

Keystroke dynamics Can attackers learn someone's typing characteristics

Fred Erlend N. Rundhaug
fred.rundhaug@hig.no



Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2007

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Abstract

This master's thesis is about keystroke dynamics in authentication situations. Users typing characteristics are checked in addition to their username and password, increasing the security of password/username authentication systems. It has been proved that keystroke dynamics can differentiate between individuals. We had two experiments in this thesis. The first experiment gathered password samples from 21 participants over a two months period. These samples were used to test various distance metrics. Many distance metrics have been proposed by other authors, we have tested several distance metrics in this thesis, discovering that some of the proposed distance metrics had a poor performance in our experiment.

Some of the authors tried to imitate legitimate users typing characteristics, but the most advanced attacks were shoulder-sniffing techniques. Their conclusion was that keystroke dynamics is resistant against attacks. We know that that it is difficult to learn sports without proper feedback, we assumed that it was the same for keystroke dynamics. We created a more advanced attack, where an attacker gets hold of a users template, and the attacker can see where his or hers typing differs from the legitimate user. By letting attackers use a program that helps them learn other's typing characteristics we were able to answer the important question; *“can an attacker learn someone's typing characteristics?”*

Sammendrag

Denne masteroppgaven er om keystroke dynamics i autentiserings situasjoner. Brukernes taste karakteristikk er sjekket i tillegg til deres brukernavn og passord, dette øker sikkerheten i passord/brukernavn autentiserings systemer. Det er bevist at keystroke dynamics kan skjelne mellom individer. Vi hadde to eksperiment i denne masteroppgaven. Det første eksperimentet samlet passord tastinger fra 21 deltakere over en to måneders periode. Disse tastinger ble brukt til å teste forskjellige distanse metrikker. Mange distanse metrikker er foreslått av andre forfattere, vi testet flere distanse metrikker i denne masteroppgaven, og oppdaget at noen av de foreslåtte distanse metrikkene fungerer dårlig i våres eksperiment.

Noen av forfatterne har prøvd å imitere brukeres taste karakteristikk, men de mest avanserte angrepene var shoulder-sniffing teknikker. Deres konklusjon var at keystroke dynamics er resistent mot angrep. Vi vet at det er vanskelig å lære en sport uten riktig tilbakemelding, vi antok at dette også var tilfelle for keystroke dynamics. Vi lagde et avansert angrep, der en angriper får tak i en brukers template, og angriperen kan se hvor hans eller hennes taster avviker fra brukerens. Ved å la angripere bruke et program som hjelper dem å lære andres taste karakteristikk, klarte vi å svare på det viktige spørsmålet; "*kan en angriper lære seg noens taste karakteristikk?*"

Contents

Abstract	iii
Sammendrag	v
Contents	vii
List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Topic	1
1.2 Research problem	3
1.3 Motivation and justification	3
1.4 Research questions	4
2 Keystroke dynamics	5
2.1 Biometrics	5
2.2 Keystroke dynamic verification system	5
2.3 Static and dynamic authentication	6
2.4 Calculation	6
2.4.1 Timings	6
2.4.2 Distance metric	7
2.4.3 Template	8
2.5 Pros and Cons	8
2.6 Examples	9
2.6.1 Uppercase and lowercase	9
2.6.2 Double letters	10
2.6.3 Backspace	11
3 State of the art	13
3.0.4 Compare others results	17
3.1 Important findings	18
4 Summary of claimed contributions	21
5 Methods	23
6 Experiment description	25
6.1 Authentication experiment	25
6.1.1 Legal considerations	25
6.1.2 Technical details	25
6.1.3 Participants	27
6.2 Imitation experiment	27
6.2.1 Experiment design	28
6.2.2 Technical details	29
6.2.3 Participants	32
7 Analysis	33
7.1 Distance metrics tried in the authentication experiment	33
7.1.1 Euclidean distance metric	33

7.1.2	Manhattan city block distance metric	33
7.1.3	Euclidean distance metric with standard deviation	34
7.1.4	Manhattan city block distance metric with standard deviation	34
7.1.5	Mahalanobidistance distance metric	34
7.1.6	Cosine correlation coefficient	34
7.1.7	The normalized minimum distance classifier	35
7.1.8	Maximum and minimum distance metric	35
7.1.9	Distance metric with probability of letter	35
7.1.10	Interval metric	36
7.1.11	Pearson correlation coefficient	37
7.1.12	Remove outliers	37
7.1.13	Overlap	41
7.2	Imitation Experiment	42
7.2.1	Threshold	42
7.2.2	Outliers in the imitation experiment	45
7.2.3	Learning curves	46
7.2.4	Descriptive statistics	50
7.2.5	Distribution	51
7.2.6	Regression analysis	52
8	Discussion	59
8.1	Authentication experiment	59
8.1.1	Password	59
8.1.2	High EER	59
8.1.3	Misspellings	60
8.1.4	Timing accuracy	60
8.1.5	Adapting function	61
8.2	Imitation experiment	62
8.2.1	Passwords	62
8.2.2	Distance metric	63
8.2.3	Victim test order	64
9	Further work	65
10	Conclusion	67
	Bibliography	69
A	Table with letter probability	73
B	Letter from NSD	75

List of Figures

1	Keystroke dynamic based biometric verification system.	6
2	Times extracted.	7
3	Durations <i>ene</i>	11
4	Imitation Program one.	30
5	Imitation program two.	31
6	Imitation program three.	32
7	ROC Curve.	39
8	ROC curve.	40
9	The reference profile.	41
10	The reference profile.	42
11	Overlap example.	43
12	Sum of all overlaps.	44
13	Enrolled user templates.	45
14	ROC curves from three participants.	46
15	Learning curves for the three programs.	47
16	Learning curves with five first typings removed.	49
17	Learning curves with eight attackers.	50
18	Histogram for program one.	53
19	Histogram for program one.	54
20	Histogram for program one.	55
21	Residuals plot.	55
22	Adjusted residuals plot.	56
23	FAR/FRR with adapting function.	61
24	FAR/FRR without adapting function.	62
25	Letter from NSD.	76

List of Tables

1	Keystroke times	9
2	Duration and latency timings	10
3	Previous reports	19
4	Experiment design	29
5	Letter probability	36
6	EER for the interval metric	37
7	Outlier removal effects on EER	39
8	Correlation table.	51
9	Min, median, mean and max from program one, two and three.	52
10	Min, median, mean and max when we remove one participant.	52
11	Min, median, mean and max when we remove the first five distance scores.	52
12	Analysis of Variance Table.	53
13	Regression analysis.	57
14	Letter probability	73

1 Introduction

1.1 Topic

In computer security we are interested in methods that securely identify and or authenticate users. Identification is when we try to find the identity of an unknown user, this is done by checking if the unknown user matches any of the known users identities. Authentication is when the unknown user gives us his or her identity, and we check if it is true or not. Username and password is one widely know authentication method. The user claims an identity (the username) and gives his or her password to prove the identity. Authentication is *one to one* verification, we check if this is the claimed user or not. Identification is *one to many* verification, like in forensics when a fingerprint is checked with a fingerprint database.

Users can be authenticated with three authentication factors. The three authentication factors are:

Know Some secret that only the user knows, like password, pin, pass phrase, etc.

Have Something the user possesses, like token, certificate, passport, etc.

