaScript events which we are monitoring occur at the timing of the quantum. The timings collected on such machines therefore are in multiples of approximately 16 ms. In comparison, the tim-

[2]The experiments conducted were approved by the Institutional Review Board of the Singapore Management University (IRB approval reference IRB-12-0031-A0039). Data collected from the participants were anonymised and protected according to the procedures described in the corresponding IRB submission documents.

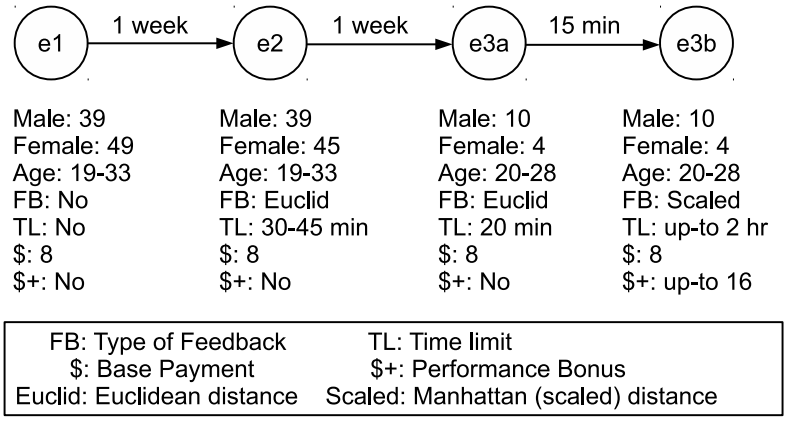


Figure 3.1: Experimental structure and demographics

ing granularity of keystroke events in the literature varies from 0.2ms [40], 1ms [25] and 10ms [47, 34]. In all the experiments, mistyped samples were discarded. All participants were paid $8 for each experiment they completed.

3.3.1 Exp `e1`: Training Data Collection

`e1` is designed based on the enrolment phase of existing keystroke dynamics based authentication systems, where each user is required to submit a certain number of samples to train the anomaly detector. Figure 3.2 shows the interface used. 88 participants took part in this experiment. This part of the study was conducted online and the participants were asked to type in the password (provided by us) in an input box via our web interface[3]. Each participant is required to type the same password 200 times without taking any break.

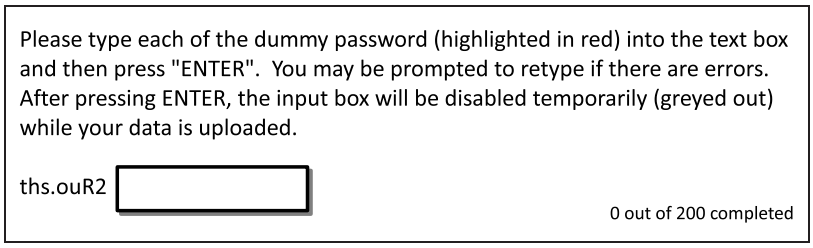


Figure 3.2: User interface for `e1`

[3]Normal network latency does not affect the accuracy of the data. This is because the keystroke timestamps were measured by the JavaScript running on the client browser. These timestamps were stored in memory. They were consolidated after the typing of each password and uploaded to the collecting server.

3.3.2 Exp e2: Imitation using Euclidean distance

In Section 3.2.2, we mentioned the need to analyst the extent to which the mean and absolute deviations vectors can be approximated by relatively few samples as well as measuring the imitation effectiveness in such a case. Based on a preliminary analysis, we found that the mean can be estimated more accurately than the absolute deviation (see Section 3.5.1 for the justification). We designed e2 to investigate the imitation effectiveness when only the mean vector is known to the attacker. Without the absolute deviation vector, the actual anomaly score for each vector cannot be calculated. The feedback therefore can only provide an approximation. We chose the Euclidean distance based anomaly score (described in Chapter 2) for this purpose.

e2 was conducted 1 week after e1. The choice of 1 week was made so that (a) participants have enough time to rest after e1 and (b) we have enough time to compute the parameters of the anomaly detectors needed for the experiment. 84 participants played the role of attackers. Ten victims were chosen randomly from among the participants of e1. Each attacker in e2 is randomly assigned a victim from the set of 10 to imitate his/her typing.

Two constraints apply to the assignment: (a) the attacker and the victim cannot be the same person, (b) the attacker and victim were assigned the same password in e1. Each attacker was given an approximate anomaly score feedback based on the Euclidean distance and required to spend at least 30 minutes. An additional 15 minutes were provided if requested. No performance bonus was offered, but the attackers were told that only the best few will be chosen for e3a and e3b (for which they will be paid up to $28). The feedback interface is more elaborated compared to e1 and is described in Section 3.4. At the end of e2, participants answered a questionnaire on the imitation experience.

Computation of the threshold for each victim requires a set of anomalous data in addition to the normal data. Following the same procedure as Killourhy and

Maxion [40], we build the anomalous data for each victim using the first 5 samples of the passwords typed from all other participants in the same category.

To help evaluate the effects of different input devices on the imitation outcome, some participants were asked to type directly on their own notebook keyboard. Others were asked to use an external keyboard provided by us.

3.3.3 Exp e3a: Additional imitation session with Euclidean distance

e3a is the second imitation experiment conducted and is very similar to e2. It was conducted 1 week after e2 to allow time for the attackers to rest and reflect, as well as for the researchers to process the data and pick the best attackers. 14 participants were chosen from the attackers of e2 using a subjective gauge of the interest level and aptitude based on (a) the enthusiasm observed during e2, (b) the number of samples submitted, (c) their response to our queries if they would like a second session with more time and (d) the improvement profile (see Figure 3.5).

The set of attacker-victim assignment, the anomaly score calculation (Euclidean distance) and the feedback interface remain unchanged. Each attacker is required to spend 20 minutes. As in e2, no performance bonus was offered. The purpose of this session is to investigate the effect an additional session has on the imitation outcome.

3.3.4 Exp e3b: Imitation using Manhattan distance

The goal of this part of the experiment is to analyze the effectiveness of keystroke imitation if an attacker is highly motivated and can obtain the full set of victim's typing pattern parameters. e3b is the final imitation experiment. It was conducted after a break of 15 minutes from e3a, so as to allow the attackers rest and refreshments.

In e3b, the interface was changed to (a) compute the anomaly score based on the Manhattan (scaled) distance, and (b) include information about the absolute

deviation vector (see Section 3.4). Two performance bonuses of equal to and double the base payment rate were offered. The first bonus is given if they can produce a consecutive run of 20 vectors all of which are scored better than their best average score in e2. The second bonus increases this difficulty by 10%. The attackers are also offered additional time up to 2 hours. All other experimental settings including the set of attackers and their demographics remain unchanged from e3a.

3.4 Mimesis

Mimesis is the feedback interface for our imitation experiments. We provide the design goals of Mimesis in Section 3.1. The design of the feedback is important because the quality of the feedback directly affects the outcome of our imitation experiments. Inadequate or inappropriate feedback may hamper the performance of the attacker. Figure 3.3 shows the Mimesis interface for the scenario where only partial information of the victim is available. We denote this interface as M_{part}. The interface for the full information scenario (denoted M_{full}) is similar. Mimesis consists of 5 components.

Top-left section This contains the password which the attacker a is trying to imitate and an input box to type in. a can also break up the password into segments and practice on each segment separately. Two buttons here provide participants with the option to hide both the tables in the central section and/or the graphical form of feedback in the bottom section.

Top-central section This contains the attack score (computed from the anomaly score using negative linear scaling and then translated to fit within 0 and 100) for a's last submitted password. It also shows the average of the recent 20 scores. For M_{part}, the scores are derived from the Euclidean distance. For M_{full}, the scores are calculated from the Manhattan (scaled) distance. The scores are only updated when

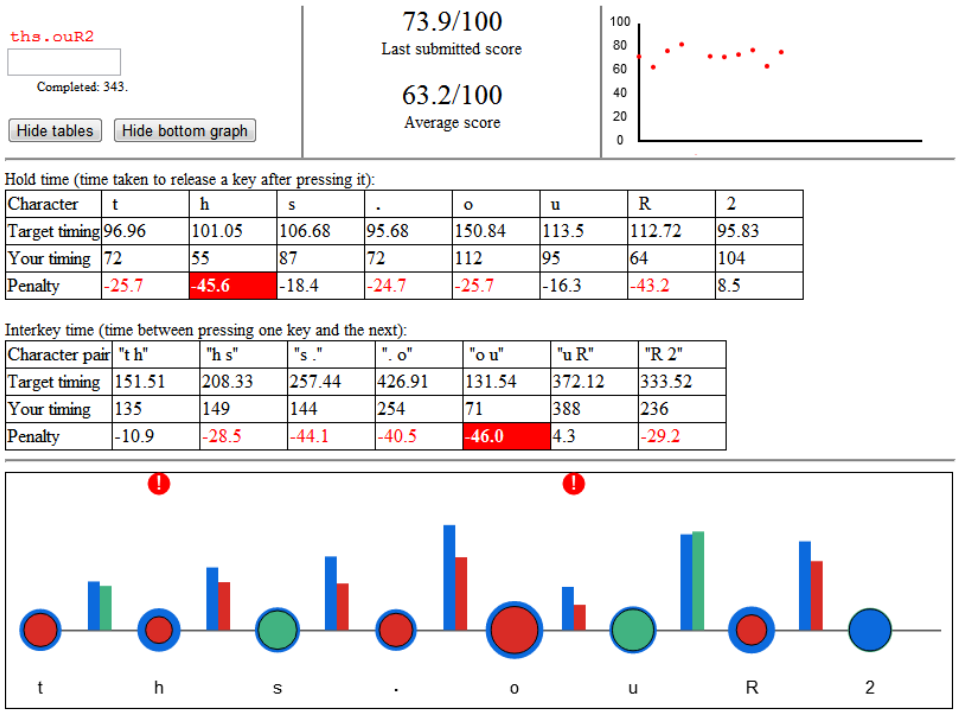


Figure 3.3: Mimesis interface with Euclidean distance

the attacker presses the enter key and only after typing the correct password. For practice sessions with password segments, no score is computed.

Top-right section This contains a graphical plot of the attack scores for the recent 20 correctly typed passwords. Our basis of comparison considers the best 20 consecutive vectors (see Section 3.2.4). This section therefore allows each attacker to easily grasp her past performance.

Central section This section contains two tables corresponding to the hold timing H and inter-keystroke timing I respectively. The tables provide numerical feedback on the victim's mean vector and the last submitted attacker vector. For M_{full}, a weight w computed from the corresponding victim's absolute deviation is also shown so that attackers know the relative importance of each key in calculating the attack score. To help the attackers make their adjustments, we also provide positive and negative feedback in the form of a penalty (the last row of each table) to the score. Penalty is computed based on the victim's and the attacker's typing. The

timing components accounting for the largest differences are highlighted in red as negative feedback. Components that are similar between attacker and victim are highlighted in a different colour as a positive feedback.

Bottom section This contains a graphical form of the information shown in the two tables. Circles represent the set of hold timings (H). Vertical bars between the circles represent the inter-keystroke timing (I). The larger/smaller the circle is, the longer/shorter H is, respectively. (Circles are chosen for the similarity with finger marks left on glass surfaces; the harder one presses, the bigger the mark.) Similarly, the taller/shorter the vertical bar is, the longer/shorter I is, respectively. As is the case for numerical feedback, both positive and negative feedback are provided, using different colour code. Red colour is used as negative feedback when the differences in component timings are large. An additional alert is placed above the component with the most critical difference. Green is used as positive feedback to indicate similarity between attacker and victim's timing components. In the case of M_{full}, a weight w (computed from the victim's absolute deviation vector) is added to the feedback as shown in Figure 3.4. w shows the relative importance of each component in calculating the attack score.

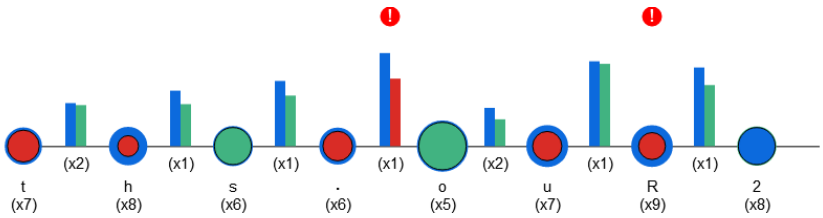
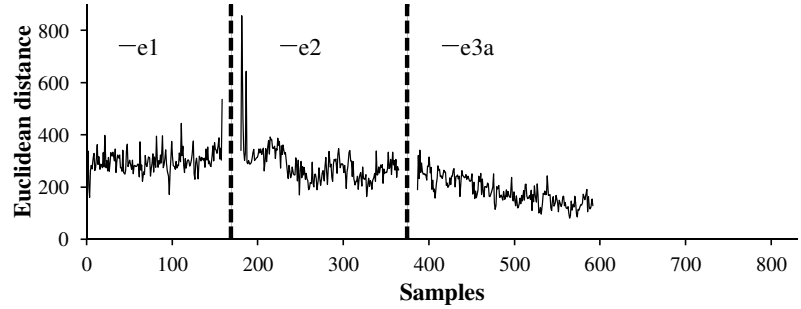


Figure 3.4: Mimesis interface (bottom section) based on Manhattan (scaled) distance

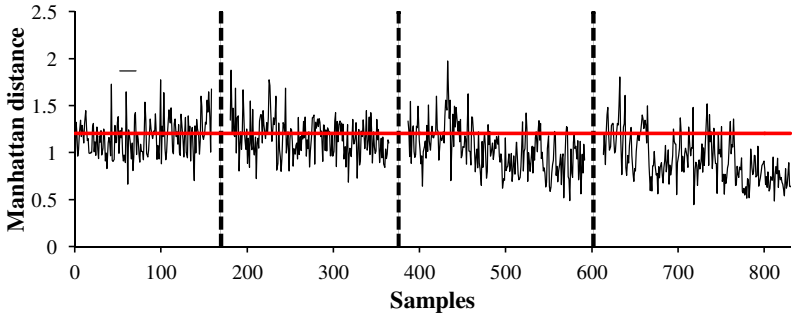
3.5 Evaluation

In this section, we present the results from our experiments. We start with the typing profile of a participant playing the role of an attacker as she progressed through each of the 4 experiments. Figure 3.5 shows the anomaly scores for both Euclidean

and Manhattan (scaled) distances against a running index of her submitted timing vectors. Vertical lines separate the vectors in each experiment; gaps in the running indices are added around each separator line for clarity; horizontal line is the equal error rate threshold of the victim; anomaly scores below the horizontal line are (falsely) accepted by the authentication system.



(a) Euclidean distance based anomaly score. The last section is omitted as the attacker is no longer practising against the Euclidean distance in e3b.



(b) Manhattan (scaled) distance based anomaly score

Figure 3.5: The anomaly scores of an attacker across all four experiments.

In e1, the attacker is required to type the assigned password 200 times. The Euclidean distance between each timing vector collected during this experiment and the victim's mean vector is plotted in the first section of Figure 3.5(a). Similarly, the Manhattan (scaled) distance is plotted in the first section of Figure 3.5(b). The data collected in e1 serves as a baseline because it is what an attacker can achieve without any imitation effort. Although we collected 200 vectors for each participant in this experiment, some vectors were discarded[4] and are not shown.

[4]Only e1 was conducted online. After it was concluded, we found that certain submitted vectors has near zero hold timing and inter-keystroke timing between the first and second characters because of network error. We filtered all such vectors from e1. The remaining experiments were all conducted in our lab and did not have the same issue.

In e2, the attacker is provided with Euclidean distance based feedback using the M_{part} interface. We can observe that the attacker took only less than 100 tries to achieve her best results for this experiment. Note that the improvement is more obvious in the Euclidean distance as compared to the Manhattan (scaled) distance.

In e3a, the attacker was given 20 minutes to repeat e2. After a one week of time along with the prior experience of e2 (learning effect), we can observe that the attacker produced a noticeable improvement in her Euclidean distance based anomaly scores. The corresponding improvement in Manhattan (scaled) distance is less pronounced. This is due to the weak correlation between Euclidean distance and Manhattan (scaled) distance. The coefficient of correlation between Euclidean and Manhattan (scaled) distance for all vectors collected in e1, e2 and e3a is 0.543.

In e3b, the attacker was given (a) feedback based on the Manhattan (scaled) distance, (b) a performance bonus to improve on her previous results and (c) additional time of up to 2 hours. These conditions simulate a motivated attacker operating under optimum conditions. We can see from Figure 3.5(b) a noticeable improvement towards the end of the experiment.

In the remainder of this section, we present the rest of our experimental findings. In Section 3.5.1 we investigate the possibility of collision attacks in e1 where the attackers are not provided with any form of feedback. We also evaluate the quality of detector parameter estimation with few samples. In Sections 3.5.2, 3.5.3 and 3.5.4 we present the outcome of e2, e3a and e3b experiments respectively.