Are Some biometric features of the user, like fingerprint, face, iris, retina, palm, voice, signature, keystroke dynamics, etc. With biometrics it is possible to authenticate and identify users based on *who they are*, not *what they possess* or *what they know*.

With biometrics, users are authenticated on something they are. Biometric can be divided in two categories, physiological and behavioural. Physiological means features that are physically related to the user, like fingerprint, face, iris, hand, etc. Physiological features are used in both computer world and in real world. We identify (recognize) people when we see their face. Behaviour means how people do things, like gait, voice, signature, fingerprint, etc. When a close friend calls we can easily identify him or her by their voice. And when an acquaintance calls we authenticate (recognize) him or her after the identity (name) is given.

Keystroke dynamics is in the behavioural category, we can identify and authenticate people by the way they type on their keyboard. During WWII telegraph operators could identify the other operator by the way he or she were typing, see [1]. The telegraph operators used one button and one finger, when we type a PIN code we use ten buttons and up to four fingers, with a password we use about 50 buttons and up to ten fingers. Due to the much higher number of buttons and fingers we can probably distinguish more accurately between various people, but the drawback is that the system will be much more complex.

Although biometrics is popular, they are seldom used in computer systems, the obstacle has often been price, installation problems, and the fact that users are negative against authentication methods that reduce their efficiency/comfort, see [2]. Some users are afraid of using the systems, one example is that users are afraid of eye damage during

iris or retina scans. Systems that capture biometric features often need special equipment, like fingerprint reader, camera, hand scanner, and so on. Keystroke dynamics on the other hand uses the computer keyboard or the numerical keyboard on an ATM machine. There is no need for new equipment, and the keystroke dynamic system can be nearly unnoticeable to the end users.

When users are being authenticated with keystroke dynamics, their typing features are compared, e.g. by finger placement on the button (center, left, top, etc), typing pressure, finger angle, and so on. But these features require a camera or a special keyboard and are probably only interesting for high security environments, future keyboard/computer designs could make this available for cost-sensitive systems. A normal keyboard can register latency between keystrokes, how long a key is pressed (duration) and which key is pressed. We are going to use these standard keyboards.

The features are compared using a distance metric, which is a function that compares two samples and outputs a value for how close or far apart they are. A real life example is when we ask someone *"how far is place A"*, we really ask what the distance between here and place A is. Depending on the person asked we will get different answers, like 150 kilometers, two hours, etc. It is the same with distance metrics, their results will be different although the inputs are the same. Many distance metrics have been tested, and none of them is perfect. Finding a suitable one is a big task in the master's thesis. Another big task is finding which features give the most unique values for people. The goal is to have a low false acceptance rate (FAR), and a low false rejection rate (FRR). FRR is the percentage of legitimate users who are incorrectly denied access. FAR is the percentage of persons who incorrectly gets accepted by the system. We must look at intra-class distance and inter-class distance when we want to find a good distance metric. In keystroke dynamics intra-class distance is the distance between typing sessions for one user, inter-class distance is the distance between different users. Our aim is to find a distance metric that gives a small intra-class distance and a large inter-class distance. The new typing sample is compared with a stored typing template. This template is created during the enrollment phase, when the user is added to the system. A template could be something like the average of ten samples. The template could also adapt to new typing characteristics of the user, for instance if the ten last successful login samples are used as the template.

A high FRR makes the system unusable, since legitimate users have to authenticate themselves several times before they are accepted, if accepted at all. A high FAR, on the other side makes the system insecure. Designers adjust the system to have suitable FAR and FRR values for the specific situation, which is achieved by adjusting the threshold and/or deciding which features is best suited. The result from the distance metric is checked against the chosen threshold, a value below means that the user is accepted, and otherwise denied. A small threshold means that we get a high FRR and a low FAR, a large threshold gives a small FRR and a large FAR. By varying the threshold we can draw a Receiver Operating Characteristic (ROC) curve, which shows the relation between FAR and FRR. We are especially interested in the point where FAR equals FRR, this point is called Equal Error Rate (EER). We want the EER for our authentication system to be as low as possible.

Many reports have tested their distance metrics by checking how easy it is to imitate users. Most of them have done this by using keystroke dynamics for identification, that

is comparing the typed text against all user templates, and if the typed text is similarly to the template of the wrong user, then that is one false acceptance. Others have tried to see how legitimate users type, and then trying to imitate that. A few have tried to videotape legitimate users, and then replicate their typing. None of the test has resulted in a large FAR rate, but we are going to perform a new and possibly better attack on keystroke dynamics. People only learn the basics of snow-boarding by watching others doing it, and will not become good without feedback. We are going to teach attackers to imitate the keystroke dynamics of legitimate users, by giving them feedback on their performance.

1.2 Research problem

Organizations use passwords and sometimes tokens. Users inside the organization have to use token and/or password many times during a day and it is normal to have different password for various systems and applications. This means that people inside the organization have to remember many different passwords. Instead of having one strong passwords they have many weak passwords, which decreases security.

Single sign on solutions with token and a strong password will reduce the problem, but it is not the perfect solution. Users can give their passwords to other users, like when a worker gives his/her token and password to a co-worker, so that the co-worker can fix something. Users often write their passwords on post-it notes in order to remember the long password.

Biometrics will improve the situation, since authentication based on 'have' and 'know' do not really identify the person being authenticated. Instead they only prove that the person possess the token and secret attached to the claimed identity. Biometrics on the other hand authenticate the person against the claimed identity, which in theory means that only the legitimate user can be authenticated as himself.

Studies have shown that it is possible to fool systems that use biometrics, especially physiological systems. Some face recognitions systems have been fooled with a picture, fingerprint systems have been fooled with latex fingerprints glued on the attacker's finger.

Biometric systems based on behavioural features are believed to be more resistant against attackers. But it is possible to fake signatures, like when the bank clerk visually checks the customer's signature against the one on the credit card. Faking a digitally signed signature is much more difficult since features like speed, pressure, angle, etc. can be compared. Studies have shown that it is difficult to imitate other people's keystroke dynamic features. But is this really true, can attackers learn to type as legitimate users? Training and feedback give good results in sports, like in synchronized swimming. But no studies have tried to use training and feedback in order to imitate other people's typing characteristics.

1.3 Motivation and justification

Keystroke dynamics can mitigate the problem with users sharing their passwords and tokens, since they are unable to login with them. Keystroke dynamics can increase the security of passwords if the system checks how the password is typed, instead of only checking if it was the correct one. It is even possible to continuously check if it is the correct user who is typing, making it very difficult to use an unlocked computer by unauthorized persons.

Keystroke dynamics is relatively easy to implement into an existing security solution, since we have all the needed equipment on a normal computer, which is a keyboard. Unlike most biometric systems which need special equipment.

Testing the security of keystroke dynamics, and how easy it is to fraud the system is important when deciding whether or not keystroke dynamics are usable as a security device. If it is easy to fraud the system, then there might be more cons than pros for using keystroke dynamics, and implementations have to protect the keystroke dynamics template like a password. However if it is almost impossible for an attacker to learn someone's typing characteristics, then keystroke dynamics could have a bright future.

1.4 Research questions

The first thing we need to do is to find a good authentication method, which is one that has a low EER. This raises several questions: Which feature to use? Which distance metric is best suited? Which password length is needed? How many enrolment samples? Where to do the test and how many participants?