3.5.1 Interesting results from e1

In this experiment, we obtained the timing vectors from 84 attackers who were asked to type their corresponding victim's password 200 times without any feedback. The victim assignment was random. 37 attackers typed the simpler password 'serndele', while the remainder typed the harder password 'ths.ouR2'. From the submitted timing vectors, we evaluate (a) the likelihood of collision attacks and

(b) the extent to which anomaly detector parameters can be estimated when few samples are available. The latter results provide the justification for the partial information scenario described in Section 3.2.2.

Collision attack

In Killourhy and Maxion's evaluation [40], the anomalous timing vectors of each user was constructed from the first 5 vectors submitted by all other users. This simulated attackers who are unfamiliar with typing their victim's password. They raised but did not answer the question of whether the FAR would change if attackers are allowed to practise typing the password. In `e1` we attempt to answer this question. We compute the overall FAR based on all vectors submitted by each attacker (instead of just the first 5).

Figure 3.6 shows the overall FAR in `e1`. Most attackers have an overall FAR of 0.2 or less. However, there exists 1 attacker (the last bar) with an overall FAR of more than 0.8. This implies that even without any imitation training, she has at least an 80% chance of pretending to be the victim successfully. This is an example of a collision attack. In practice, given a target organization with 10 high value targets, if a team of 84 attackers were to be assembled, we expect to find on average, one attacker with the same typing pattern as one of the high value targets.

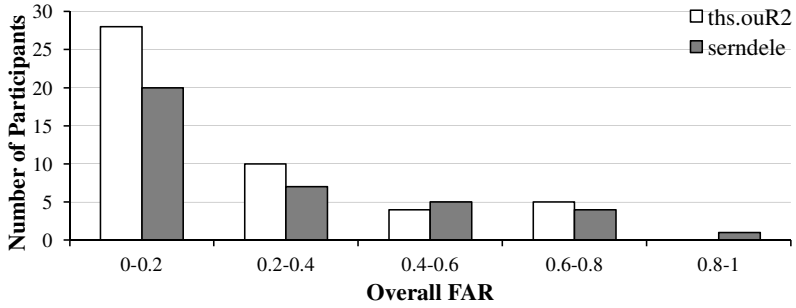


Figure 3.6: Overall FAR in `e1`

Estimation of anomaly detector parameters from few samples

We conducted a Monte Carlo simulation based on the timing vectors collected in e1. For each participant in e1, 10 samples were randomly picked from the collected data and the mean and absolute deviation were estimated based on these 10 samples. This is compared against the actual mean and absolute deviation computed using all available samples. We repeat this process 10,000 times for each participant. It was found that on average, the estimated mean is within $\pm 10\%$ of the actual mean 74% of the time. On the other hand, the estimated absolute deviation is only close to the actual absolute deviation 21% of the time. This shows that the scenario where an attacker can infer only the mean vector but not the absolute deviation vector is plausible. This provides the justification for the partial information scenario of e2 and e3a.

3.5.2 Imitation outcome of e2

In e2, each attacker is provided feedback based on the Euclidean distance assuming the partial information scenario of Section 3.2.2. 84 participants participated in this experiment. We present the change in FAR (see Section 3.5.2), followed by an analysis of how this change was affected by (a) choice of keyboard (see Section 3.5.2), (b) password difficulty (see Section 3.5.2), (c) attacker typing consistency (see Section 3.5.2). We also analyzed the optimum duration per training session in Section 3.5.2.

Improvement in FAR after imitation training

Figure 3.7 shows the FAR improvement in e2 as compared to e1. More than two-third of the attackers (56) improved their FAR from e1. However, there were some attackers (12) with no improvement in the FAR, while a small proportion (16) degraded. The last point demonstrates the shortcoming of using the Euclidean distance to approximate the Manhattan (scaled) distance in the partial information scenario.

Intuitively, if the attacker decreases her differences in one component of her vector, but increases in another, whether the corresponding Manhattan (scaled) distance increases or decreases depends on the scaling ratio of the two components, which is not known to the attacker. The partial information scenario is plausible, but not ideal.

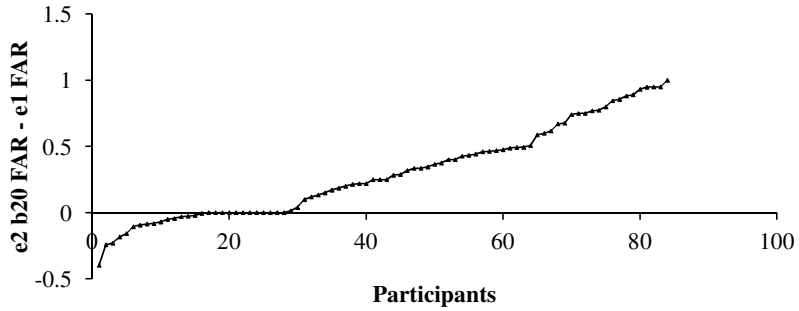


Figure 3.7: Improvement in FAR in e2 b20 from e1

Effect of keyboards

Prior to the experiments, we speculated that external keyboards, compared to notebook keyboards, facilitate the imitation. Feedback from attackers also supports this conjecture. To investigate this, in e2, 40 participants out of 84 participants used an external keyboard provided by us. The remainder used their own notebook keyboard. To verify our conjecture, we used a 2-sample Student's t test assuming unequal variance. The null hypothesis states that there are no differences between the mean of the b20 FAR for attackers using external keyboards, compared to those not using one. We use a two-tailed test as there is no conclusive evidence to support the use of a one-tailed test.

Groups	Mean	Variance	t Stat	P(T≤t)	t Critical
Own	0.564	0.146	1.217	0.227	1.989
External	0.462	0.148			

Table 3.1: Effect of keyboards

The results are shown in Table 3.1. While there are differences, the p value of 0.227 is not significant enough to conclude that an external keyboard made any

difference.

Effect of password difficulty

Figures 3.8(a) and 3.8(b) shows the change in overall FAR in e2 for different passwords. In e1, only 1 attacker practising on the easier password had a similar typing pattern as her victim ($0.8 \leq$ FAR ≤ 1). The training in e2 increased the number of such attackers to 6. For the harder password, no attacker is similar to her victim in e1. After e2, there were 2 such attackers. Statistical analysis using a 2-sample t-test with unequal variance showed that for the harder password, the change is not statistically significant (see Table 3.1(b)). In contrast, the change is highly significant for the easier password (see Table 3.1(a)).

(a) 'serndele'

Groups	Mean	Variance	t Stat	P(T≤t)	t Critical
e1 overall	0.241	0.065	-3.586	< 0.001	1.993
e2 overall	0.471	0.085			

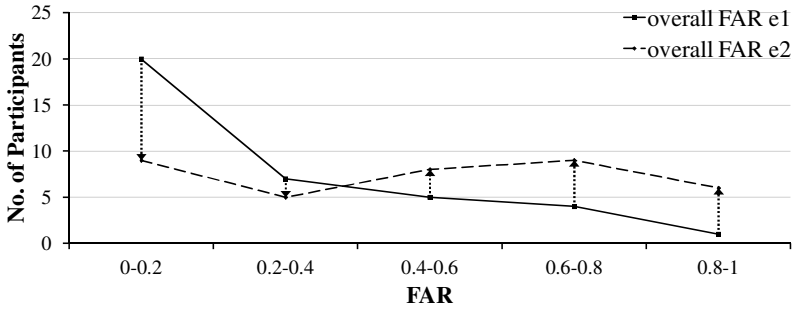
(b) 'ths.ouR2'

Groups	Mean	Variance	t Stat	P(T≤t)	t Critical
e1 overall	0.196	0.050	-1.769	0.081	1.987
e2 overall	0.288	0.075			

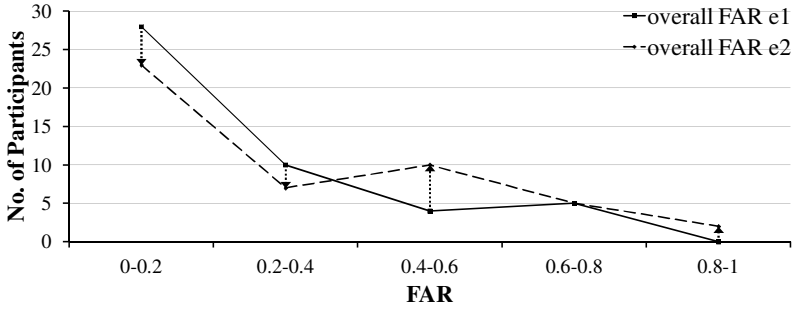
Table 3.2: t-test on overall FAR in e1 and overall FAR in e2

In Section 3.2.4, we explained why using the overall FAR underestimates the effects of imitation. Therefore, we also evaluate the change in b20 FAR. These are plotted in Figure 3.8(c) and 3.8(d) respectively. As expected, the number of attackers similar to their victims show a marked increase to 22 and 11 (as opposed to 1 and 0) for the easier and harder passwords respectively. This suggests that even imitation with partial information (based on M_{part} interface) helps the attacker in her performance significantly.

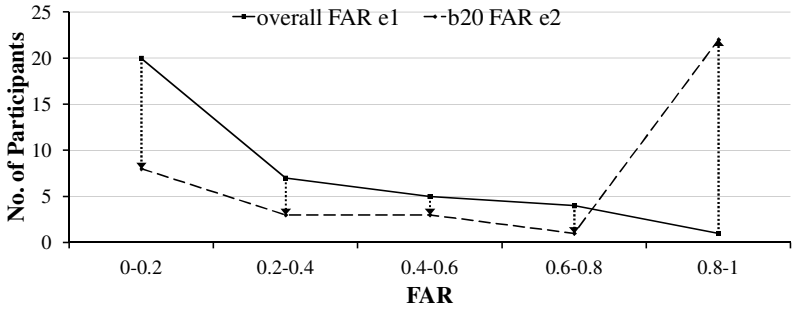
Statistical analysis using a 2-sample t-test with unequal variance showed that for both passwords, the change is highly significant (see Tables 3.2(a)and 3.2(b)). The differences in mean between the easier and the harder password suggest that



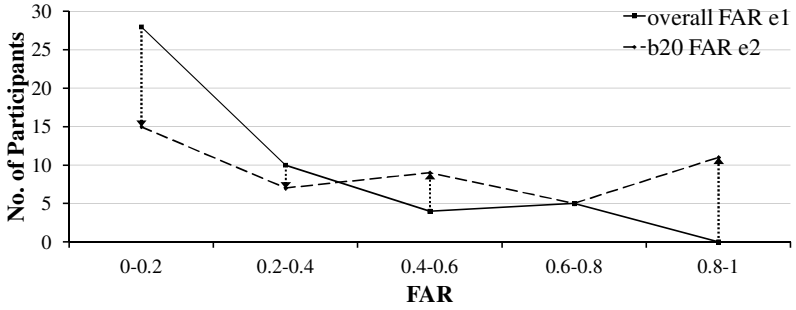
(a) 'serndele' - Using all samples



(b) 'ths.ouR2' - Using all samples



(c) 'serndele' - Using best consecutive 20 samples



(d) 'ths.ouR2' - Using best consecutive 20 samples

Figure 3.8: Improvement in FAR in e2 from e1

passwords that are easier to type are also easier to imitate. The question raised by Section 3.2.5 is therefore answered: the effectiveness of keystroke biometrics in mitigating weak passwords is lesser than previously assumed.

(a) 'serndele'

Groups	Mean	Variance	t Stat	P(T≤t)	t Critical
e1 overall	0.241	0.065	-5.126	< 0.001	1.998
e2 b20	0.633	0.150			

(b) 'ths.ouR2'

Groups	Mean	Variance	t Stat	P(T≤t)	t Critical
e1 overall	0.196	0.050	-3.678	< 0.001	1.991
e2 b20	0.425	0.131			

Table 3.3: t-test on overall FAR in e1 and b20 FAR in e2

Effect of attacker consistency

Intuitively, if an attacker is more consistent she should be able to exercise better control over her keystrokes, which should lead to better imitation outcome. To investigate the validity of this conjecture, we define a measure of consistency c. Referring to the vector notation of table 2.1, the standard deviation vector[5] ($\boldsymbol{s}$) can be computed using the following

$$s_j = \sqrt{\frac{\sum_{i=1}^{n}(t_{i,j} - \bar{x}_j)^2}{n-1}}$$

For each component of $\boldsymbol{s}$, the larger its value, the larger the variability and therefore the lesser the consistency score. The consistency score c for each participant is

[5]Note the difference between the standard deviation vector $\boldsymbol{s}$ vs. the absolute deviation vector $\boldsymbol{d}$ of the Manhattan (scaled) detector.

defined as the inverse of the average deviation in $\boldsymbol{s}$:

$$c = \frac{l}{\sum_{j=1}^{l} s_j}$$

The relation between imitation outcome and consistency is shown in Figure 3.9 which plots the b20 FAR against attacker consistency. Each point in the plot corresponds to one attacker. We can observe that there is no correlation between the imitation outcome (b20) of e2 against attacker's consistency score for both the easy password and the harder password. The coefficient of correlation for the easy password is 0.11 and for the harder password is -0.09. Therefore, there is no evidence to support our intuition that consistent attackers imitate better.

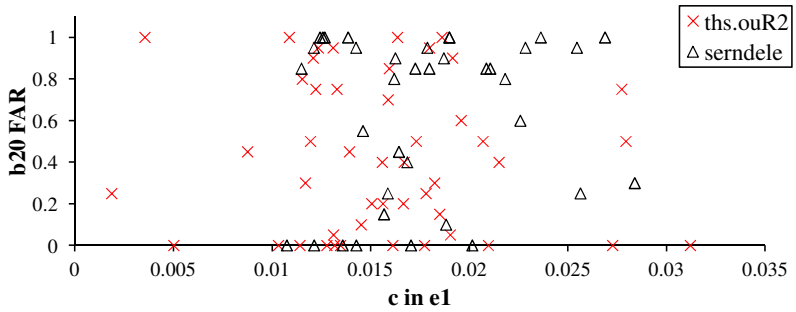


Figure 3.9: Imitation performance based on consistency in e1

A related and perhaps more interesting question is whether imitation training also improves attacker consistency. Figure 3.10 compares the consistency score of each attacker in e1 and e2. The attackers who are on the left of the vertical line were assigned the harder password 'ths.ouR2'. Those on the right were assigned the easier password 'serndele'. The attackers are sorted in a descending order according to their consistency score in e1.

Visual inspection of Figure 3.10 showed (a) there is no correlation between the c in e1 and e2, and (b) imitation training also improves the attacker's typing consistency regardless of the password complexity. (a) is confirmed by the coefficient of correlation, which has a near-zero value of 0.088. To verify (b), we used a 2-sample t-test with unequal variance. The results are shown in Table 3.4. The p value is less

than 0.001 and confirms that the difference in consistency score c between e1 and e2 is highly significant.

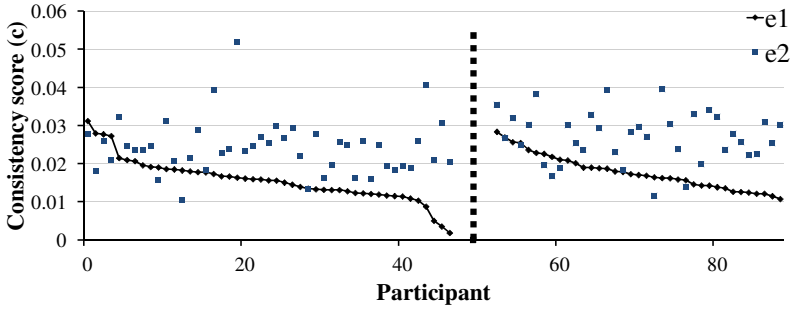


Figure 3.10: Consistency scores (c) in e1 and e2

Groups	Mean	Variance	t Stat	P(T≤t)	t Critical
e1	0.016	2.880E-05	-9.158	< 0.001	1.975
e2	0.025	5.005E-05			

Table 3.4: t-test on c in e1 and e2

Optimum duration per training session

In Figure 3.11, we show the time required for the attackers to reach their b20 performance in e2. 47 out of 84 (56%) attackers took less than 20 minutes. However, we also saw in Figure 3.5 that there is further room for improvement when given a second session. This suggests that instead of a single long session, imitation may be more effective when conducted in multiple sessions of shorter duration. A full investigation into the outcome for various combinations of session duration and number of sessions is however out of the scope of this chapter and we leave it for future work.