When we have found a good authentication method, we can find answers to these questions: Can an attacker learn someone's typing characteristics? How easy is it to imitate someone's typing characteristics?

2 Keystroke dynamics

Keystroke dynamics, also known as keystroke analysis or typing rhythms, is a biometric technique that authenticates and or identifies the person typing on the keyboard. It is possible to authenticate and or identify users by comparing their typing characteristics with a stored template.

2.1 Biometrics

In biometrics, users are identified and or authenticated on something they *are*. Other identification and authentication methods are *know* and *have*. Fingerprint is a widely used biometric technique in forensics, voice recognition is used daily during phone calls where we identify the other person by his or her voice.

Biometrics can be divided in two categories, physiological and behavioural. Physiological means features that are physically related to the user, like fingerprint, face, iris, hand, etc. Behavioural relate to how people do things, like gait, voice, signature, fingerprint, etc.

Keystroke dynamics is in the behavioural category, where we can identify and authenticate people by the way they type on their keyboard. In crime fiction we often read that an adversary use a stolen finger or eye to gain access into a high security lab. This is less likely to work in real life due to the liveness test in those physiological systems. But the authors in [3] proved that it is relatively easy to create fake fingerprints, there are also examples of facial and iris biometric systems being fooled with a picture. Physiological features are normally not hidden and can therefore be captured by attackers, some features are however more difficult to get, like retina. Biometrics in the behavioural category are believed to be harder to copy. You could videotape someone's gait or keystroke features, but it would be very difficult to imitate the small details that is unique for the victim.

People's keystroke dynamics are difficult to copy, and if someone manages to imitate a user's keystroke characteristics, the user could change his or her password and the typing features in the new password would probably be very different from the old. The down side with some behavioural biometric systems are that user behaviour changes over time, users typing pattern will change over time and injuries, sickness, alcohol, stress, sleep, etc. can influence users typing, and therefore their probability of being authenticated will decrease.

2.2 Keystroke dynamic verification system

A biometric verification system based on keystroke dynamic is a system that check whether or not the user trying to be authenticated is the correct user or not, see Figure 1. A user that tries to login gives his or her username and password, the typing features are extracted from the typed password and or username. The matcher gets the saved reference template for the specific user, and compares the collected typing features with the stored template in a distance metric. The distance metric score is compared with a threshold,

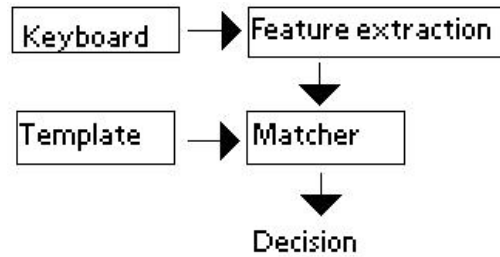


Figure 1: A user type on the keyboard, the typing features are extracted, the typing features are compared with the user's stored template in the matcher. And the matcher decides whether or not it is the correct user.

and the user is accepted or denied access. Keystroke dynamic can easily strengthen existing password based systems, by also checking how the password was typed and not only if it is the correct password.

2.3 Static and dynamic authentication

There is a difference between static and dynamic authentication systems. Static systems authenticate users based on a fixed text, like a password. Dynamic systems continuously checks if the correct user is using the system, so when a user is writing a free text the systems continuously checks if the typing features are correct, which is normally achieved by comparing di- and tri-graphs, see [4]. In this master thesis we are focusing on static authentication systems, in our authentication experiment it is a fixed password for all participants.

2.4 Calculation

2.4.1 Timings

With a normal keyboard it is possible to capture the key-up and key-down time, several timings can be calculated from these captured times. Time between two key-downs, time between key-down and key-up on the same key, time between key-up on one key and key-down on the next key, and time between key-up of one key and key-up of the next key, see [5]. We use two timings in our experiment, duration and latency.

Duration is the time a key is pressed down, this timing is found by subtracting a key's key-down time from the same key's key-up time, see Equation 2.1. Latency is the time between keystrokes, which is found by subtracting the first key's key-up time from the next key's key-down time, see Equation 2.2. In Figure 2 we see that the duration for the first letter *A* is $T_2 - T_1$, and the latency between the letters *A* and *B* is $T_3 - T_2$, where T_1 , T_2 , T_3 and T_4 is the time when a key-up or key-down event occurred. The latency can be negative and the duration is always positive. Time between two key-downs can be found by adding duration for the first key with latency between the keys. And time between key-up of one key and key-up of the next key can be found by adding duration of the second key to latency between the keys. We only used duration and latency in our experiment, since the other two timings are based on duration and latency.

$$\text{Duration} = T_i^u - T_i^d \quad (2.1)$$

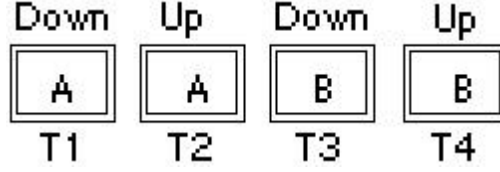


Figure 2: The key-down and key-up times that we can capture, A and B are two keys, T1, T2, T3 and T4 are times when a key-up or key-down event happened.

$$\text{Latency} = T_{i+1}^d - T_i^u \quad (2.2)$$

Where T_i^u is the key-up time, T_i^d is the key-down time. T_i is the first key, and T_{i+1} is the next key. There are other definitions on the latency, some use the key-down to key-down time, but we use the same definition as described in the most recent articles, see [6, 7, 8].

The duration and latency timings for each keystroke in the password are stored in a vector. We decided to store all duration values for the password first, and then all latency values between the letters in the password. The order of the features does not matter, as long as both the template and test vector have the same ordering.

2.4.2 Distance metric

When a user is identified and or authenticated, it is the duration and latency timings that are compared against a stored template. This comparison is done with a distance metric. A distance metric is used as a measure of similarity between the test pattern and the reference pattern, see [9]. In other words a distance metric defines how similar two sequences of values are.

In our experiments we compare the typing characteristics in the freshly typed password against a stored user template for the same password. It is the typing features that are compared, this means duration and latency timings for each keystroke. There are many different distance metrics and variation of these, choosing a good distance metric is difficult. A good distance metric is one that has a large inter person distance, and a small intra person distance, see [10]. Inter person distance is the minimum distance between a sequence of persons, intra person distance is the maximum distance of sequences from the same person, see Equation 2.3 and 2.4.

$$D^{\text{intra}} = \max_i D_i^{\text{intra}} \quad (2.3)$$

$$D^{\text{inter}} = \min_{i,j,i \neq j} D_{i,j}^{\text{inter}} \quad (2.4)$$

Where i and j are the persons.

One of the most used distance metrics is the Euclidean distance metric, we used the Euclidean distance metric in the authentication experiment, see Equation 2.5. The distance metric did serve as a feedback to the participants, so they could see how similar their typing was compared with their stored template, we did not authenticate participants or store the values returned from the participant programs. The Euclidean distance metric was given two vectors, the template and test vector, both containing timings for the same features and with the same number of elements.

$$d(X, Y) = \sqrt{\sum_{i=1}^n (X_i - Y_i)^2} \quad (2.5)$$

Where Y is the stored template and X is the feature vector that we test against the template. In our experiment we used the duration and latency values for all the letters, except for the last letter where only the duration is used.