3.5.3 Imitation outcome of e3a

After e2, the 14 best attackers (based on their imitation performance and consistency score) were selected and given a week's rest. They were then recalled for a repeat of e2. Based on the findings in e2 we limit the duration of e3a to 20 minutes. The question we want to investigate in this section is that under the partial information

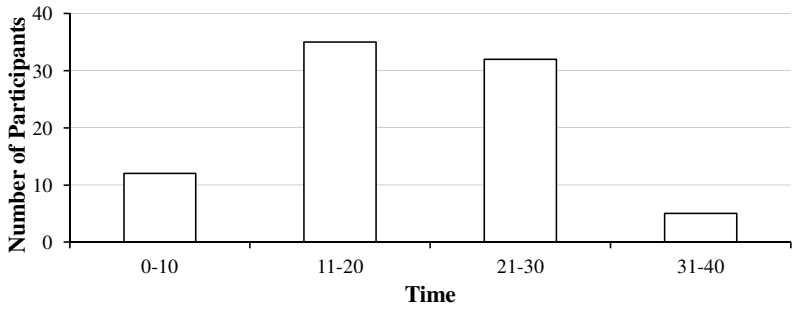


Figure 3.11: Time required in e2

scenario, do attackers reach their peak performance within the first 30 minutes or they are capable of further improvements when given more time to rest, reflect and repeat their earlier efforts.

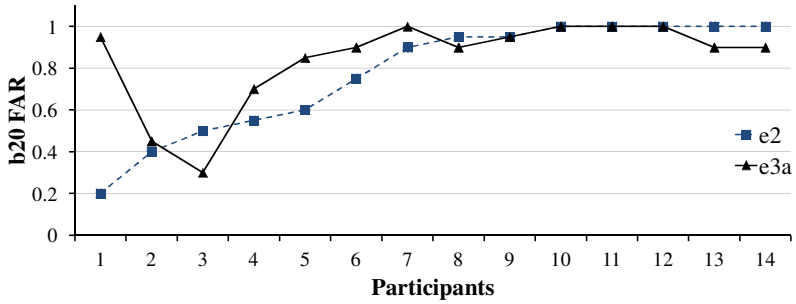


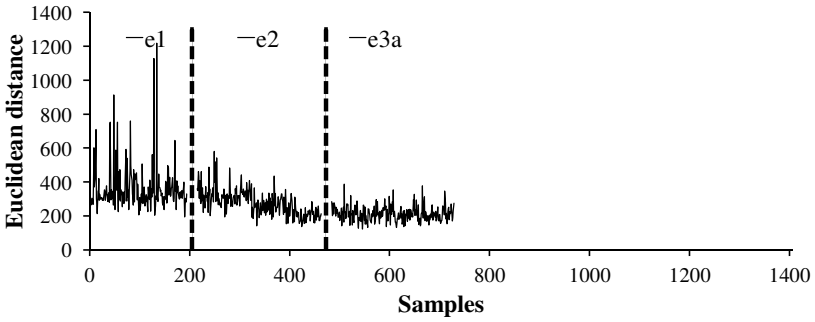
Figure 3.12: b20 FAR in e2 and e3a

Figure 3.12 shows the improvement in the b20 FAR between e2 and e3a. We found that there is no significant difference between b20 FAR obtained in e2 and e3a (see Table 3.5). Out of the 14 attackers, 6 improved their b20 FAR, 4 were unchanged while 4 actually worsened.

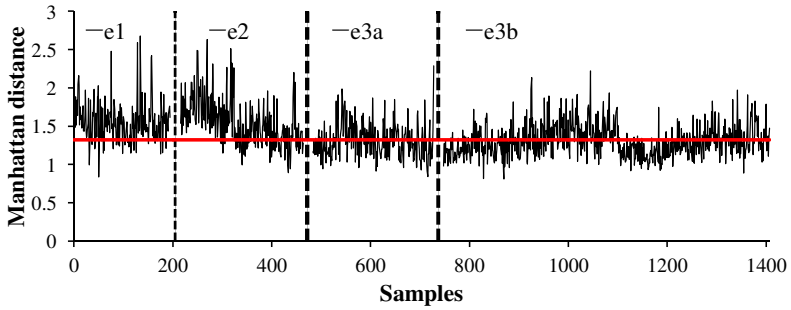
Groups	Mean	Variance	t Stat	P(T≤t)	t Critical
e2	0.771	0.073	-0.770	0.448	2.059
e3a	0.842	0.046			

Table 3.5: t-test on b20 FAR in e2 and e3a

To explain the anomaly where the performance of some participants actually degraded, we examine the profile of one such attacker with the worst e3a FAR (participant no. 3 in Figure 3.12). Figure 3.13 shows the typing profile of this participant where the anomaly scores are plotted against a running index of each timing vector. We can observe that the Euclidean score of this attacker actually



(a) Euclidean distance based anomaly score



(b) Manhattan (scaled) distance based anomaly score

Figure 3.13: The anomaly scores of the worst performing attacker of e3a

improves in e3a. However, the improvement in Euclidean score did not translate into a marked improvement in Manhattan (scaled) score for this attacker. The reason is due to the weak correlation between these two distances.

3.5.4 Imitation outcome of e3b

Experiment e3b was conducted following e3a after a break of 15 minutes. This experiment simulates highly motivated attackers operating under optimum conditions (full victim information, performance bonus and more time). The attackers were told to try to achieve the 2 targets (see Section 3.3.4 for details). Out of the 14 attackers, 2 managed to achieve the lesser bonus and another 3 achieved the full bonus. (Note that qualifying for the bonus is more difficult than crossing their victim's acceptance threshold.) We can observe from Figure 3.14 that almost all attackers were able to achieve near perfect imitation of their victims. The results of a 2-sample t-test with unequal variance is shown in Table 3.6. The p value of 0.022 confirms that the difference in the FAR for e3a and e3b is statistically significant.

Groups	Mean	Variance	t Stat	P(T≤t)	t Critical
e3a	0.842	0.046	-2.594	0.022	2.160
e3b	0.992	3.29E-04			

Table 3.6: t-test on b20 FAR in e3a and e3b

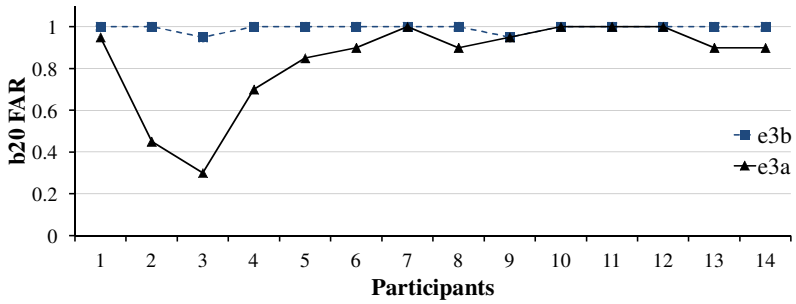


Figure 3.14: b20 FAR in e3a and e3b for participants of all four experiments

Figure 3.15 shows for one victim, her original FRR from e1, original FAR from e1 and the FAR of her 2 assigned attackers (a1 and a2) from e3b. During the training phase, the threshold is set at a Manhattan (scaled) distance of 1.2, resulting in an EER of 0.2 for the detector. Imitation training markedly increases the FAR curve for both attackers. If the threshold remains unchanged, FAR increases to 1. This means the detector is unable to differentiate between the attackers and this victim. If the threshold is recalibrated to a distance of 0.75, it results in an unacceptable EER of 0.7.

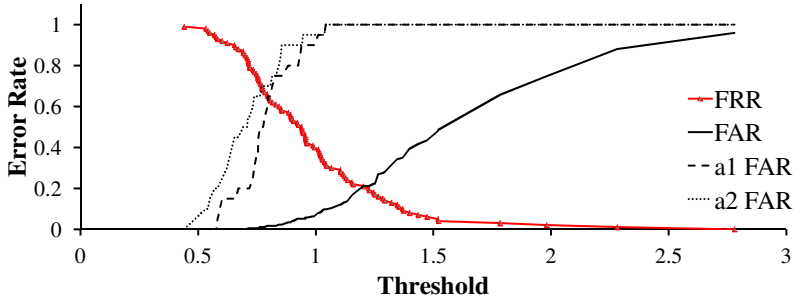


Figure 3.15: Effect of imitation training on FRR and FAR

The amount of time taken by each attacker in e3b to reach their b20 performance is shown in Figure 3.16. For 9 out of 14 (64%) attackers, their performance peaked in 20 minutes or less. This is consistent with our observations in Section 3.5.2. Two highly motivated participants took nearly 2 hours. One of them achieved both

performance bonuses at the end.

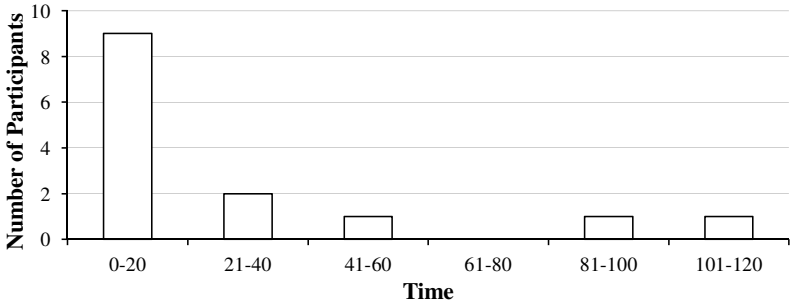


Figure 3.16: Time required in e3b

3.6 Discussion

In this section, we discuss the various factors affecting the outcome of the imitation, such as gender (see Section 3.6.1), typing speed (see Section 3.6.1), imitation strategy (see Section 3.6.1) and similarities in typing patterns (see Section 3.6.1). In Section 3.6.2, we discuss the attackers' interface preferences. Their perception towards the difficulty of hold timings or inter-keystroke timings is discussed in Section 3.6.3. Finally, we state the limitations in Section 3.6.4.

3.6.1 Factors affecting imitation outcome

Various hypotheses were put forward by both researchers and participants in the course of our experiments to account for the differences in imitation outcome. We evaluate each of them in the following discussions.

Gender

In the experiment e2, our pool of participants consists of 39 males and 45 females. In Figure 3.17, we show the average e2 b20 FAR for both genders across the two passwords used. We can observe that male attackers achieved an average FAR of 0.51 and 0.81 for the harder and easier password respectively, compared to 0.33 and 0.51 for the female attackers. Male participants therefore perform significantly

better than females in the imitation experiments. Furthermore, for both genders, the easier password to type is also the easier password to imitate, although the difference is more pronounced in males.

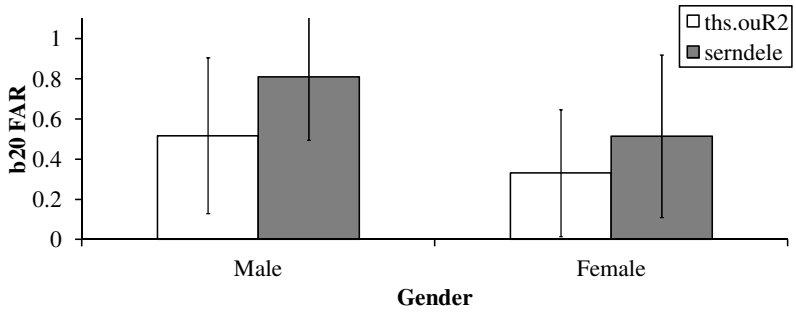


Figure 3.17: FAR based on gender

Groups	Mean	Variance	t Stat	P(T≤t)	t Critical
Female	0.420	0.133	-2.548	0.006	1.664
Male	0.629	0.149			

Table 3.7: t-test on b20 FAR in e2 on gender

We confirm the differences using a 2-sample t-test assuming unequal variance. The null hypothesis states that there are no differences between the mean of the b20 FAR between male and female attackers. The results are summarized in Table 3.7, which shows that the null hypothesis can be rejected. Since the p value is less than 1%, the test is highly significant.

Typing speed

During the experiment, feedback from certain attackers indicated that they believed slower victims are easier to imitate. To evaluate this, a measure of their typing latency is required. For each attacker and victim, we compute the average timing of all components in each participant's mean vector $\bar{x}$ as a measure of their latency:

$$v = \frac{\sum_{j=1}^{l} \bar{x}_j}{l}$$

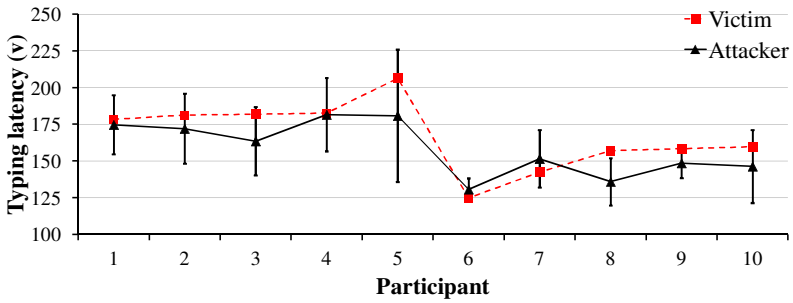


Figure 3.18: Typing latency of each victim and their attackers

Figure 3.18 shows the latency profile of all 10 chosen victims using the typing keystrokes from e1. Each victim was assigned 8 to 9 attackers. The attackers' latencies are shown on the same plot as vertical lines. The midpoint of each line indicates the mean attacker latency. The top and bottom of each line are one standard deviation away from the mean. The first 5 victims and their corresponding attackers were assigned the harder password 'ths.ouR2' and the last 5 practised on the easier password 'serndele'. The assigned attackers are generally spread out, with a mix of faster, equal and slower attackers in typing relative to the victims.

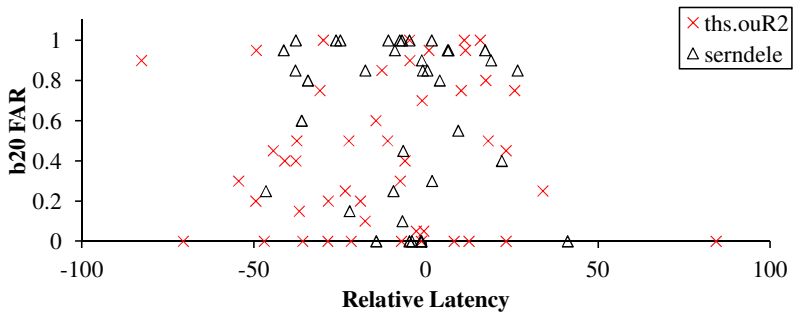


Figure 3.19: Relative latency v/s b20 FAR in e2

To investigate whether it is easier for a faster attacker to imitate a slow victim, the relative latency of each attacker w.r.t. her victim is computed. The coefficient of correlation between the b20 FAR in e2 and the relative latency is 0.02 and -0.12 for the harder and easier passwords respectively. Therefore, contrary to participant's feedback, there exists no correlation between the typing speed and the imitation outcome. This is shown in Figure 3.19.

Number of trials per minute

During our experiments, we notice that participants vary greatly in the number of samples submitted. Different attackers adopt different strategy to get to their best high score. Certain participants would submit samples after samples, while others spend more time studying and reflecting on the feedback mechanism. For example some would pause and tap on their palm to grasp the rhythm. We want to know which approach is better and whether there is any effect on the FAR.

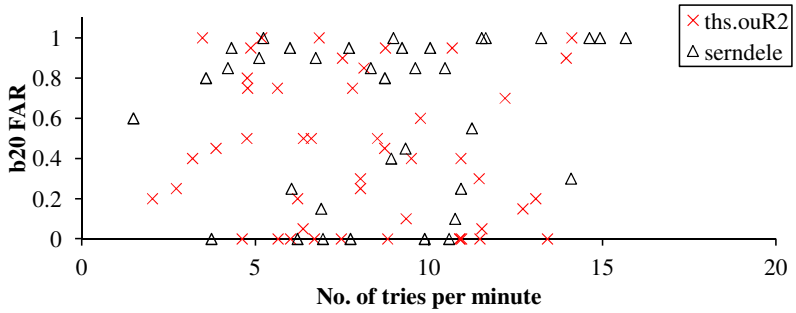


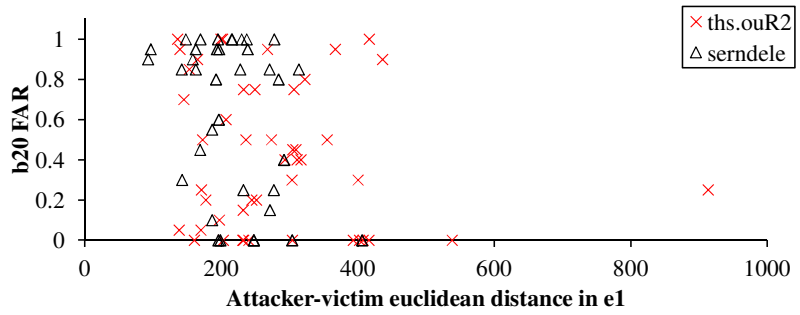
Figure 3.20: Tries per minute in e2

Figure 3.20 shows the b20 FAR in e2 against the number of tries per minute. The coefficient of correlation is 0.069. We found no correlation between how each attacker goes about improving their imitation and the acceptance rate. This also suggests that there is no standard way to perform better in imitating someone else.

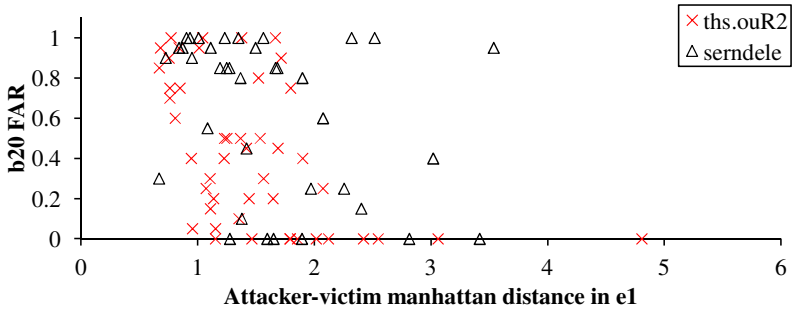
Initial typing similarity with the victim

If an attacker's typing pattern is already similar to the victim's before imitation training, intuitively, one would expect that even a slight improvement would result in a noticeable change in the FAR. In Figures 3.21(a), we show the b20 FAR in e2 of each attacker against the Euclidean distance between her mean vector and that of the victim's. In Figure 3.21(b), we show the b20 FAR in e2 of each attacker against the Manhattan (scaled) distance of the attacker's mean vector from the victim's mean vector (with the scaling based on the victim's deviation vector).

The coefficients of correlation are -0.26 and -0.38 for Figures 3.21(a) and 3.21(b) respectively. Therefore there exists a weak correlation between the e2 imi-



(a) Against Euclidean distance between attacker and victim



(b) Against Manhattan distance between attacker and victim

Figure 3.21: Correlation between b20 FAR in e2 and the typing similarity between attacker and victim

tation outcome and the similarity between the attacker and victim's typing pattern. From Figure 3.7, we can observe that the correlation is weak because the extent of improvement varies for different attackers.

3.6.2 Mimesis interface

In Section 3.4, Mimesis provided feedback in both a table of numerical timings as well as a graph. Figure 3.22 shows the preference among attackers for these 2 feedback options. There are 51 participants who relied predominantly on the graph, while 23 relied on the raw data shown in the tables. 6 used both and another 4 used neither. For the latter, this implies that they relied only on the attack score and the colouring scheme adopted. Among participants who used the table and/or graph, feedback indicated that the reliance is only during the initial part of the experiment to get the rhythm correct. Thereafter, participants usually rely on their gut feeling to imitate.

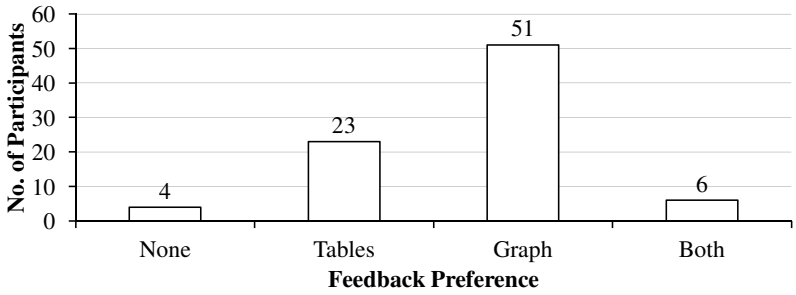


Figure 3.22: Preferences based on the feedback

3.6.3 Imitating hold v/s inter-keystroke timing

In this section we compare whether it is easier to imitate hold timing or inter-keystroke timing. From Figure 3.23, there are 42 attackers who find hold timing easy to imitate and inter-keystroke timing difficult. On the other hand, 30 attackers found hold timing difficult and inter-keystroke timing easy. 3 attackers found both are easy to imitate, while 9 think that both are equally difficult.

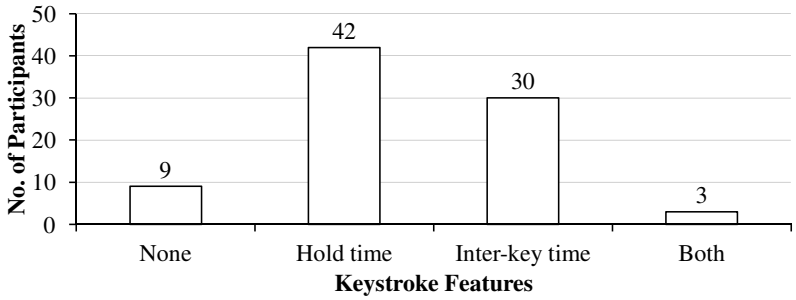


Figure 3.23: Easier to imitate

	Notebook keyboard	External keyboard
H easy, *I* difficult	19	23
H difficult, *I* easy	20	10

Table 3.8: Effect of the keyboard in hold timing and inter-keystroke timing.

It turned out that the type of keyboard made a difference. Refer to Table 3.8. Of the 72 attackers who found one timing easy and the other difficult, 39 used their notebook's keyboard while 33 used an external keyboard. We found that among those using an external keyboard, the number of people who find hold timing easy to imitate is significantly more than those who do not. The reason could be because pressing and releasing of a key is more apparent in an external keyboard as

compared to the notebook's keyboard.

3.6.4 Limitations

In Section 3.2, it was stated that we deliberately exclude the key up-down timing as a precaution that it may be difficult to understand and therefore affects the experimental results. A question therefore arises on whether it is really difficult to imitate key up-down timing. If true, it implies that a simple countermeasure against imitation would be to include and give a greater weight to key up-down timing in the anomaly score calculations. We did not address this issue in this thesis and leave it as future work.

3.7 Summary

This chapter shows that contrary to the beliefs of prior studies, when a victim's typing pattern is known, imitation is possible. If the attacker has an incomplete model of the victim's typing pattern, her success rate is around 0.52 after imitation training. For the best attackers, imitation training increases the FAR to nearly 1, rendering keystroke biometrics based authentication systems unusable. Furthermore, when the number of attackers and victims are sizeable, the chance of a natural collision in typing pattern (without any imitation training) is significant.

Among the key factors affecting the imitation, we found that the easier the password, the easier the imitation. Males were also found to be better at imitation compared to females. On the other hand, various factors such as use of external keyboard, typing consistency, typing speed, imitation strategy and similarities in typing patterns were found to have much less influence on the imitation outcome.

Chapter 4

Can typing patterns be collected from JavaScript applications?

4.1 Introduction

Rich and complex JavaScript applications provide sophisticated GUI updates and fast client-server communications that approaches the capabilities of traditional desktop applications. The front-end JavaScript communicates using HTTP(s) with the back-end server in response to various events such as keypress. In this chapter we explore whether the improved GUI creates a timing side channel. We hope a detailed study of one application yields insights on the threats that may apply to the entire class of such applications.

Similar side channels attack had been demonstrated by Chen et al. [29] (inferring encrypted JavaScript traffic from packet size) and Song et al [47] (reducing search space of SSH passwords from packet timing). This chapter differs from prior work in the following ways. It is the first to analyse JavaScript timing side channels and use it to derive typing patterns. Inference of typing pattern raises privacy concerns. Prior research showed that natural typing patterns are unique enough for user identification [25, 39, 43, 42]. Natural typing occurs when no attempts are made to consciously alter the typing.

Personalized typing patterns also improves the SSH attacks by Song et al. [47] and allows imitation attacks (see Chapter 3) on keystroke biometrics systems [35, 43, 25, 34, 38]. JavaScript timing side channels are challenging to analyse because keystroke and network traffic timing are only loosely correlated. This is because JavaScript applications (a) are far slower as compared to native binary applications and (b) typically run in a single threaded co-operative multitasking execution model.

In the following sections, we first study Google Suggestions' communication model in Section 4.2. We derive a set of techniques to construct a keypair timing model (probability distribution of keypair intervals) for each pair of keystroke from unencrypted Google Suggestions traffic in Section 4.3. We conducted a user study on 11 participants to collect their keystrokes and timings. The user study and results are shown in Section 4.4. We discuss the limitations and summarise this chapter in Sections 4.5 and 4.6 respectively.

4.2 Communication model

Our approach to study the Google Suggestions communication model is based on both black-box testing and white-box analysis. We used the setup of Figure 4.1. The client under testing connects to the Google servers in the back-end through a proxy server. For blackbox testing, we captured Google query packets using a packet sniffer [23] installed on the client. For white-box testing, we hosted a copy[1] of the Google HTML and JavaScript files on another web server and selectively redirect the proxy server [22] to fetch our copy rather than from the actual Google server. This allows us to make arbitrary changes to the scripts for our testing.

4.2.1 Approach

Black-box analysis allowed the quick identification of traffic patterns and content. For example, we quickly found that different network traffic patterns are possible

[1] Retrieved Apr 2012

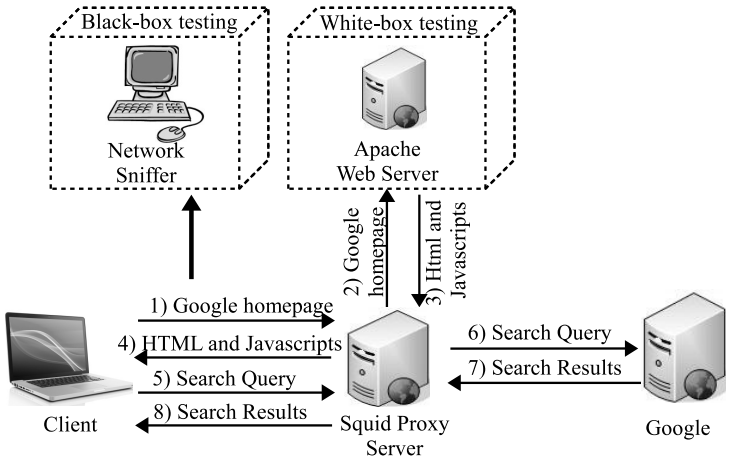


Figure 4.1: Setup for black-box and white-box testing

even for the same query (see Table 1.1). On the other hand, when we analysed the timings of the keypress and the packets, we encountered significant difficulty correlating them. For example, after the user pressed a key, the corresponding HTTP request can be observed on the network from between approximately 3 ms to over 100 ms later with 2 distinctive frequency peaks at around 7 ms and 45 ms. White-box analysis is therefore necessary.

Our approach is to first manipulate the obfuscated source code using the tool JS-Beautifier [15]. The decision to host a separate copy of the script files in Figure 4.1 allowed us to make arbitrary changes to the JavaScript source code independently of the Google servers. Next, we use the console logging feature of Firebug to pinpoint the code that initiated the HTTP requests. This formed the starting point for subsequent investigations, where we incrementally assign meaningful symbols to the variables and functions through (a) monitored calls to standard functions, and (b) selectively breaking execution and examining the call stack and variables. Please note that our investigation focuses specifically on the timing aspects. Hence we did not deobfuscate all the script code involved in Google Suggestions. The rest of this section documents our findings.

4.2.2 Communication model obtained

JavaScript uses an event driven execution model [10]. For Google Suggestions, there are 3 classes of event handlers of interest. $\underline{\texttt{H}_{\texttt{poll}}}$ is a handler for polling events. The polling is setup and removed when the query input box receives and loses focus respectively. Although the specified polling interval is 10 ms, the actual firing interval fluctuates. The reason is likely to be due to other events firing and executing, thereby delaying the execution of this handler. $\underline{\texttt{H}_{\texttt{ui}}}$ handles UI events, e.g., `keydown`, `keypress`, `keyup`, etc. The same handler code fires for different events but with different closure scope. The handler function for the `keydown` event, named $\texttt{H}^{\texttt{kd}}_{\texttt{ui}}$, is installed during GS code initialization. The Google Suggestions code includes a mechanism to defer execution of a function. $\underline{\texttt{H}_{\texttt{defer}}}$ handles the events which are deferred. The 2 key parts to this are the `postMessage` JavaScript function and an array of deferred functions ($\texttt{Arr}_{\texttt{defer}}$). At load time, Google Suggestions setups a `message` event handler. This handler fires when a message is posted to it using `postMessage`. When fired, it removes the first function from $\texttt{Arr}_{\texttt{defer}}$ and executes it. If $\texttt{Arr}_{\texttt{defer}}$ is not empty, it posts a message to itself and exits. Any JavaScript code deferring execution calls a wrapper function `defer` which first pushes the function to defer onto the bottom of $\texttt{Arr}_{\texttt{defer}}$ and then post a message to $\texttt{H}_{\texttt{defer}}$.

When a user presses and releases a key, JavaScript fires four events in this order: `keydown`, `keypress`, `input`, `keyup`. Under normal circumstances, $\texttt{H}^{\texttt{kd}}_{\texttt{ui}}$ uses the deferred execution mechanism to invoke a function named $\texttt{H}_{\texttt{x}}$ to send out the network traffic. However, it is also possible for $\texttt{H}_{\texttt{poll}}$ to run before $\texttt{H}^{\texttt{kd}}_{\texttt{ui}}$. In such a case, $\texttt{H}_{\texttt{poll}}$ invokes $\texttt{H}_{\texttt{x}}$ directly (without deferring execution) to send out the query. Regardless of the path taken, $\texttt{H}_{\texttt{x}}$ is executed at most once for each keystroke. The end result is a race (to execute $\texttt{H}_{\texttt{x}}$) between the synchronous mechanism of $\texttt{H}_{\texttt{poll}}$ and the asynchronous mechanism of $\texttt{H}^{\texttt{kd}}_{\texttt{ui}}$, resulting in the introduction of a variable delay. The race is won mostly by $\texttt{H}^{\texttt{kd}}_{\texttt{ui}}$.

Another factor affecting the execution delay is the number of task on each execution path. For example, the first keypress for Google Suggestions also updates the UI in preparation for not just the suggestions of Google Suggestions, but also the results of Google Instant. This additional code increases the execution delay by approximately 6 times ($\sim$45 ms).

A third factor affecting the delay is submission throttling [41]. Most search engines used this technology to limit the amount of search traffic to their website while the user is typing the query. In the case of Google Suggestions, regardless of whether $\mathtt{H}_{\mathtt{poll}}$ or $\mathtt{H}^{\mathtt{kd}}_{\mathtt{ui}}$ won the race, $\mathtt{H}_{\mathtt{x}}$ is always invoked. The role of $\mathtt{H}_{\mathtt{x}}$ is to send queries and receive results from the Google servers. Listing 1 shows how submission throttling is implemented in Google Suggestions when $\mathtt{H}_{\mathtt{x}}$ is invoked.

Listing 1

```
trace Hx
    if xhr_timer not pending then
        xhr_m enter
            ...
        exit
    end if
end trace

trace TimerEvent
    Call xhr_m
end trace

procedure xhr_m
    if unsent_query then
        xhr_i enter
            ...
            send query to Google
            ...
        exit
        timeout ← compute_timeout
        create TimerEvent
    end if
end procedure
```

The sending mechanism of Google Suggestions uses a timer named $\mathtt{xhr}_{\mathtt{timer}}$. This timer is initially cleared. When $\mathtt{H}_{\mathtt{x}}$ is invoked it checks this timer. If $\mathtt{xhr}_{\mathtt{timer}}$ is

cleared, $\texttt{H}_\texttt{x}$ calls a sub function $\texttt{xhr}_\texttt{m}$ to send out the query immediately. Otherwise, it exits without sending any HTTP traffic. When $\texttt{xhr}_\texttt{m}$ runs, it sets up the timer $\texttt{xhr}_\texttt{timer}$ to call itself ($\texttt{xhr}_\texttt{m}$) again after a timeout value. The detailed computation of the timeout is out of the scope of this chapter, but on a fast network, this value is 100 ms. After this timer is set, $\texttt{xhr}_\texttt{m}$ will not run again until the timer expires. Any keystrokes typed during this time accumulate and are sent together in the same HTTP request when the timer expires.

If $\texttt{xhr}_\texttt{m}$ runs but does not find any unsent query (that is, between the previous and current invocation of $\texttt{xhr}_\texttt{m}$, the user did not press any key), it does not set any new timer. $\texttt{xhr}_\texttt{timer}$ therefore becomes cleared again. If a new key is now pressed, $\texttt{xhr}_\texttt{m}$ will again send it out without delay. The described process then repeats itself. The implication is that correlation between keystroke and packet timing is poor whenever $\texttt{xhr}_\texttt{m}$ is in timer mode.

4.3 Recovery of keypair timing model

Section 4.2 identifies the timeout mechanism and atypical execution path as major noise contributors. Packet timings thus affected are considered unreliable. Figure 4.2 shows that if we discard the unreliable timing, the delay between pressing of keystroke and sending of packet becomes significantly more consistent. (t_{ks} and t_{pkt} refer to the keystroke and packet timing respectively.) This allows the recovery of the derived keypair timing model.