2.4.3 Template

Keystroke dynamic can identify and or authenticate the person typing on a keyboard, but it needs to compare the person typing against a stored template often called reference profile. A user needs to create a template before he or she can use the system, which can be done by typing a password several times or a longer text. We can then calculate the average typing pattern, and store that as the user template.

In our authentication experiment our participants had to write a password twelve times, before the program calculated their average typing. During analysis we also calculated the standard deviations, and used both average typing and standard deviations in the template.

We improved the template by removing outliers, outliers are points that greatly differs from the other points, see Section 7.1.12. In the authentication experiment we removed outliers that were outside an interval of 1.5 standard deviations above mean and 2.5 standard deviations below mean. During analysis we tested several intervals to see what gave the best results. An asymmetric interval seems to be best, when we have the smallest standard deviations above mean. Especially distance metrics without standard deviations greatly benefits from removing outliers.

2.5 Pros and Cons

Biometric is used in high security systems and on some computers, but it is rarely used in web-shops, online banking, remote access, etc. Price and the need for special equipment are two obstacles that keystroke dynamics can mitigate. Almost all computer systems have a keyboard, and keystroke dynamics can also be used in ATMs, cell phones and other equipment with keyboard or keypad. All that is needed is fairly simple software that can collect the key-up and key-down timings, it is also possible to use plug-in boards, hardware and firmware. By using keystroke dynamics it is possible to mitigate many password problems, like user sharing their passwords, weak passwords, brute force attacks, etc.

Since user's typing pattern change over time and their typing is not as unique as fingerprints, there will always be False Acceptance (FA) and False Rejections (FR). FA is falsely accepting a person as the legitimate user, FR is when a legitimate user is denied access. Systems based on keystroke dynamic perform worse than many other biometric systems, like fingerprint, iris, retina, etc. Different keyboards will also change users typing. A user that changes the computer or keyboard could be denied access especially if the keyboard layout is different (qwerty or dvorak), see [11]. There are also some technical issues that have to be solved before keystroke dynamics can be widespread, like how clients can securely authenticate them self to a server, how to assure that it is a real person typing and not just a playback, how to protect the typing data, and so on.

Table 1: Keystroke times

Action	Time	Key
KEY_DOWN	8,4647942685453	P
KEY_UP	8,58327524866987	P
KEY_DOWN	8,73962500820635	D5
KEY_UP	8,83454489327554	D5
KEY_DOWN	9,18003489270284	U
KEY_UP	9,26723983076061	U
KEY_DOWN	9,49167869100682	X
KEY_UP	9,60711068026802	X
KEY_DOWN	9,65000384127033	RShiftKey
KEY_DOWN	9,93200839771535	A
KEY_UP	10,0494034856385	A
KEY_UP	10,0956303105562	RShiftKey
KEY_DOWN	10,8396111796332	C
KEY_UP	10,9524690987262	C
KEY_DOWN	11,2702187009802	D6
KEY_UP	11,3574376072937	D6
KEY_DOWN	11,6978928886213	E
KEY_UP	11,8107424267609	E
KEY_DOWN	11,8838628931889	M
KEY_UP	11,9403133638493	M
KEY_DOWN	12,0310961309328	M
KEY_UP	12,1106702108788	M

Every keystroke in the password has two times, key-up and key-down. This is the captured keystroke key-up and key-down times for the password *p5uxAc6emm*.

2.6 Examples

In our authentication experiment, presented in Section 6.1, we gave a program to all participants, where they could enter a fixed password whenever they liked. The program automatically sent us the keystroke timings every time the participant had written the password and ended the program. These keystroke timings were automatically put in a MySQL database, and we could collect the data whenever we wanted to. It is the keystroke duration and latency values that were sent to us, the participant programs calculated these two timings for each keystroke. When the program captures the keystrokes in a password, it captures the key-up and key-down times. An example of how these key-up and key-down times looks like is Table 1. These key-up and key-down times are processed in the participant program. Duration is key-down subtracted from key-up from the same key, and latency is key-up from one key subtracted from key-down of the next key. The result from this calculation can be seen in Table 2, we received these results from the participants.

2.6.1 Uppercase and lowercase

All keys in Table 1 are uppercase, which is how the program captures the keyboard events. It knows which button is pressed and if it is a key-down or key-up event, but it does not know if it is a small letter or a capital letter. With a standard keyboard there is of course only one button for each letter, not one uppercase and one lower case. All

Table 2: Duration and latency timings

Key	Duration	Latency	Next key
P	0,11848098012457	0,15634975953648	5
5	0,0949198850691904	0,345489999427301	U
U	0,08720493805777	0,22443886024621	X
X	0,1154319892612	0,0428931610023096	ShiftKey
ShiftKey	0,44562646928587	-0,163621912840849	A
A	0,11739508792315	0,7902076939947	C
C	0,112857919092999	0,317749602254001	6
6	0,0872189063135007	0,3404552813276	E
E	0,112849538139599	0,0731204664279996	M
M	0,0564504706604012	0,0907827670834997	M
M	0,0795740799459992		

Every keystroke in the password has two timings, duration and latency. Except for the last keystroke that only has duration, this is because the latency is from the specific keystroke to the next keystroke, and the last letter does not have any subsequent keystroke.

keystrokes are therefore denoted with uppercase letters, since we press the same button for both upper and lowercase letters.

In the password “*p5uxAc6emm*”, there is one uppercase letter, the “A”. And we see in Table 1 that the shift button is pressed before the “A” button, and released after the “A” button is released. This means that the program must check which letters are pressed while the shift button is pressed.

Uppercase letters therefore need two keystrokes, the shift key and the letter. Passwords with uppercase letters will have more keystrokes than passwords with only lowercase letters. This is a good thing for keystroke dynamics since we get more features to check, see [12].

The shift key can also give more information about the user, and the authors in [11] discovered that there are four different shift key users. Strictly left-shift key users, strictly right-shift key users, opposite shift key users, and users without a shift key pattern. The strictly right- or left-shift users are easy to check, the opposite shift key users are using the same shift key for a specific letter and also easy to check. The users without a shift key pattern were not truly erratic, and the majority of the letters were connected with a specific shift key. And only four of the fifteen participant in their experiment were inconsistent with the shift key. The problem with the shift key is that it is easy to discover which shift key the user is using through shoulder sniffing. And passwords are often short texts with few uppercase characters, making an opposite shift key user very difficult to discover. But checking if the correct shift key was used is an easy check and could improve the keystroke dynamic verification system.