For each keypair, the recovery process involves:

1. Identifying the corresponding packet-pairs
2. Determining if each packet timing is reliable
3. Choosing packet-pairs where the earlier timing is reliable
4. Further dividing the chosen packet-pairs into a set where the latter packet timing is reliable and another set where it is not. The size of the first set (reliable

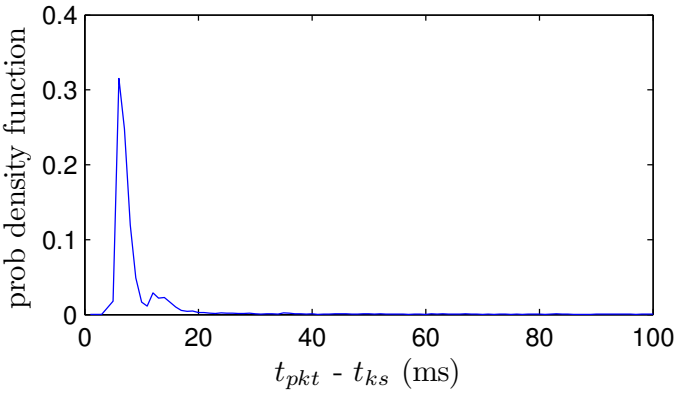


Figure 4.2: Noise model

latter packet timing) is denoted by $\underline{N_o}$. The size of the complementary set is denoted by $\underline{N_u}$.

5. Computing the mean and variance of the packet-pair timing model

6. Applying a correction to the variance to obtain the derived keypair timing model.

This process requires an assumption of normally distributed timing models which are independent. Prior work [47] investigating keypair timing model found the normal distribution to be a reasonable approximation.

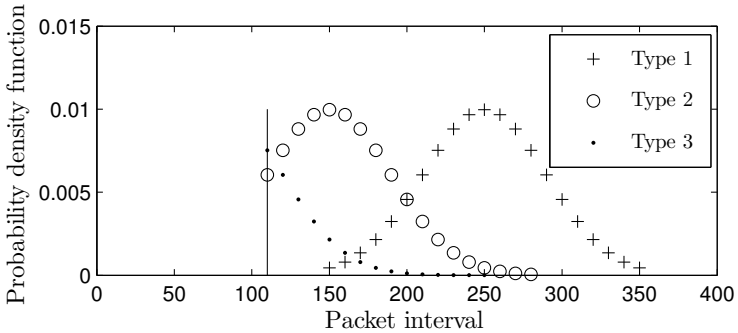


Figure 4.3: Different scenarios for the building of $\texttt{DTM}_{\texttt{key}}$ from $\texttt{TM}_{\texttt{pkt}}$

In step 5, depending on the values of N_o and N_u, there can be 4 different scenarios, 3 of which are shown in figure 4.3. $\texttt{TM}_{\texttt{key}}$ denotes the keypair timing model, $\texttt{TM}_{\texttt{pkt}}$ denotes the packet-pair timing model, and $\texttt{DTM}_{\texttt{key}}$ denotes the derived keypair

timing model, which is an approximation of $\texttt{TM}_{\texttt{key}}$ obtained by applying a variance correction to $\texttt{TM}_{\texttt{pkt}}$. In type 1 scenarios, $N_u = 0$. The mean and variance are calculated directly from the observed intervals.

For type 2, $N_u < N_o$. The timeout translates to a cutoff time beyond which certain intervals are not observed. The peak though is still visible. We first estimate the median from N_u and the observable parts of the distribution. We next obtain the mean which is equal to the median. We estimated the missing part of the distribution by reflecting the observable part about the mean. From the reconstructed distribution, we can calculate the variance.

For type 3, $N_u \geq N_o$. The peak is not visible. Our aim is to fit a normal distribution (with unknown mean $\bar{x}$ and std. deviation s) based on the interval observations on the right tail. Let l denote the cutoff time (timeout + a small allowance). $\bar{x}$, s, l, α are related by $l = \bar{x} + \alpha s$, where α is a multiplier s.t. for a standard normal distributed random variable X, $\Pr(X > \alpha) = N_o/(N_o + N_u)$. The curve fitting iterates over a possible list of values for $\bar{x}$, and computes the corresponding s. The values providing the best fit (least squares) are chosen.

In type 4 scenarios, N_o = 0. The mean of the keypair timing is far less than the timeout, resulting in no interval observations. Hence it is not plotted in Figure 4.3. To recover the timing for a keypair such as $c_1 - c_2$ where c_i denotes a key pressed, we need to have the parameters of another 2 distributions: the keypair $c_2 - c_3$ and the triplet $c_1 - c_2 - c_3$. The latter 2 distributions would be observable if there exists c_3 such that $c_2 - c_3$ is much longer than the timeout.

Given two independent normally distributed random variables X and Y, the random variable $Z = X + Y$ is also normal [17] with mean $\bar{z} = \bar{x} + \bar{y}$ and standard deviation $s_z = \sqrt{s_x^2 + s_y^2}$. The relation between $c_1 - c_2$, $c_2 - c_3$ and $c_1 - c_2 - c_3$ is analogous to that of X, Y and Z. Therefore, if we let Z and Y represent the distribution for $c_1 - c_2 - c_3$ and $c_2 - c_3$ respectively, we can obtain the mean and standard deviation of the unobservable $c_1 - c_2$ from X.

In step (f) we need to apply a correction to the variance but not the mean. This

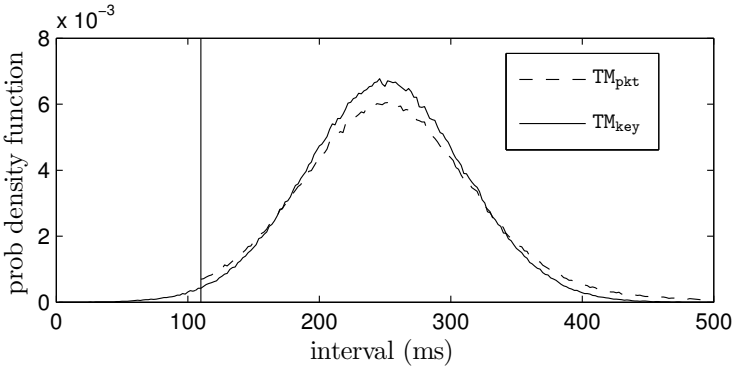


Figure 4.4: Difference in the $p.d.f.$ of $\texttt{TM}_{\texttt{key}}$ compared to the corresponding $p.d.f.$ of $\texttt{TM}_{\texttt{pkt}}$.

is because there is residual noise even after accounting for the timeout and atypical execution path. This noise affects both timing observations of a packet-pair. It cancels out for the interval mean, but adds to the variance. Figure 4.4 shows this effect for a keypair. To compute the required variance correction, we use a simple heuristic. A Monte Carlo simulation based on the model of Section 4.2 computes the observed variance for a set of variances. The differences are stored in a table and looked up whenever a correction is needed.

4.4 User study

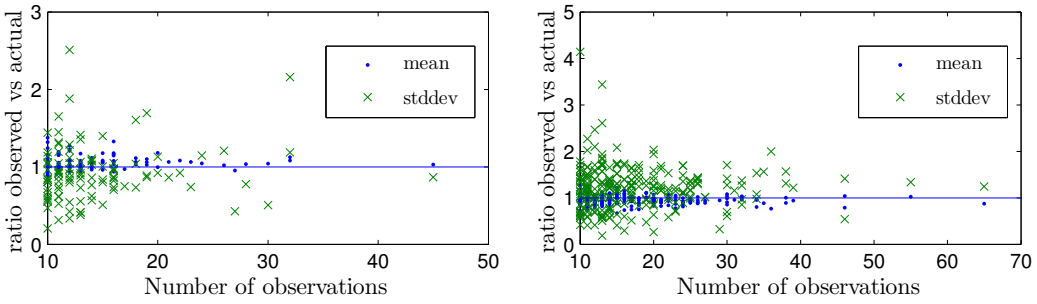
To verify the theory of Section 4.3, we conducted a user study. 11 participants are asked to install a plugin on their browser which captures the keystroke timings of Google Suggestions queries. The duration of the study ranges from 32 to 49 days. Users are allowed to inspect and delete any sensitive entries in the capture log before submission. Towards the end of the study, users with too few queries were given a chance to go through a Q&A worksheet using Google to find the answers. This is so that they get more opportunities in using Google search. The collected keystrokes are anonymised and post processed to retain only English alphabets and the SPACE char. Queries with BACKSPACE are broken up. The resulting logs are consolidated on a single machine running Ubuntu 11.10 (AMD Athlon(tm) 64

S/N	Q	KS	KP	TP	KP_{obs}	TP_{obs}	N_{sig}
1	502	3114	2612	2110	642	487	6
2	421	2666	2245	1824	682	506	7
3	1206	6607	5401	4195	2447	1715	93
4	688	4243	3555	2867	601	403	6
5	593	3604	3011	2418	1368	993	34
6	774	4592	3818	3044	1752	1284	58
7	405	2517	2112	1707	733	561	8
8	696	4610	3914	3218	1181	893	29
9	327	2163	1836	1509	270	185	1
10	1041	6042	5001	3960	2359	1697	96
11	700	3964	3264	2564	1233	853	29

Table 4.1: Statistics of user study. Q: total number of queries by user. KS: total keystrokes typed. KP: total keypairs typed. TP: total 3 char sequence. KP_{obs}: sum of N_o for all keypairs. TP_{obs}: sum of N_o for all triplets. N_{sig}: total number of keypair/triplets for which $N_o \geq 10$. This is also the number of recovered $p.d.f$.

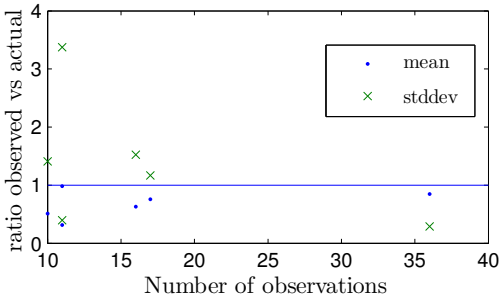
X2 Dual Core Processor 4000+ 2110 MHz with 3 GB RAM). The keystrokes in each query are injected programmatically using the `uinput` [11] interface and the corresponding query packets are collected.

The outcome of the user study is shown in Table 4.1. There is a positive correlation between the number of queries submitted and the number of $p.d.f$. (last column) recovered in $\texttt{DTM}_{\texttt{key}}$ for each participant. This suggests that long term collection of queries would recover far more $p.d.f$. than our user study. The outcome of the methods for type 1 to type 4 are shown in Figure 4.5. Relatively fewer samples were collected for type 3 and type 4 due to the low probability of finding observations at the tail and finding both triplet and keypair accounts. Generally, the mean can be recovered accurately although larger observations tend to result in more accuracy. The variance, on the other hand, is less accurate, particularly for fewer observations. Like the mean, however, the accuracy improves as the observations increase.

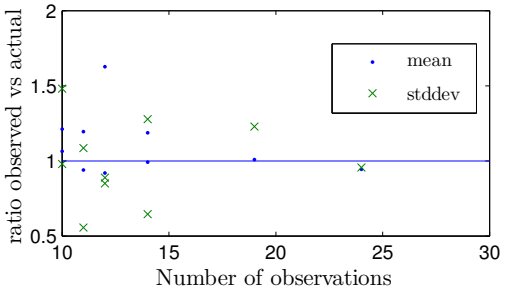


(a) Type 1 recovery

(b) Type 2 recovery



(c) Type 3 recovery



(d) Type 4 recovery

Figure 4.5: Recovery of $p.d.f.$ from various types of packet observations.

4.4.1 The optimal timeout

Given that the current timeout value of 100ms allows the derivation of $\texttt{DTM}_{\texttt{key}}$ (from $\texttt{TM}_{\texttt{pkt}}$), we also investigated the possible countermeasures. These countermeasures are equally applicable to any JavaScript application with rich interactivity that wishes to deny potential adversary the opportunity for UI events harvesting. We conducted a Monte Carlo simulation using the set of keystroke data collected from the user study. We varied the GS timeout and computed the simulated packet timing based on the noise model and the findings of Section 4.2.

Figure 4.6 shows the variation of the count of recovered keypairs and triplets vs. the timeout. Choosing a timeout figure of 200-250 ms eliminates most observations, but the responsiveness is more than halved. Given that in Listing 1, $\texttt{xhr}_{\texttt{m}}$ exits timeout mode whenever there is no keystroke activity in the previous timeout cycle, an alternative is to increase the number of timeout cycles to 3 while keeping the timeout unchanged at 100 ms. This eliminates all observable intervals without

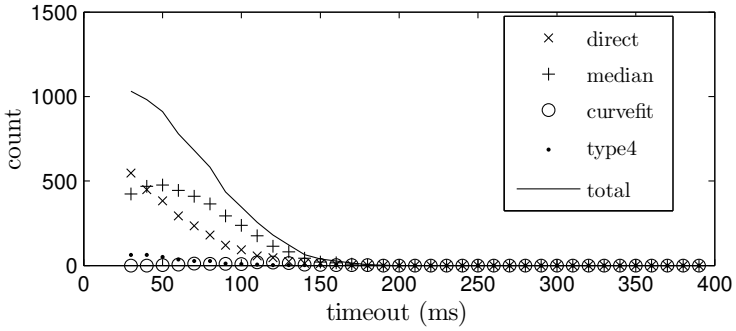


Figure 4.6: Variation of the total recovered keypair or triplets for all users given a particular timeout setting.

affecting the responsiveness.

4.5 Limitations

In our study, the keypair timing model is based on keydown-keydown intervals. Many biometric authentication techniques use such intervals [35, 34, 25]. Our work therefore affects such systems. However, biometric authentication is not limited to just keydown-keydown metrics. Keydown-keyup, keyup-keydown and even keypress pressure are examples of alternatives. For the first 2, active attacks injecting malicious JavaScript code can capture both keydown and keyup, but this is not investigated in this thesis. Keypress pressure, however, cannot be measured by JavaScript applications and is therefore unaffected.

The keystroke injection part of the user study was done on a dedicated machine. We therefore did not model the additional execution delay that may result if the machine is also running multiple compute intensive processes concurrently.

The description of Google Suggestions [4] indicated that it may behave differently in different geographical locations. We did not manage to isolate any geographically specific code during our investigations. This is either due to our limitations or different source code was delivered to different locations. Our findings therefore apply only to geographical locations with similar settings as our evaluation environment.

4.6 Summary

In this chapter, we investigated the recovery of personalized keystroke timing information using Google Suggestions. We found that it is possible to construct a user's typing pattern from the timing of the queries sent over the network. This is of concern because the availability of typing pattern is a prerequisite for (a) achieving the best outcome in timing side channel attacks and (b) imitation attacks on keystroke biometrics. It can also be used to identify users. This suggests that logs recording network traffic of interactive JavaScript applications should be considered confidential and handled accordingly. Otherwise, the operators of search engines as well as proxy server administrators can mine the typing pattern of their users from the traffic logs. We suggest that designers consider alternative options such as multiple timeout cycles to shut down the leak effectively.

Chapter 5

Do typing patterns change under different environment, as well as physical and emotional conditions?

5.1 Introduction

As a form of behavioural biometrics, keystroke biometrics has lower accuracy when compared to physical biometrics, due to the inherent variation in human behaviour. Such accuracy is usually evaluated using the false acceptance rate (FAR), the proportion of anomalous attempts which are wrongly classified as legitimate, and false rejection rate (FRR), the proportion of legitimate attempts which are wrongly classified as anomalous. For the latter, prior research had suggested that typing patterns are not resistant against external influence such as rhythms [30, 46], emotional changes [37] and posture [26].

In this chapter, we investigate the usability of keystroke biometrics under various external influencing factors and conditions. We identified a list of common conditions that could affect user typing patterns and investigate the extent to which the FRR is affected. We evaluate keystroke biometrics under factors which may have short-term effects (listening to fast music, application of a plaster on fingers,

using ergonomic keyboard, using standard external keyboard, typing under time pressure, typing under low light, and typing while standing) and long-term effects (typing after a one-hour-long session of gym or playing computer games). We measure the change in FRR using two of the best classifiers [25, 44] (as evaluated or recommended by various surveys [40, 26]).

The organization of this chapter is as follows. Design considerations and approach to our investigation are described in Section 5.2. Following that, we described the experimental setup in Section 5.3. We present our results in Section 5.4. Section 5.5 discusses some of our findings from our study. Finally, we discuss the limitations and summarise this chapter in Sections 5.6 and 5.7 respectively.