2.6.2 Double letters

In Table 2 we see that the two *M* keystrokes durations are different, this is strange since it is the same button and the duration should be fairly consistent. But it seems like both duration and latency depend on the surrounding keystrokes. Latency will of course depend on the next letter, since it is the time between keystrokes and it depends on placement, etc. between the keys. Duration on the other hand should be relatively similar for the

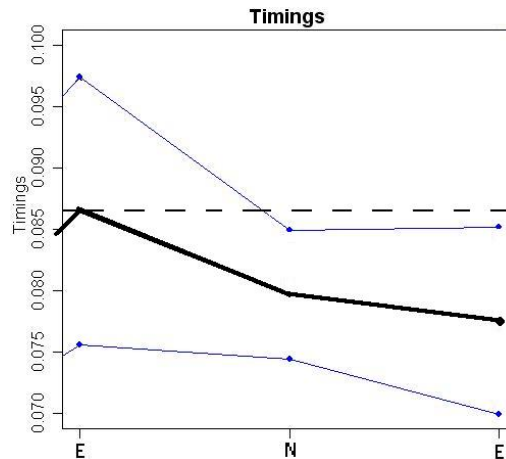


Figure 3: The durations for the keystrokes “ene”, in the password “p7eneZuh”. The blue/thin curves are one standard deviation above and below mean, the black/thick curve is the mean. This is the enrolled template for an experienced typist who typed the password thirty times. The horizontal dashed line has the same height as the first e’s mean duration.

same letter, since it is the same finger doing the same movement as done thousands of times before, if the person typing is an experienced typist. Inexperienced typists will probably have larger variations in their typing, than experienced typists. But with the password “p5eneZuh” we discovered that the two e’s had different durations, see Figure 3. In the figure the blue/thin curves are one standard deviation above and below mean, the black/thick curve is the mean. And we have drawn a horizontal dashed line with the same height as the first e’s mean value. These mean values and standard deviations are from one experienced typist who typed the password thirty times. The first e has a larger standard deviation than the second e, but the mean duration of the first e is higher than the second e’s duration added with one standard deviation from the second e.

2.6.3 Backspace

Users do make spelling mistakes when they enter their password. Sometimes it is because they hit two letters, but could also be that they remembered their password wrong. When a user discovers that something went wrong, before pushing the login button, users normally use the backspace to erase the mistake, and try again. This would have been accepted in a normal username/password system, but it is different with keystroke dynamics.

If the user erases the entire password and start from the beginning, the password will be very similar to a normal password typing, and should therefore be accepted. This can be achieved by removing all keystrokes before the password, which is the previous misspelled password, and some backspaces. And then compare the new password against the template. If the user erases the misspelled letter and continues writing the remaining part of the password, the problem is more difficult to solve. The latency timing between the “last good” letter and the corrected key is lost, the duration of the corrected key can also be different, as we discovered with the double letters. This means that there will be fewer features to compare in the distance metric, and the decision will probably be less accurate. It is of course possible to adjust the distance metric to accept longer or shorter

passwords. This could be done by using the average distance instead of the normal, which is the sum of all distances. It is also possible to use the backspace in the password, if users also had enrolled the backspace. This should probably be done in advanced keystroke dynamic systems, since the backspace most likely contains discriminating information about the user. The authors in [13] wrote that users could deliberately use the backspace in their password, the ordinary password and the users keystroke template would then be of unequal length and hard to attack.

We did not use the backspace in our experiments, if the user typed a backspace, our system would not calculate the distance score, instead the systems returned *wrong password* to the user. A usable keystroke dynamic system for large organisations should handle backspaces.

3 State of the art

There has been written many reports on keystroke dynamics, but it is hard to find literature on working systems and programs. BioPassword is one commercial system, but they have not released any information about their system, although it seems to be good, see [14]. Another commercial system with little information is the TypeSense from Deepnet Security, see [15].

Some reports have tested the possibility to continuously verify the user during the session. The user can work as normal, but he or she is constantly verified. The authors of [4] had an experiment with 205 individuals, where their system issued less than 5% false alarms, and impostors were caught in 99,995% of the situations. Giving a FAR rate of 0,005% and a FRR of 5%.

The authors of [16] tried keystroke dynamics for a PIN code, where they included the typing force through the use of a special number pad. They also tried to videotape users typing their PIN, and having a group of impostors trying to authenticate as the videotaped user. Their group of impostors was not able to imitate the videotaped users, and force was the hardest to get right. But this was for a small number of buttons, and only a four digit PIN. In [17] the authors included typing force with a special keyboard, where they used a 6 character long password without any special characters or numbers. Their participants had to give 20 enrolment samples, and only a few authentication samples. This will of course give fictitious results which we cannot trust, since very few authentication samples compared with the high number of enrolment samples and all samples were gathered in one sitting. But we can conclude that typing force is distinctive, and typing force combined with timing data would probably give better results than only timing data.

In [18] keystroke dynamics was used to not only check that a password was correct, they also checked how it was typed. They managed to get a False Rejection Rate (FRR) of 11.57% and a False Acceptance Rate (FAR) of 1.9%, but they claimed a FRR of 1.45% when they allowed user to try twice before they where rejected. This can be called cheating, but it shows that systems with a high FRR rate can be useable since 98% of the users will be authenticated after two tries. They used an adoption mechanism that updated the template, by discarding the oldest sample and adding the new sample that was accepted. The standard deviation and mean for each feature is recalculated based on the updated template. The classifier was based on statistical standard deviation and the threshold was a fixed number of standard deviations from the mean. Another important discovery was that a higher timing accuracy decreases FAR and FRR. They also tried to imitate legitimate users, by letting the impostor see how the victim was typing, and later tried to mimic the legitimate users. They managed to get a FAR of 3.66%, which is worse, but still acceptable FAR. They concluded that keystroke dynamics are secure against shoulder sniffing, and that a familiar text (password) gives better results than a random text.

When keystroke dynamics are used to complement passwords in a login situation, there are some problems that arise. The users could find it cumbersome, if they have to

give username and password, and then a short sentence for the keystroke dynamics. A better solution could be to check on both username and password. The authors of [19] claims that this convenient solution could reveal information about the password. If we exclude the password and only check the username, there will be a high rate of misclassifications if the username is shorter than 10 letters [12, 20], but a good distance metric will reduce the problem. There are several solutions like longer username, check username and a very short sentence, or check username and password. The last solution requires encryption, but that is probably the case for all solutions. The problem with revealing information about the password is more important when using online verification systems through SSH or similar systems. Since the attacker could eavesdrop on the communication, and reveal the timings if the information is sent in *real time*. The authors in [21] have shown that if the latency times between keystrokes are revealed then it is much easier for an attacker to find the password. Instead of having to test 50% (the birthday paradox) of the password space in a brute force attack, only 1 or 2% have to be tested before the password is found. The authors of [22] have shown that keystroke dynamics can make passwords more secure against brute force attacks. Even weak passwords would require much work by the attacker, but this depends on the implementation.

Different distance metrics have been tested. In [20] the authors tested two distance metrics. The normalized minimum distance classifier (3.1) and the normalized Bayesian classifier (3.2):

$$D_i(X) = \frac{(X - m_i)^t (X - m_i)}{\|X\| \|m_i\|} < T_1 \quad (3.1)$$

$$d_i(X) = \frac{(X - m_i)^t C_i^{-1} (X - m_i)}{\|X\| \|m_i\|} < T_2 \quad (3.2)$$

Where X is the test vector, m_i is the mean vector, C_i is the covariance matrix, and T_1 and T_2 are threshold values.

Both statistical and neural classification method were tested in [23], the neural methods *fuzzy ARTMAP*, *RBFN*, and *LVQ* gave the best results. The statistical methods *Bayes rule* and *potential function* gave both good results, but they were beaten by most neural techniques. The problem with this report is that the authors did not publish the distance metrics used, or the dataset.

The authors of [24] gathered typing samples with a java applet. They had 143 patterns of legitimate users, 251 legitimate logins, and 170.391 invalid user login attempts. Very few legitimate logins compared with the high number of invalid login attempts (attacks). They tested a scoring method based on standard deviation. They compared digraph latencies in the password, every digraph inside a threshold of n standard deviations from the mean were given the value 1.5. Latencies outside of this threshold were given the value 1. These values were added together and a user was accepted if the sum was higher than a decision threshold, like 70% of the maximum score achievable. They managed to get an EER of 5.85%. It has been proved that letters scattered across the keyboard provide more accuracy to keystroke dynamics than letters that are close together. They used this knowledge and gave latencies between some letters a higher value, this gave an EER of 5%.