5.2 Design considerations and approach

As discussed in Section 5.1, we are particularly interested in an investigation of the usability and change of FRR in keystroke biometrics. High FRR results in usability problems because legitimate users would not be able to access the system or find it very frustrating to do so. This results in security problems, because it tempts users to bypass the system and results in pressure to tune the system in ways that compromise security. For example, users may simply leave their screens unlocked if the authentication process is too inconvenient. For this reason, it is important to identify usability issues. This section documents the considerations that affect our measurement of usability and our experimental approach.

5.2.1 Participant fatigue

Chief among our concern is participant fatigue. In order to make statistical comparison, we need to collect a sufficient number of samples. However, a large number of samples may result in participant fatigue and skew the results, making analysis

difficult. A field trial collecting keystrokes would be the most ideal alternative in terms of eliminating such issues, but it has its own downside: (a) Different users are likely to use different passwords. So the difficulty of password typing and its length introduces complications into in comparing the results. (b) Capturing user keystrokes as they type their password raises major concerns on confidentiality. (c) A much longer experimental duration is needed. (d) A large amount of effort is required from participants to report their usage conditions consistently and timely.

In a laboratory environment, using a random userid and password will not raise these concerns. We mitigate the fatigue concern in the following manner: (a) The collection periods were short and we limit the number of samples in each collection to 55. Based on a preliminary analysis, this takes no longer than 8 minutes while providing an FRR granularity of 0.018. (b) We have multiple collection periods in each session by introducing breaks of around 10 minutes in between consecutive collection periods.

We plan for post experimental analysis to determine the validity of our experimental structure. The outcome is presented in Section 5.4.

5.2.2 Feedback vs. without feedback

In practical authentication scenarios, the user knows if the last login attempt succeeded or failed. On the other hand, during keystroke biometrics enrolment, the data can only be collected without feedback.

The focus of this chapter is to investigate the success of keystroke biometrics user authentication after various user activities and under various conditions. To provide a more realistic evaluation, we decided to collect the data in our experiments with feedback. Existing literature [25, 44] had not explicitly stated whether data was collected with or without feedback, although it seems likely that collection occurred under similar conditions as the enrolment, i.e., without feedback.

We are interested to know the effects on FRR when feedback is provided. An

additional collection period after enrolment is added to collect typing patterns without feedback. This allows us to evaluate whether existing evaluation methods that do not provide feedback are sound. The outcome is presented in Section 5.4.

5.2.3 Choice of conditions

There are a large number of possible activities and conditions that could potentially induce changes of FRR in authentication. In this section, we describe those that were chosen and the rationale.

Background music and noise

Background music occurs fairly commonly in certain workplaces or during celebratory period. Certain users may prefer to work while listening to music. Certain environments may also be noisier under various circumstances. Prior research by Cho et al. and Hwang et al. shows that if a rhythmic music is played and users are instructed to actively follow the rhythm when typing, there is a change in their typing pattern [30, 46].

In the scenarios under consideration, users are under passive influence. It is possible that a similar though reduced effect exists. Although it would be interesting to determine if background noise level can be measured through user typing pattern, we decided that for an initial investigation, background music would be more suitable as it is in our opinion a more common scenario. Complicating this decision is the abundant choice of musical tracks. We chose for this experiment, the fast paced 'The Dark Knight Rises' soundtrack by Hans Zimmer.

Low lighting and time pressure

Low lighting conditions arise in meeting rooms darkened for presentation or voluntary blackout events such as Earth Hour. For users who need to orientate their fingers by looking at the keyboard but who are not using a backlit keyboard, it may

slowdown their typing.

Time pressure is a very common emotional condition of modern life that may affect the typing pattern. Prior research by Khanna et al. [37] showed that it is possible to infer changes in the emotion from the typing pattern. We ask in this chapter whether the FRR change is significant when there is a change in emotion due to time pressure. However, the inducement of time pressure needs to be done without causing the participants distress or causing them to type haphazardly. Our mechanism is to display a ticking timer and to inform the participants to complete their typing before the timer runs down while offering a reward for the participant with the lowest FRR.

Another motive in including these conditions is to answer the question – during an emergency in a critical infrastructure environment can an operator working under emergency backup lights and time pressure unlock his control station?

Plaster

A plaster applied around a finger is a common remedy for minor ailments. Although its occurrence may be rare, once the plaster is applied, any effect on the typing pattern persists until the removal of plaster or the recovery of the underlying ailment. It is therefore interesting to evaluate its effect because existing evaluation methods are likely to exclude it.

Standing

We included standing because of past experience where it was observed. Such a scenario may also arise in field work where standard office amenities are not expected to be replicated. It is known that posture affects typing pattern [26]. Our focus in this chapter is the quantitative change.

Change of keyboard

Our emphasis on measuring change in FRR when there is a change of keyboard is quantitative. This scenario is interesting because the outcome determines whether the keystroke biometrics database needs to be trained for different input devices. BYOD (Bring Your Own Device) developments, shared workstations, and even helpdesk technicians who may need login access to user devices are examples where a quantitative analysis is useful.

We include in our experiments only participants who are laptop users. Our experiments measure the change in FRR when they switch from the laptop keyboard to a standard external keyboard or an ergonomic keyboard.

Computer gaming and physical exercise

Both computer gaming and physical exercise are very common activities. Both induce emotional changes and possibly physical muscle strain. For our experiments, we look for participants who already have existing gaming or exercise habits. Experimental structure is designed to fit in with their routine rather than having the researchers dictate to them what they should do. For exercise in particular, we need to emphasize this point to the participants to avoid increasing the chance of injury.

For the gaming and exercise conditions, there is a possibility that the effect is long lasting and persists well beyond the end of the activity. We included an experiment to measure the FRR change after different rest periods.

5.2.4 Choice of userid and password

In Chapter 3, we had shown that keystroke biometrics is susceptible to imitation. An investigation of usability (FRR) is orthogonal to the investigation of imitation (FAR). That is, keystroke biometrics is practical only if further research can identify solutions to both usability and imitation problems. We chose as password the string `ths.ouR2` which had been shown to be more difficult to imitate (see Chapter 3).

This password also fits the complexity criteria in user environments and is in our opinion not atypical of actual user passwords.

We chose `user1024` containing lower case alphabets and some numbers as the userid. We acknowledge the great diversity of userids in use. However, we have the constraint of choosing only one userid for all participants to avoid introducing the effect of typing difficulty into the sample data. We also need to avoid choosing a userid (e.g., based on a common name) that may be more familiar to some participants than others. The string `user1024` represents a good compromise in our opinion.

5.2.5 Choice of classifier

We compute the results using two classifiers. We chose the Manhattan (scaled) classifier by Araujo [25] because it was identified as the best classifier in a survey by Killourhy and Maxion [40]. We chose for the second classifier the bioinformatics based classifier by Revett [44] which was identified in another survey by Banerjee [26] as a good classifier.

For the experiments involving feedback, it would be unrealistic to provide feedback from two classifiers at the same time. We chose arbitrarily to provide feedback for only the Manhattan (scaled) classifier.

5.3 Experiment

In this section, we describe our experimental study setup given the considerations discussed in Section 5.2. The basic idea of the experiments is to engage the participants in certain activities and conditions, and to measure the FRR of their authentication attempts. The whole study is divided into three sessions: s1, s2, and s3. s1 was conducted to build an anomaly dataset as a requirement for training the classifier. s2 and s3 were conducted in a lab environment. s2 involved experiments that did not have any long-term effects on typing patterns while s3 involved experiments

that may have long-term effects. Figure 5.1 shows the various stages and flow of each session. Note that s2 and s3 were not performed on the same day.

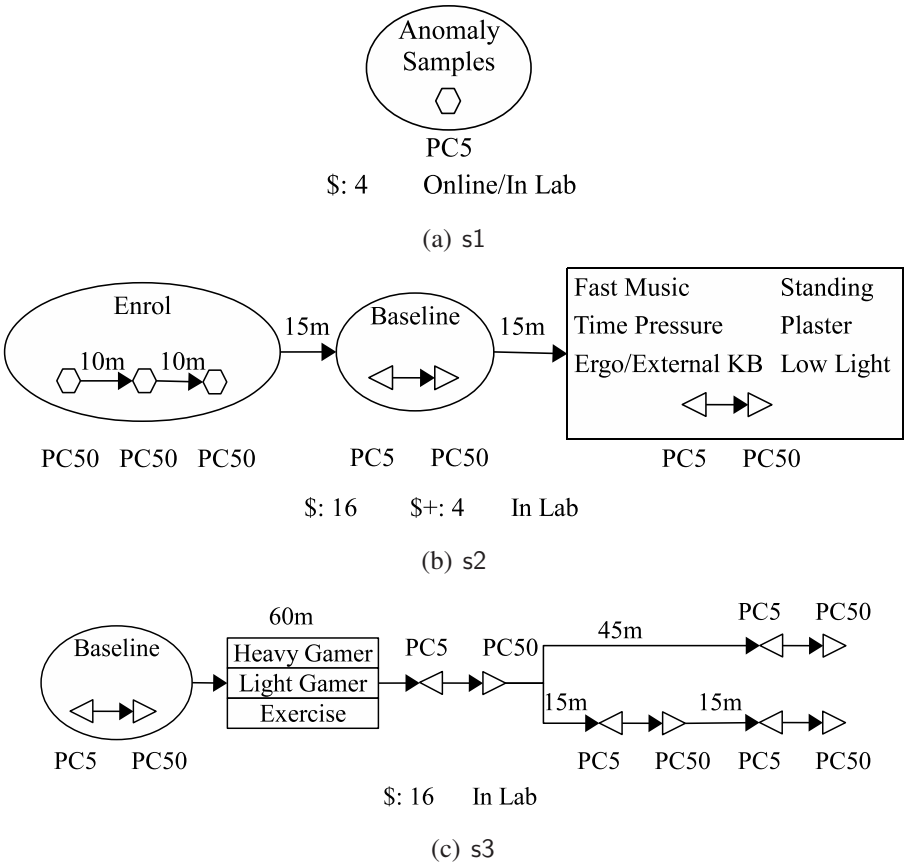


Figure 5.1: Experimental structure — PCn: Type userid and password n times, ⬡: No feedback to user on the acceptance of their typing pattern (by the classifiers), ◁: First users perception was asked and then provided the feedback, ▷: Feedback is provided to user

5.3.1 Participants and Setup

We recruited a total of 111 students in our tests. We received IRB approval from our university, and compensated the participants for completion of the entire set of tasks. Table 5.1 summarizes the participant demographics of our study.

Experiment	Male	Female
Background Noise	11	7
Low Light	16	7
Time Pressure	10	8
Plaster	9	8
Standing	14	4
Ergonomic KB	4	4
Standard External KB	6	3
Exercise with 45 min rest	14	4
Exercise with 15 min rest	11	5
Heavy Gamer with 45 min rest	16	4
Heavy Gamer with 15 min rest	11	7
Light Gamer with 45 min rest	9	9
Light Gamer with 15 min rest	9	12

Table 5.1: Demographics

5.3.2 s1: Anomaly dataset collection

The purpose of s1 was to build an anomaly dataset to train the classifiers. Figure 5.2 shows the interface used. This part of the study was conducted online and required the participants to type in a userid and password (provided by us) via our web interface. As some participants were unable to access the web interface due to technical glitches (e.g., browser compatibility), they completed this task in a lab environment prior to the start of s2. However, the data collected in the lab was not included in the building of the anomaly dataset (too late to be included for use in subsequent sessions). Samples were collected without providing any feedback to the user on their typing consistency.

5.3.3 s2: Experiments with short-term effects

s2 was designed to collect training data for each participant as input to the classifier, and to investigate the effectiveness of keystroke biometrics under certain scenarios that we consider to have short-term effects. s2 was conducted in a lab environment post s1.

Participants were first required to answer a questionnaire capturing their emo-

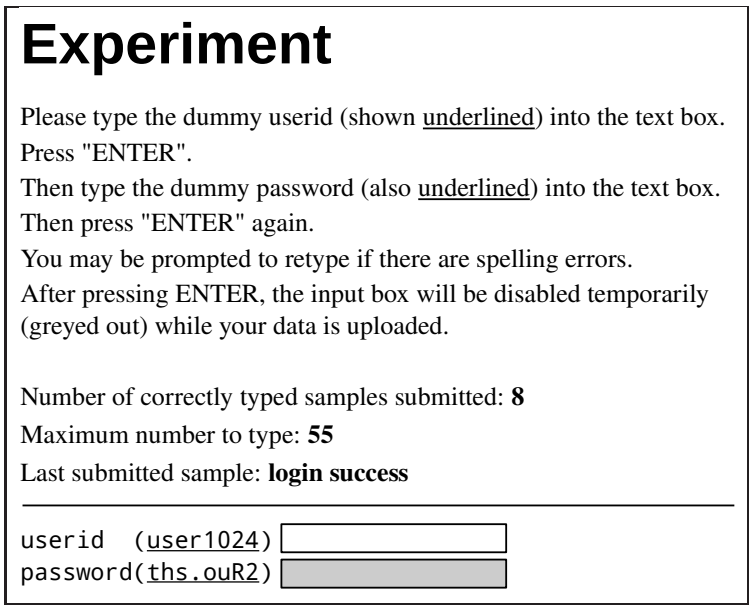


Figure 5.2: User Interface for data collection

tional and physical state. This was followed by a creation of the training set which had participants type 50 samples of the userid and password pair thrice with a gap of 10 minutes between each set. As discussed in Section 5.2, we designed the session to ensure that each sample collection interval is short. We used the first 100 samples to train the classifier and the next 50 to calculate the FRR under no-feedback conditions. Note that participants were not provided any feedback on the consistency of these samples.

Next, after a rest of 15 minutes, participants were again asked to type 55 samples but with feedback. Thus the participants were aware of their typing consistency. We use the submitted samples to calculate the FRR, with feedback, for that user. This FRR is compared with the FRR (under no-feedback conditions) to observe the impact of feedback (if any) on a participant's FRR. We show the result of this evaluation in Section 5.4.

Finally, after another 15 minutes of rest, participants were asked to submit 55 samples this time while performing one of the activities (listening to fast music, putting a plaster on the right index finger, using an ergonomic keyboard, using a standard external keyboard, typing under time pressure, typing under low light, typ-

ing while standing) (see Section 5.2.3).

Participants were encouraged to type the samples consistently and were further incentivised by giving the participant with the most accurate typing in each experiment an additional $4. We hope that this step minimised any intentional inconsistencies in a participant's typing pattern. At the end of s2, participants answered a questionnaire on the task experience. Note that samples with any typographical errors were not collected, but the count and order of such samples were recorded. Participants are required to retype whenever they make a typographical error.

5.3.4 s3: Experiments with long-term effects

The aim of s3 was to investigate the effectiveness of keystroke biometrics under scenarios that we consider to have relatively long-term effects. Similar to s2, participants first answered a questionnaire stating their emotional and physical states. We then collected the baseline typing pattern (55 samples) of each participant.

After the collection of baseline samples, participants were asked to perform the assigned activity (either playing a computer game or going to the gym). The activity lasted 60 minutes following which participants returned to the lab to type 55 more samples as in s2.

After this sample collection, participants were divided into two groups. The first group was allowed to rest for 45 minutes following which 55 more typing-samples were collected. This was done to observe the effect of a long rest duration on the accuracy of their typing. Users were not allowed to use their smartphones or laptops during this rest period. They were, however, allowed to read magazines and other periodicals (provided by us) to pass time. This was to ensure that no typing was performed during the rest period and thereby minimise any inconsistencies in subsequent sample collection.

The second group was allowed to rest only for 15 minutes after which they were asked to type 55 samples. This cycle of a 15 minute rest followed by typing 55

samples was repeated again. Here, we hope to observe the effect of intermittent rest on the accuracy of their typing. As in the first group, the participants did not perform any other typing besides the samples collected. At the end of s3, participants answered a questionnaire on the task experience.

5.4 Results

In this section, we present the results of our experiments. In our analysis, the significance level is computed using a paired student's t-test. Unless stated otherwise, a single-tail t-test is used. The significance level follows a standard text book classification [36], where the descriptions 'highly significant', 'significant' and 'weakly significant' are used to describe p-values in the ranges of [0, 0.01), [0.01, 0.05), and [0.05, 0.10), respectively. The criterion for pairing is based on the user.

5.4.1 Participant fatigue and feedback

In Section 5.2, we raised two concerns relating to participant fatigue and the provision of feedback that may affect the validity of our results. We now evaluate each of these concerns based on the results collected from the experiments.

Number of samples collected

First, we want to know whether the number of samples typed (55) is overwhelming and causes those typed later to be significantly different from those typed earlier. Figure 5.3 shows the relation between the average FRR for all participants and the sample index for the s2 baseline experiment.