The authors of [25] had an experiment with 63 users, where the users ran the program from their own machines at their convenience. The typing samples were collected

over a period of 11 months. They extracted duration and latency of digraphs in a typed text, and tried to identify users with four different classifiers. Euclidian distance gave a correct identification rate of 83.22%, non-weighted gave 85.63%, weighted probabilistic gave 87.18%, and Bayesian gave 92.14%. They did not publish their tested text length, but it must have been significantly longer than a regular password. They also discovered that fixed text gave better results than free text.

The authors of [13] have compared the published results from fifteen scientific reports. Many of the authors have reported FAR and FRR better than 2.5%. But many of the good performers require users to write long texts before they are authenticated. This makes many of the systems unusable in real authentication situations, it is unacceptable for users to write long text several times a day. Keystroke dynamics would be a very costly solution for a company, if their workers had to spend several minutes a day only to be authenticated. Some of the solutions require the authentication system to be retrained every time a new user is added, or when users typing behaviour change over time. Systems that have to be retrained are almost unusable for large organizations. Only two of the fifteen reports had more than 50 participants, and the majority had less than 25 participants.

The authors of [8] had thirty-three participants in their experiment, and their users had to write their username, password, first name and last name. The participants were then authenticated based on these four texts, through the use of a distance metric similar to Manhattan City Block distance. Their decision threshold was based on a measure of variability of the specific participant, like mean plus one standard deviation. The participants wrote eight login samples during enrolment, and then the participants tried to login five times. All samples were collected during a single session, making the experiment less realistic. Their participants had different passwords and usernames, which meant that the FAR could not be easily calculated. They chose six users that the remaining 27 users should try to login as, and the attackers got five tries against each of the six user accounts. The attackers did not witness the targets typing. They got good results, 0.25% FAR and 16.36% FRR. But we know that [26] tried the same experiment and they got 7.72% FAR and 37% FRR, which is much worse. Although [26] used a shorter text, the results achieved by [8] are undeservedly high. The authors of [8] did also propose an idea where the signature shapes are compared, and not only the absolute distances.

The authors of [27] tested Artificial Neural Networks (ANN) and K-Nearest Neighbour Algorithm. They had ten participants with ten different passwords, and one hundred unauthorized participants. The unauthorized participants tried to write legitimate users' passwords a limited number of times, but they had no other knowledge than the password. Using different passwords for each user will give better results, especially FAR. Different passwords and only ten users make their results less trustworthy. K-Nearest Neighbour Algorithm gave 15.4% FRR and 1.03% FAR, ANN gave 1% FRR and 29% FAR. A very high FAR with ANN, which could be reduced with further training of the systems. This training phase is the big drawback with ANN's.

A parallel decision tree (DT) based method was tested by the authors in [26], but first they tested the experiment in [8] with a shorter text, and got much worse results. The decision tree-based method requires the system to be trained, just like ANN's. The DT experiment was conducted with 43 participants, typing a text string of length 37. Nine typing samples from each participant were gathered during the enrolment phase.

Additional training data was simulated based on these nine typing samples and both simulated and real samples was used during the training phase. The text string contained no capital letters, and if the participants used DELETE or BACKSPACE they had to retype the entire text string. They managed to get 9.62% FRR and 0.88% FAR with the long text, and with an eight character text they got 13.97% FRR and 9.19% FAR. They claim that a string size of 30 characters is required in order to get acceptable results.

The authors in [28] did one experiment where the participants downloaded the keystroke program, and used it on their own computers. Typing samples were gathered for seven weeks, and the program automatically e-mailed typing samples to the authors. The participants did not have to remember the text they should type, it was always displayed on the top half of the program. They initially had 42 participants, but they reduced the number of participants to 31 due to erroneous timing results. The reference profile for each participant was computed by calculating mean and standard deviation of both latency and duration. The reference profile was further refined by removing values that were T standard deviations from mean. This was done until no more *outliers* could be removed, recalculating the mean and standard deviation each time. This technique produced better results than the shuffling technique in [12, 20]. They discovered that frequent combinations like *th* are more distinctive than less frequent combinations like *tl*, and frequent combinations should be given more emphasis.

The authors in [29] performed one experiment where the participants typed a 52 character long sentence plus their name. They had eleven participants who typed the sentence approximate 15 times, most participants wrote all sentences in one sitting. The participants had to write the sentence all over again if they did one mistake, they could not use delete or backspace. They used duration and $D_1 \rightarrow U_2$ latency, which is the timings between the first letter in a digraph is pressed and the second letter is released. Users in this experiment were accepted if over 80% of the latencies (D_1 to U_2) was within one standard deviation from the mean. They varied the percentage and number of standard deviations, resulting in an EER of approximate 20%.

The authors in [30] tried to authenticate mobile users with keystroke dynamics. Modern mobile phones are capable of online banking, and they are therefore a valuable target for criminals. The authors proposed both static and dynamic authentication, static is here when the users type their pin, and the dynamic is when the users type a text message or telephone number. In their experiment the pin and eleven digit telephone number was fixed. They had 32 participants, all typing the fixed pin and telephone number 30 times in one sitting. They used the 20 first typings to generate the reference profile, and the last ten typings where used as authentication typings. Their text experiment had thirty participants typing thirty messages with an average of 14 words in each message. They achieved good results with artificial neural network techniques, PIN gave 9.4% EER, telephone number gave 8.1% EER, and the text messages gave about 20% EER. They used six letters in the text verification experiment. Their participants had very varying results, one had an EER of 7.2%, and another had an EER of 42.6%. They believed that the users with high EER do not naturally have discernible typing characteristics. They suggest an adoption mechanism where accepted typing samples are moved into the user's profile, so that the user profile size is constantly growing.

The authors in [6] carried out an experiment where participants were authenticated based on how they type their name. Their name had a maximum length of 40 characters,

which is a much longer text than our experiment. They got good results with a distance metric based on the Manhattan city block distance:

$$D(X, Y) = \frac{1}{n} \sum_{i=1}^n \frac{|X_i - Y_i|}{\sigma_i} \quad (3.3)$$

They got the best results with both duration and latency, and by using only 90% of the features. They removed the features with high standard deviation. They got an EER of 4% with different thresholds for each participant, and with a global threshold they got an EER of 8%. A global threshold is a threshold which is the same for all users, this is what we will use for our experiments. But this means that we will get worse results than we could with individual thresholds.

The authors in [31] wrote about secure client authentication on the web. They claim that users often have the same password for various sites, which means that if an adversary manage to get a password for one user, the adversary would probably be able to gain access on other systems with the same password.

3.0.4 Compare others results

In [18] the authors reduced their FRR rate by only counting the situations where users failed to authenticate both the first and second trial. A correct FRR rate is 11.57%, but they claimed 1.45%, see Table 3 for a short list of other reports and their results.

In [12] the authors used the text “UNIVERSITY OF MISSOURI COLUMBIA” as a password. They only had ten participants, and they did not count when a user typed the text wrong. They only published percentage of misclassifications, since they used their system for identification instead of authentication. Although their results are good (1.2%), it is important to remember that they only had ten participants and their checked text was very long compared to the number of participants. They got the best results when their text was typed twice. And the two captured samples were combined into one sample, where the smallest latency was chose, this reduced noise from hesitations, etc. With a ten characters text they would get more than 15% of misclassifications.

The authors in [20] did three tests, in the first they tried to identify ten people who all typed the same text, only 1.2% were unidentified. The first test was the same as the one in [12], probably the exact same test. The other two experiments were a verification experiment with 26 participants, and an overall recognition experiment. The overall recognition experiment used data from experiment one and had 22 participants that tried to be verified as one of the ten users from experiment one. The last two experiments showed that users who are unfamiliar with computer keyboards will have a higher FRR, and it is also easier to imitate these users.

In [4] the authors did a large experiment on dynamic authentication. They tested if keystroke dynamics can test users during their session, after authentication. Their average text length was 800 characters of free text, where the users was free to type whatever they wanted and decisions were take on the available information. They used digraphs, trigraphs and four-graphs to compare samples and template. A digraph is two often used letters like *th* or *he*, a trigraph are three letters combinations like *the*, four-graphs are combinations like *wait*. They used *JavaScript* to capture data that were typed in a *HTML* form. Based on their good results we can say that keystroke dynamics can be used in dynamic authentication, at least in some web-based applications.

In [32] the authors had three distance metrics that they used a sum rule on to get

the best results. The three distance metrics were statistical (median and standard deviation), disorder between two vectors, and time classification.

The authors of [33] used neural networks as a distance metric. They got problems with negative latency, so they had to use the key down to key down latency DD . They had 91 valid users, and 61 impostors. At first their FAR rate was 6.56%, and their FRR rate was 2.22%. Then they reduced the root mean square error (RMSE) from 0.07 to 0.03, through further training of the system, this means that their template is improved. The new FRR was 1.11%, and the new FAR was 0%. They had too few impostors to get good results, and they did not use statistics to create an ROC curve.

There are several guidelines for creating secure passwords, passwords should consist of upper and lower case letters, numbers. Passwords should also be longer than eight characters and they should be entirely random. In [34] the authors discovered that random passwords were hard to remember, but pass phrases were easy to remember. They also discovered that pass phrases was as secure as random generated passwords when it comes to brute-force attacks and dictionary attacks. The recommended length from their study was nine characters or longer.

3.1 Important findings

Text length The authors of [12] discovered that texts shorter than ten characters give many misclassifications. The authors of [18] discovered that texts longer than ten characters annoy the users.

Capital letters The shift button alone detected 70% of the impostors in [18], so capital letter will improve the FAR.

Keyboard partition Passwords with letters from several partitions of the keyboard make decisions more accurate. [24]

Enrolment The authors of [18] discovered that the number of samples gathered during enrolment should be about 6. But it should be possible to have fewer samples if an adoption mechanism is used.

Adoption mechanism The authors in [18] discovered that the typing template should be adjusted after each successful login. Since users typing pattern changes over time.

Time The users typing samples should be collected in different time period, never all at once. This will make the experiment more realistic, see [18].

Timing accuracy The results can be improved with better timing accuracy, and it should be 1ms or smaller, see [18].

Users Users that are unfamiliar with computer keyboards will have a higher False Rejection Rate, and it is easier to imitate their writing. [20]

Brute force attacks A password system with keystroke dynamics will make brute force attacks obsolete. The rejection of a password does not necessarily mean that the password was incorrect. Brute force attacks will also be too time consuming, since each password takes a few seconds to write. See [35].

Table 3: Previous reports

[REF]	[20]	[12]	[33]	[18]	[4]	[32]
Year	1990	1991	1997	2005	2005	2005
ST/DY	ST	ST	ST	ST	DY	ST
Distance metric	Bayes and minimum	Fishers / minimum distance	Neural network	Statistical σ	Disorder	Sum of three
Features	UD	UD	DD, DU	DD, UD, DU	DD	UD, DU
Text length	11-17	31 (62)	Password (7-)	10	Free	25
Participants	(10) 26, 32	10	90(61)	30	205	15
FRR	8.1, 3.1 %		1.11 %	11.57%	5 %	6 %
FAR	2.8, 0.5 %		0 %	1.89 %	0.005 %	0.5 %
Real FAR	2.8 %		Higher	3.66 %		
Attack	Shoulder sniffing	NONE	Identification	Shoulder sniffing	Identification	None
Description	(1.2%)	(1.2%)	61 impos-tors	Claimed FRR: 1.45%	Text average: 800 char	

This table compare the results of other reports. *Year* is when the report was written, *ST/DY* is static or dynamic authentication, *distance metric* is which distance metric technique used, *features* are which features they checked, *text length* is how long the checked text length was, *participants* is the number of participants, *FRR* is the False Rejection Rate, *FAR* is the False Acceptance Rate, *real FAR* is the FAR rate they got by applying their attack, and *attack* is how they attacked if they did any attack. The features *DD* is keystroke latency between two successive key downs, *UD* is the real latency taken from the first key is released until the next key is pressed, *DU* is the duration a key is pressed. If we remove the first keys *DU* time from the *DD* time between two keystrokes, we get *UD* time.

4 Summary of claimed contributions

This master's thesis is divided into two parts, where the first part is finding a good authentication method for keystroke dynamics, while the second part is to evaluate keystroke dynamics against attacks. In the studied literature, some attacks on keystroke dynamics authentication systems are described, but the most advanced attacks are shoulder-sniffing techniques. We believe that more advanced attacks can happen in the real world. We therefore want to evaluate the security of keystroke dynamics against advanced attacks, by developing a program that helps attackers learn others typing characteristics. If the use of this training program results in a significantly lower distance score than the use of a normal authentication system, keystroke dynamics could be unusable as a secure method of authentication.

The first part of this thesis is there to find a good distance metric which we need in part two. There are written many reports on distance metrics, our goal will be to find an adequate distance metric from the reported distance metrics. However, our main focus is the second part, where we investigate whether or not it is possible for an attacker to learn someone's typing characteristics.

5 Methods

In the imitation experiment we wanted to answer the question “*can an attacker learn someone's typing characteristics?*”, see Section 6.2. We needed a suitable distance metric to answer this question, the authentication experiment was designed to find this distance metric, see 6.1.

To answer the question we needed to measure the attackers learning. Since peoples typing characteristics are relatively similar, the distance score between people will decrease for each typing, until a certain point is reached. This means that all attackers will decrease their distance score against a victim, independent of the used program. We believed that it was possible to measure the attackers learning, by giving the attackers various feedbacks on their typing against the victims typing. If the attackers improved their imitation with more feedback, it means that attackers can learn someone's typing characteristics.

We created three programs with various feedback levels, full feedback, a score feedback and almost no feedback. The program with the least feedback was similar to a real authentication system, where the attackers get an accepted or not accepted feedback.

Since we knew that we would have a limited number of participants in the imitation experiment, we needed an experimental design that did not require many participants. We therefore decided to go for a time-series design. The program with the least feedback is the baseline. Since we were afraid that the program order could influence the distance scores, we could not set the baseline before the treatments, where the two other programs were the treatments. It is of no use comparing another password, since peoples typing characteristics in static keystroke dynamics are strongly connected with the typed text. We therefore decided to use a design similar to the alternating treatment design, see [36]. This means that we try different treatments in a time-serie. Instead of alternate between two treatments, we have three groups that tested the three programs in different order.