The coefficient of correlation between FRR and sample index are 0.4339 and 0.3994 for the Manhattan (scaled) and bioinformatics classifier, respectively. This means that there is statistical support for the intuition that latter samples tend to have higher FRR than earlier samples. The slope of the regression lines through the two sets of data are, however, just 0.0007885 and 0.0007228, respectively. Therefore,

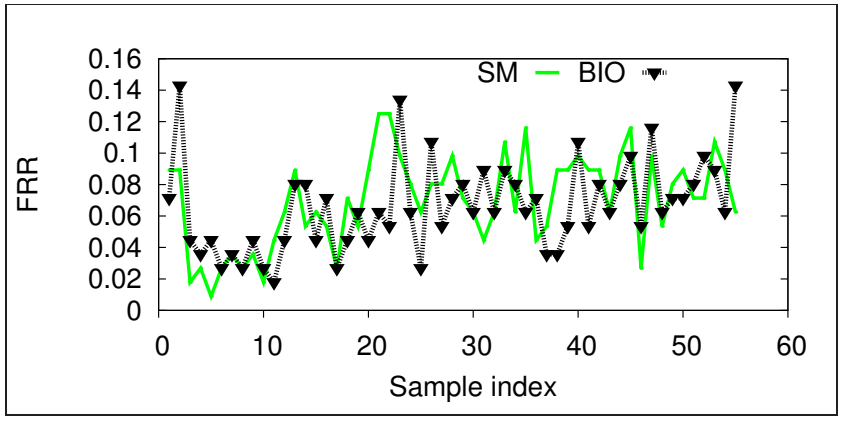


Figure 5.3: Effect of number of samples collected on the FRR

the effect is not overwhelming. The time taken to collect 55 typing samples range from 3.5 min to 12.2 min, with an average of 5.3 min. The typing time is therefore short relative to the rest period.

Feedback vs. no feedback

We compare for all participants the FRR of the samples collect without feedback with their session baseline which was collected with feedback. For the Manhattan (scaled) classifier, the mean FRR of samples collected with feedback is 0.01106 higher than that without feedback. However, the difference is not statistically significant. On the other hand, the difference for the bioinformatics-based classifier is 0.01427 and is highly significant.

If participants ignored the feedback, there should not be any significant differences. The results therefore suggest that users are not ignoring feedback and, surprisingly, are affected by it in a negative way. A possible explanation is that when users are trying hard to make sure they are typing "correctly", their typing pattern actually becomes slightly different from their enrolment pattern. This factor is not accounted for in existing literature and justifies our decision to collect typing samples with feedback.

5.4.2 Change of input devices

Table 5.2 shows the change in FRR when the input device is changed from the laptop keyboard to an external keyboard or an ergonomic keyboard. For both cases, the changes are large and highly significant. The Manhattan (scaled) classifier is more adversely affected compared to the bioinformatics based classifier. Most people are unfamiliar with ergonomic keyboards and that has resulted in an extremely high FRR.

Exp	N	SM-B	SM-D	BIO-B	BIO-D
Ext. KB	9	0.1010	0.1697	0.0747	0.0889
Ergo. KB	8	0.0818	0.7500	0.0841	0.6182

Table 5.2: Effect of device factors on typing pattern. N: number of participants. SM-B: Baseline FRR (Manhattan (scaled)). SM-D: Increase in FRR (Manhattan (scaled)). BIO-B: Baseline FRR (Bioinformatics). BIO-D: Increase in FRR (Bioinformatics). All changes are highly significant.

It is clear from our results that patterns obtained using one keyboard should not be used to authenticate users when typing on a different type of keyboard. Given that in both home and corporate environments, owning multiple devices such as both a PC and a laptop is a common scenario, the results raise the question of how keystroke biometrics deployment should handle multiple input devices. A straightforward solution is to enrol the user multiple times, once for each device. However, it remains unclear whether a user can maintain a consistent typing pattern for each device when one is used only occasionally. We do not explore this issue in this thesis and leave it as future work.

5.4.3 Effect of various conditions on FRR

Figure 5.4 summarizes the effect of the conditions discussed in Section 5.2. We can see that other than the fast music experiment under the Manhattan (scaled) classifier, all other conditions exhibit varying degrees of statistical significance. Different classifiers and groups of participants are affected differently by these conditions. The

standing condition, plaster, and exercise are among the factors making the larger differences, although such changes in FRR are not as big as those observed with the change in input devices.

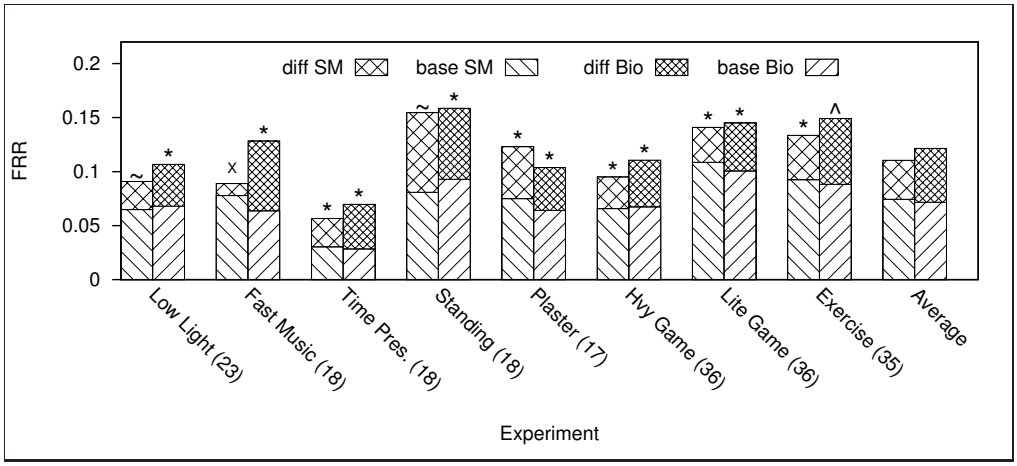


Figure 5.4: Effect of various conditions on typing pattern (baseline and increase in FRR). Numbers in brackets indicate the number of participants. $\wedge$: highly significant. *: significant. $\sim$: weakly significant. x: not significant.

The result for fast music was interesting. The change in FRR for the Manhattan (scaled) classifier is minor and statistically insignificant. This contrasts with that observed for the bioinformatics classifier. An explanation can be made based on a shape based interpretation of these two classifiers (see Section 2). Any changes in typing pattern corresponds to a multi-dimensional movement. It is therefore possible to move beyond the enclosure of one classifier region while remaining in the other classifier region.

Participants in the fast music experiment reported that they typed faster. Referring to Figure 2.3, this corresponds to a diagonal movement towards the approximate direction of the origin in a 2-d setting. The exact direction depends on the relative extent of movement in each dimension. This means that for the fast music experiments, the participants' typing pattern moved out of the bioinformatics region, but remained in the Manhattan (scaled) region.

5.4.4 User and classifier dependency

We now investigate whether the penalty due to the experimental conditions depends on the classifier or participant. There are two questions of interest. Firstly, for each user, if the baseline performance of 1 classifier is better, does it imply a lower penalty when the user is influenced by an experimental condition. Secondly, for each classifier, if a user has low FRR for his baseline, does it imply a lower penalty when the user is influenced by an experimental condition?

Table 5.3 tabulates the answers. The first row answers the first question, while the second and third rows answer the second question for each classifier. In all three cases, there is little correlation (very small coefficient of correlation), suggesting that each condition has varied effects on the users.

Relation	Coeff. Correlation
SM vs BIO	-0.0891
Baseline vs penalty (SM)	-0.0519
Baseline vs penalty (BIO)	-0.0633

Table 5.3: Are different classifiers or users more resistant to the experimental effects? SM: Manhattan (scaled) classifier. BIO: Bioinformatics classifier.

5.4.5 Effect of rest and recovery

Figure 5.5 shows the effect of a 45-minute rest on the FRR of participants in the heavy gaming, light gaming, and exercise scenarios under different classifiers. With the exception of heaving gaming under Manhattan (scaled), the remaining scenarios are not statistically significant.

Light gaming showed a decrease in FRR after the rest period. However, both heavy gaming and exercise participants showed a surprising increase in the FRR. One hypothesis is that the long rest period, in which the participants were requested not to use their phones or computers, resulted in boredom and affected the experimental data. We ruled out this possibility because such an increase was not seen

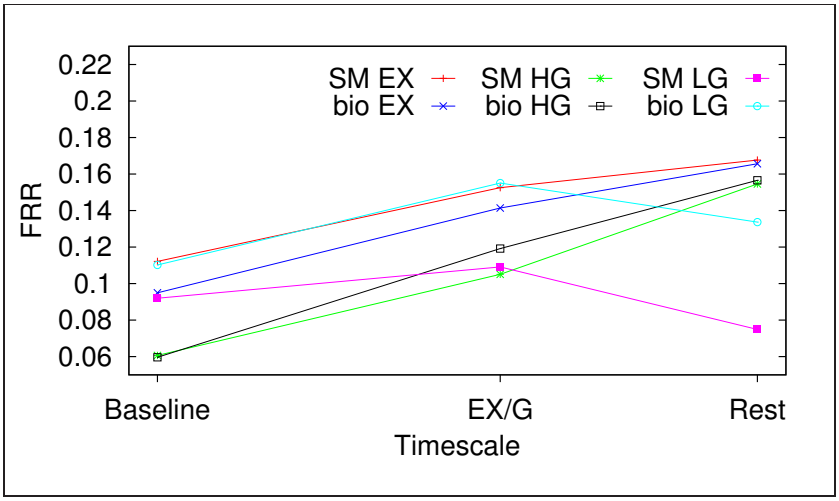


Figure 5.5: Effect of rest on exercise, heavy gaming and light gaming. SM: Manhattan (scaled) classifier. Bio: Bioinformatics classifier. EX: Exercise. HG: Heavy gaming. LG: Light gaming. EX/G: Measurement taken after exercise or gaming. Rest: Measurement taken after 45 min rest.

for light gaming. A more likely hypothesis is that certain effects of heavy gaming and exercise persisted after the rest. Another experiment designed to specifically answer this question would be needed to clarify these findings.

The results also suggest that the effect of heavy gaming and exercise on typing patterns is relatively long lasting. Referring to Figure 5.5, FRR for both cases did not recover to the baseline level within the 45-minute rest period.

In our experimental design, we also catered for 2 short rest periods of 15 minutes each. However, as none of the results obtained are statistically significant and there are no obvious trends seen for the short rest, we exclude them from our reporting.

5.4.6 Reality versus Perception

Unlike fingerprint readers and retina scanners that have gained widespread acceptance as a result of motion pictures, keystroke biometrics is still in its infancy. In fact, from our participant recruitment survey, out of 343 undergraduate students that responded only 6% had heard of keystroke biometrics as an authentication mechanism. We therefore believe that it was pertinent to also capture user perception (through an easy 5-point Likert scale) of keystroke biometrics across the different

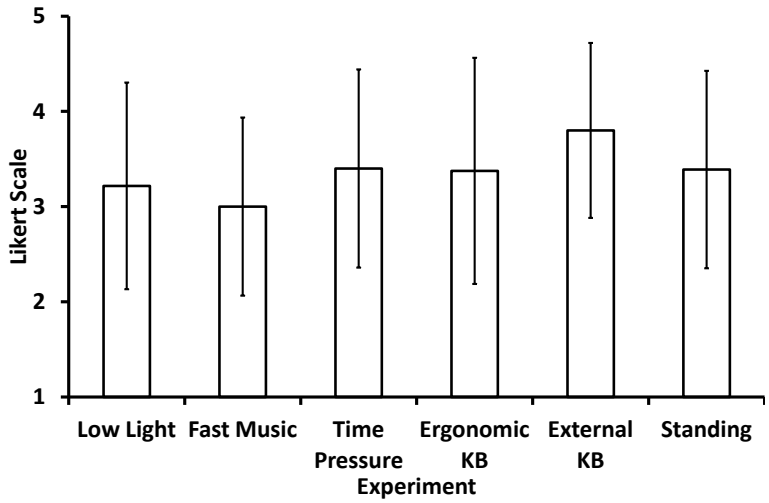


Figure 5.6: (Perception) Effect of various activities on typing pattern.

stages of the experiment. This allows us to gauge whether users are able to detect changes in their own typing pattern. If this is possible, it allows mitigation measures against the usability issues (such as activating a backup authentication mechanism).

Figure 5.6 shows the perception participants have on the impact of the different s2 activities on their typing pattern. On average, most participants perceived that the various activities would indeed have an impact on their typing pattern. This matches the observed increase in FRR across the different activities (with the exception of fast music).

Figure 5.7 shows the perception participants have on the impact of the different s3 activities on their typing pattern. Contrary to the FRR measurement results, most participants were very confident that they could type normally after having rested for 45 minutes. While this was indeed the case with light gaming, the other two scenarios did not observe a decrease in FRR. As suggested before, participants were perhaps unaware that effects of heavy gaming and exercise persisted even after resting. We also observe that heavy gaming and exercise participants suggested that their typing pattern would be impacted post gaming or exercise while light gaming participants did not feel there would be any change.

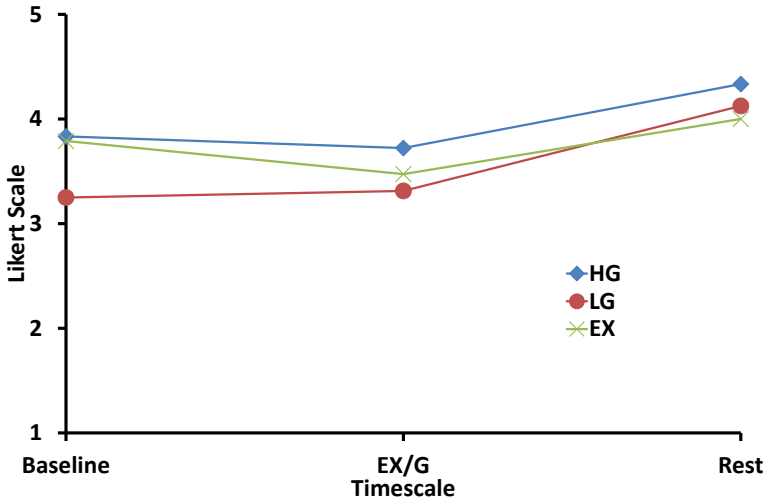


Figure 5.7: (Perception) Effect of exercise, heavy gaming and light gaming (with a 45 minute rest) on typing pattern.

5.5 Discussion

In this section, we discuss some of the other interesting findings that were observed in our experiments.

5.5.1 Lockout and annoyance

Even without keystroke biometrics, there is an FRR due to typing when users remember their passwords correctly, but typed them wrongly. It is not a serious problem in practice because if a user made 2 typing errors on the first 2 tries, he can slow down and ensure the 3rd attempt is correct. With keystroke biometrics, such a strategy is no longer workable. If a user slows down, he increases the FRR due to keystroke biometrics. If he does not slow down, he increases the typing FRR. Implementing keystroke biometrics therefore increases the overall FRR of the system by more than just the keystroke biometrics FRR.

In this subsection, we investigate what the impact of this increase would be, in terms of both the likelihood of account lockout and the user reaction. To estimate the effect of the overall FRR on the user experience, we conducted a Monte Carlo simulation. We divide the users into groups based on the conditions shown in

Figure 5.4. The simulation first picks a group, then a user within that group, both randomly. The overall FRR is computed for 2 cases: (a) baseline only, without any influence of the selected condition and (b) baseline + condition. The FRR takes into account both typing errors and keystroke biometrics error. The simulation computes for each login session the number of tries required to login.

Figure 5.8 shows the cumulative percentage of users who will encounter a lockout after a certain number of login sessions. For example, approximately 30% of users will encounter a lockout before their 50th authentication session if the lockout policy is 3 attempts and if half of the time, they are under the influence of one of the conditions of Figure 5.4.

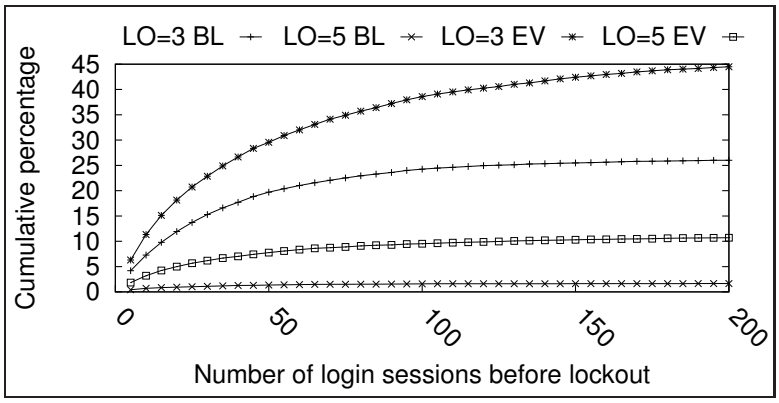


Figure 5.8: Number of authentication sessions before user gets lockout. LO: Lockout policy. BL: Baseline (no environmental factors). EV: Environmental factors in effect 50% of the time.

Figure 5.8 shows that keystroke biometrics is not for everyone. Even with a lockout policy of 5, approximately 10% of users will find keystroke biometrics very frustrating to use. This suggests that a change of research direction is necessary, from finding classifiers that work well for all users, to finding classifiers that work well for the majority. The unsuitable minority requires an alternative login mechanism.

Figure 5.8 also shows that deployment of keystroke biometrics must come with an adjustment in lockout policy, but at a cost in security. Keystroke biometrics was considered as a security mechanism to enhance the security of weak password [43].

However, if the password is weak, the lockout is adjusted to 5 and given the typical range of FAR obtained from surveys [40], the added security is modest. Table 5.4 shows the chance that an attacker can break into a system if he already knows the password. For an FAR value of 0.05, the overall FAR is 0.23 if the attacker is allowed to make 5 attempts. In other words, approximately 1 in 4 such attempts will succeed.

	FAR			
Lockout Policy	0.001	0.01	0.02	0.05
1	0.00100	0.01000	0.02000	0.05000
2	0.00200	0.01990	0.03960	0.09750
3	0.00300	0.02970	0.05881	0.14263
4	0.00399	0.03940	0.07763	0.18549
5	0.00499	0.04901	0.09608	0.22622

Table 5.4: Chance of attacker success before lockout (detection).

Other than lockout, we are interested in possible user annoyance. We asked the participants for their reactions if, on average, they need to type their credentials multiple times before they can login. Table 5.5 summarizes the responses.

	2 tries	3 tries	4 tries	5 tries
Okay	37.62	5.94	0.99	0.00
Mildly annoying	51.49	51.49	16.83	3.96
Very annoying	10.89	42.57	82.18	96.04

Table 5.5: User response (%) to the number of tries require for authentication. E.g. 10.89% of users are very annoyed if they need 2 tries to login.

Figure 5.9 shows the cumulative percentage of users who will encounter a login session in which they succeed but find very annoying. For example, more than 40% of users will encounter an annoying session before their 50th authentication session if the lockout policy is 5 attempts and if they are under the influence of the conditions in Figure 5.4 half of the time. The occurrence of annoyance is lower when the lockout policy is 3 because users get locked out before they have a chance to succeed but feel annoyed. This shows that even after solving the lockout problem, keystroke biometrics still has to contend with user frustration over the multiple login

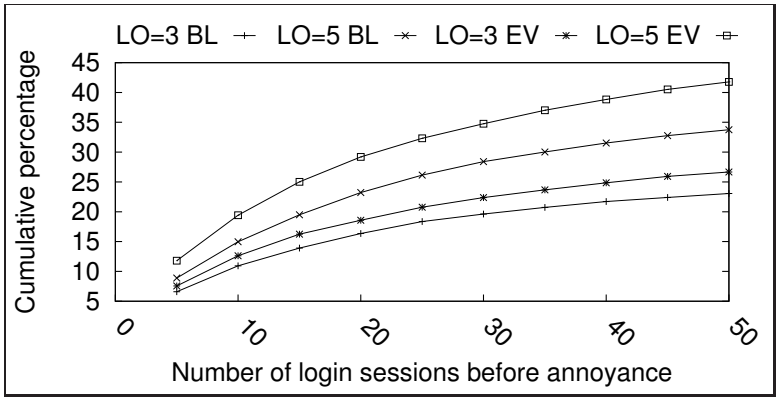


Figure 5.9: Number of authentication sessions before user encounters a very annoying one. LO: Lockout policy. BL: Baseline (no environmental factors). EV: Environmental factors in effect 50% of the time.

attempts required.

5.5.2 Best time to collect samples

Inspection of Figure 5.3 shows that the 3rd to the 10th samples of each collection period have the lowest (best) FRR. In other words, user typing patterns are initially inconsistent, stabilize rapidly, then become less consistent for the rest of the collection period. This suggests that enrolment samples can be kept to 10. For classifiers that require larger number of training samples (e.g., [27, 30, 54], it may be fruitful to explore whether generating additional samples using a Monte Carlo simulation based on a few high quality samples would give better results than collecting the samples from the participants directly in a lengthy enrolment.

5.5.3 Change in fingers used for typing

At the end of each experiment, we asked the participants which fingers they used to type the assigned credentials. This information was consolidated and analyzed to determine if the participants changed the way they typed. Note that this information is just an estimate, because the participants were asked at the end of each experiment, not at the end of each sample. Table A.1 (Appendix) shows a summary of the number of participants who switched the fingers used when typing each

keystroke and the fingers involved. Generally, for keys on the left side of the keyboard, switching is between the left ring and left middle, or left middle and left index fingers. A similar pattern is observed for the right side of the keyboard for the corresponding right fingers. For keys near the middle, the switching is between the left and right index fingers. Any switching can only increase the inconsistency of the typing. An interesting research question is whether there exists a sufficiently large set of passwords for which finger switching is minimised. If such a set can be found, the password space is likely to be much smaller. However, it offers the possibility of trading password space for typing consistency. For keystroke biometrics deployed in a continuously monitoring mode, words that are more likely to be consistently typed would result in better FRR. Investigating the answer to these questions is however out of the scope of this thesis.

5.6 Limitations

We discuss some limitations of our experiments in this section. In both sessions s2 and s3, we asked participants to abstain from exercise and gaming for the prior 24 hours, to avoid introducing any possible influence from these activities. We are dependent on the participants to observe these restrictions in good faith. Similarly, for responses such as those reported in Table 5.5, A.1 and Figures 5.6,5.7 we depend on the participants to answer accurately and submit correctly.

In our experiments, we collected the typing samples via a web based interface using participant's own laptop. The latter is because we want to ensure they are using a familiar keyboard layout. However, this also implies that the input devices comes in a variety of OS and hardware. In particular, the timing granularity varies. For Windows XP systems, this can be up to 16 ms. In the literature, the timing granularity varies from 0.2 ms [40], 1 ms [25] and 10 ms [47, 34].

A related issue is a selection bias on the hardware. Before each experiment, we run a small computationally intensive test on the participant's laptop. The aim is to

detect participants who are submitting their keystrokes through a browser in a virtual machine [24, 18], or whose laptop is heavily loaded with background processes. The timing granularity is affected under such cases. For laptops which fail this test, we asked the participants to stop unnecessary background processes and remove unneeded hardware, such as wireless mouse. If they fail the test again, they are not allowed to participate. Netbooks with relatively low computational power are the most affected. The selection bias allows us to eliminate these influences on our results. However, in practice, it would be naive to make such assumptions on the hardware and background computational load.

In all our experiments, a single password is used. That password was chosen to be difficult to type, which unfortunately increases the chance of typing it wrongly, and in turn increases the FRR. An easier password would have fewer issues, but we believe our chosen password is more representative of those mandated by corporate security policies.

5.7 Summary

We found that various environmental factors such as playing computer games, exercise, lighting conditions, fast music, emotional pressure, and even the application of a plaster on the fingers result in various increase of FRR. Different users are also affected differently. Certain users are shown to be unsuitable for keystroke biometrics. This suggests that instead of a research direction that attempts to find classifiers that work for all users, a more practical approach may be to find users who have low baseline FRR and are tolerant towards environmental changes. This implies a hybrid approach where keystroke biometrics is implemented for only a subset of users, and is complemented by an alternative authentication mechanism.

Chapter 6

Conclusions

We showed in the preceding chapters that:

1. Contrary to the beliefs of prior studies, when a victim's typing pattern is known, imitation is possible in static mode. This affects the FAR of keystroke biometrics systems. Furthermore, when the number of attackers and victims are sizeable, the chance of a natural collision in typing pattern (without any imitation training) is significant.

2. The inter-keystroke timing of typing patterns can be recovered from the timing of the unencrypted query packets sent over the network. This compromises the secrecy of typing patterns in continuous mode.

3. Various environmental factors such as playing computer games, exercise, lighting conditions, fast music, emotional pressure, and even the application of a plaster on the fingers result in various increase of FRR in static mode. Different users are also affected differently. Certain users are shown to be unsuitable for keystroke biometrics.

Therefore the deployment of keystroke biometrics systems faces a security and usability problem in static mode, while continuous mode typing patterns may be leaked by interactive JavaScript applications. This suggests that care should be exercised when deploying such systems. In our opinion, the limitations of keystroke

dynamics make it more suitable for use in an intrusion detection system (whose FAR and FRR requirements are less stringent), rather than in an authentication system.

Bibliography

[1] About Google Instant. http://goo.gl/62qhC.

[2] AdmitOne Security. http://goo.gl/ZUeOG.

[3] AuthenWare. http://www.authenware.com.

[4] Autocomplete. http://support.google.com/websearch/bin/answer.py?hl=en&answer=106230.

[5] Behaviosec. http://www.behaviosec.com.

[6] biochec. http://www.biochec.com.

[7] Deepnet security. http://www.deepnetsecurity.com.

[8] Delfigo security. http://www.delfigosecurity.com.

[9] Dibisoft. http://www.dibisoft.com.

[10] DOM Events. http://en.wikipedia.org/wiki/DOM_events.

[11] Getting started with uinput: the user level input subsystem. http://thiemonge.org/getting-started-with-uinput.

[12] Id control. http://www.idcontrol.net.

[13] imagicsoftware. http://www.imagicsoftware.com/.

[14] Intensity analytics. http://www.intensityanalytics.com/.

[15] JSBeautifier. http://jsbeautifier.org/.

[16] keytrac. http://www.keytrac.de/.

[17] Normal Sum Distribution. http://mathworld.wolfram.com/NormalSumDistribution.html.

[18] Oracle VM VirtualBox. https://www.virtualbox.org/.

[19] plurilock security solutions inc. http://www.plurilock.com/.

[20] Probayes. http://www.probayes.com.

[21] Psylock. http://www.psylock.com.

[22] Squid: the proxy server. http://www.squid-cache.org/Intro/.

[23] TCPDUMP: the command-line packet analyzer. http://www.tcpdump.org.

[24] VMware Virtualization Software. http://www.vmware.com/.

[25] L.C.F. Araujo, L.H.R. Sucupira, Jr., M.G. Lizarraga, L.L. Ling, and J.B.T. Yabu-Uti. User authentication through typing biometrics features. Trans. Sig. Proc., 53(2):851–855, February 2005.

[26] S.P. Banerjee and D.L. Woodard. Biometric Authentication and Identification using Keystroke Dynamics: A Survey. Journal of Pattern Recognition Research, 7:116–139, 2012.

[27] S. Bleha, C. Slivinsky, and B. Hussien. Computer-access security systems using keystroke dynamics. IEEE Trans. Pattern Anal. Mach. Intell., 12(12):1217–1222, December 1990.

[28] Michelle Boatwright and Xin Luo. What do we know about biometrics authentication? In Proceedings of the 4th annual conference on Information security curriculum development, InfoSecCD '07, pages 31:1–31:5, New York, NY, USA, 2007. ACM.

[29] Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, SP '10, pages 191–206, Washington, DC, USA, 2010. IEEE Computer Society.

[30] Sungzoon Cho, Chigeun Han, Dae Hee Han, and Hyung il Kim. Web based keystroke dynamics identity verification using neural network. Journal of Organizational Computing and Electronic Commerce, 10:295–307, 2000.

[31] Simon A. Cole. More than zero: Accounting for error in latent fingerprint identification. Journal of Criminal Law and Criminology, 95(3), 2005.

[32] C. Epp. Identifying emotional states through keystroke dynamics. Master's thesis, University of Saskatchewan, 2010.

[33] B Geller, J Almog, P Margot, and E Springer. A chronological review of fingerprint forgery. Journal of forensic sciences, 44:963–968, 1999.

[34] S. Haider, A. Abbas, and A.K. Zaidi. A multi-technique approach for user identification through keystroke dynamics. In IEEE International Conference on Systems, Man and Cybernetics, SMC 2000, pages 1336–1341, 2000.

[35] Rick Joyce and Gopal Gupta. Identity authentication based on keystroke latencies. Commun. ACM, 33(2):168–176, February 1990.

[36] G. Keller and B. Warrack. Statistics for management and economics. Number v. 1 in International student edition: Wadsworth. Thomson/Brooks/Cole, 2003.

[37] Preeti Khanna and M. Sasikumar. Article: Recognising emotions from keyboard stroke pattern. International Journal of Computer Applications, 11(9):1–5, December 2010. Published By Foundation of Computer Science.

[38] Kevin Killourhy and Roy Maxion. Why Did My Detector Do That?! Predicting Keystroke-Dynamics Error Rates. In Proceedings of the 13th international conference on Recent advances in intrusion detection, RAID'10, pages 256–276, Berlin, Heidelberg, 2010. Springer-Verlag.

[39] Kevin S. Killourhy. A Scientific Understanding of Keystroke Dynamics. Dissertation, Carnegie Mellon University, 2012.

[40] K.S. Killourhy and R.A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In Dependable Systems Networks, 2009. DSN '09. IEEE/IFIP International Conference on, pages 125 –134, Jul 2009.

[41] Michael Mahemoff. Ajax Design Patterns. O'Reilly Media, Inc., 2006.

[42] Fabian Monrose and Aviel D. Rubin. Keystroke dynamics as a biometric for authentication. Future Gener. Comput. Syst., 16(4):351–359, Feb 2000.

[43] Alen Peacock, Xian Ke, and Matthew Wilkerson. Typing Patterns: A Key to User Identification. IEEE Security and Privacy, 2(5):40–47, September 2004.

[44] Kenneth Revett. A bioinformatics based approach to user authentication via keystroke dynamics. International Journal of Control, Automation and Systems, 7:7–15, 2009.

[45] Fred Erlend N. Rundhaug. Keystroke dynamics Can attackers learn someone's typing characteristics. Master's thesis, Gjovik University College, 2007.

[46] Seong seob Hwang, Hyoung joo Lee, and Sungzoon Cho. Improving authentication accuracy using artificial rhythms and cues for keystroke dynamics-based authentication. Expert Systems with Applications, 36(7):10649 – 10656, 2009.

[47] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on SSH. In Proceedings of the 10th conference on USENIX Security Symposium - Volume 10, SSYM'01, pages 25–25, Berkeley, CA, USA, 2001. USENIX Association.

[48] Chee Meng Tey, Payas Gupta, and Debin Gao. I can be You: Questioning the use of Keystroke Dynamics as Biometrics. In Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, CA, Feb 2013.

[49] CheeMeng Tey, Payas Gupta, Debin Gao, and Yan Zhang. Keystroke timing analysis of on-the-fly web apps. In Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, Applied Cryptography and Network Security, volume 7954 of Lecture Notes in Computer Science, pages 405–413. Springer Berlin Heidelberg, 2013.

[50] D. Umphress and G. Williams. Identity verification through keyboard characteristics. International Journal of Man-Machine Studies, 23(3):263–273, September 1985.

[51] Wikipedia. Legato — Wikipedia, the free encyclopedia, 2012. [Accessed 02-August-2012].

[52] Wikipedia. Staccato — Wikipedia, the free encyclopedia, 2012. [Accessed 02-August-2012].

[53] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Authentication Protocol, Jan 2006. `http://www.ietf.org/rfc/rfc4252.txt`.

[54] E. Yu and S. Cho. GA-SVM wrapper approach for feature subset selection in keystroke dynamics identity verification. In International Joint Conference on Neural Networks, pages 2253–2257, 2003.

Appendices

Appendix A

Table summarising changes in typing finger

Finger Used	u	s	e	r	1	0	2	4	t	h	s	dot	o	u	R	2
LI,LM		1	15	16	4		11	13	10		7				13	13
LI,LM,LR		2	3		3		2	1			3	1			1	3
LI,LM,RI								1	3		1					1
LI,LM,RM		1		3		1	2	2		1					4	
LI,LR		5			1		2	2			2				1	3
LI,RI	10			2		1		3	14	18	1			9	11	
LI,RI,RM	3					2			1	2		2	1	2	2	
LL,LR					10											
LM,LR		14	4		19		19				18					28
LM,LR,RM	1					1	1				3					2
LM,RI,RM	4				1			1		2					1	1
LM,RM	2	3	1		2	4	3	1	2	1	4	3	5	1		1
LR,RI	1				1						3	1				1
LR,RR					1	4						2	2			
RI,RM	26					16			2	10		13	10	23	3	1
RI,RM,RR	1					5						3		1		
RI,RR	1					5						3	5	2		
RL,RR						5						5				
RM,RR						12						12	20			

Table A.1: Summary of changes in finger used. Key for left/right: L: Left, R:Right. Key for finger: L: Little. R: Ring. M: Middle. I: Index. T: Thumb. For example the number '26' appears for the row labelled 'RI,RM' and first column labelled 'u'. This means 26 participants reported that when they typed 'u' for the userid 'user1024', they switched between RI (right index finger) and RM (right middle finger). There are 111 participants. Rows with all cells less than 3 are removed for brevity.