One group could test program three, then program two and finally program one. Another group tested program one, then program three and finally program two. It was also important that the participant started with a new password for each victim they were trying to imitate, since we needed the learning process to be the same for each new victim. With three different passwords, the attackers would get a new password for each victim they were trying to imitate. We were concerned that the password could be important for the attackers ability to imitate, and that some password/user combinations could be easier to imitate. We therefore needed to organize the order of the password/victim/program combinations to be different for each group.

We managed to create an experiment setup with no correlation between program, password and victim. This was very important to satisfy one of the regression analysis conditions, namely that all independent variables should be uncorrelated.

To test if it is possible to learn someone's typing characteristics we created learning curves for all three programs, and performed statistical analysis like regression analysis, f-statistics, and descriptive statistics.

We create two hypotheses in the experiment.

H_0 It is not possible to learn someone's typing characteristics.

H_1 It is possible to learn someone's typing characteristics.

If we keep H_0 , a user's distance score from program three is not significantly lower than program one. We choose to have a 95% confidence interval in the hypothesis testing.

6 Experiment description

In this master's thesis we are interested in answering the question "can an attacker learn someone's typing features?". To answer this we first need to have a working verification system, which is capable of deciding if it is the claimed user or not that is typing the password. We carried out two experiments in the master's thesis. The first one was to find a good distance metric, and a good verification system. The second experiment was to see if attackers could learn someone's typing features

6.1 Authentication experiment

With this experiment we wanted to find a suitable distance metric for the imitation experiment, but we also wanted to compare some of the techniques used in literature.

We wanted the participants to type the password as naturally as possible, like when they type their computer password, their *MSN* password, and other login passwords. It was believed that this would be achieved if the participants downloaded a program and ran the experiment on their own computer, so the environment was as natural as possible, using their own computer, keyboard, chair and office or house. A similar experiment setup was done in [28, 25], where a program was given to all the participants. And similar to those experiments, the keystroke timings were sent to us.

6.1.1 Legal considerations

Keystroke dynamic is a biometric technique and handling of biometric data needs to be reported and applied for pursuant to the Norwegian law. We applied to the Norwegian Social Science Data Services (NSD) [37] and they concluded that the authentication experiment was obligated to report according to § 31 in Personopplysningsloven [38]. NSD also concluded that our security measures regarding protection of the participants identity and their typing characteristics were sufficient good, according to the law. It took five weeks before the experiment was legally approved by NSD, see Appendix B.

6.1.2 Technical details

The program was programmed with C#, and designed to replicate a normal login procedure, where the participants write their password in a password field. The password characters are displayed as * instead of characters as in [28]. We could have created a Java applet instead of the C# program, but the Java applet has various timing accuracies between different systems. Java applets running on Microsoft systems can have 10 ms or worse timing accuracy, and we know from the authors in [29] that the timing accuracy should be better than 1 ms.

Before the participants could try to login they had to enrol into the system. They enrolled into the system by writing their password several times, from which the reference profile was created. The participants would be validated against their own reference profile when they tried to login during the authentication phase. The authors of [8] claim that six typing samples would be enough to create a good reference profile. However, we discovered that six samples were too few, especially with longer random passwords. With

shorter passwords, like “*elmafbr*” six samples were enough, if the participants tried the password several times before they enrolled into the system. But we would be cheating if we claimed that our system performs well with few enrolment samples, when the participants had familiarized themselves with the password before enrolling. It is then better to use a higher number of enrolment samples.

The reference profile is the average duration and latency timings for each letter in the password. But we improved the reference profile by removing outliers, which are timings that are very different from the other timings. These outliers can come from hesitations, and are very common during the enrolment phase when users are learning their new password. There are several techniques for improving the reference profile, a shuffling technique is proposed by the authors in [12, 20]. This shuffling technique combines two entries, by choosing the lowest value for each feature, like the lowest duration for a specific letter in both entries. The authors used five shuffled entries to generate the mean vector, which means that their participants had to enter the text ten times. The authors in [8, 28] used a technique based on statistics, and this technique proved more accurate than the shuffling technique. By removing values further away from the average than X standard deviations and doing this several times, recalculating the average and standard deviation each time, until all remaining values are within the accepted range. Another solution is to use the median, which is the numerical centre of a sorted set of data, with the same number of values above and below median, see [36]. The technique in [8, 28] is easy to implement and effective, we therefore decided to use this technique. But since hesitations normally results in long timings, and these long timings will make the normal distribution positively skewed, we adjusted the technique so that we allowed fewer standard deviations above mean than below. This reduce the risk of removing the best typing sessions. The authors in [8, 28] did only remove the outliers above mean, but we believe that it is possible to also get outliers below mean, like when users weakly tap a button.

When users are authenticated with keystroke dynamics their typing sample is compared against their reference profile. This comparison is achieved with a distance metric, but first the typing sample must be converted into a vector that is compared against the reference vector. We decided to use the Euclidean distance metric for comparing these two vectors, as in [8]. Although the authors in [26] got bad results with the same approach, when they used a *short* password. We are not going to deny or accept users based on the output from the Euclidean distance metric, we are only presenting the distance score to make the program more interesting for the participants.

We decided to give the participants information about the experiment goals and how the program works. We could have had one group of participants unaware of this information, but this would increase the risk that participants let other people try to login on their system. If other people try the participants system, this would certainly reduce the accuracy of our findings. Another experiment proved that there is no significant correlation between knowledge of the verifier or not, see [8].

The program will only send three features, key code, duration and latency. The key code is needed so we can verify if it is the correct password or not. We can calculate other timings from these two timings, for instance *DDL* is duration of first letter plus latency. If the participants has to use backspace or delete, our program will not calculate their distance score, but we will use the data so we can calculate statistics like how often

participants enter the wrong password or make spelling mistakes. Incorrect typed samples are normally discarded in others experiment, like [12, 20]. However incorrect typed samples are not allowed during enrolment, since these will create inaccurate reference profiles.

We did not create an adaptation mechanism in the participants program, and this means that their score from the distance metric will get worse during the experiment. This will not give us any problems, since we do not use the score generated by the participants program, and are only interested in the participants typing characteristics. When we evaluate the received data we will test adaptation mechanisms.

This experiment will have fewer characters in the tested text than most other experiments, and more participants than most other experiments. This will make our results competitive, but most likely worse than other experiments, due to the short password and number of participants, see [13].

6.1.3 Participants

Most of the participants in this experiment are very good typist, only a couple of them are inexperienced typists. The good typists are using most of their fingers when they type, and they can write without looking at the keyboard, while the inexperienced typists has to spend more time searching for letters and they are using few fingers. The majority are students and teachers here at *HIG*, and the rests are in various job positions. There where 21 participants and since they were free to type whenever they liked, some gave less than 20 samples, and other gave more than 100 samples. One participant gave more than 300 typing samples, with an average of 95 typing samples for each participant. We gathered 2000 samples in total, and about 16.9% of the samples were misspellings. This means that about every sixth password sample is typed wrong. This fail rate is probably lower for shorter passwords, but it should be taken into account when designers create a keystroke dynamics authentication system.

Both men and women participated, but the majority was men. The participants were aged between 18 and 55, with the majority between 20 and 30 years old.

6.2 Imitation experiment

The imitation experiment is designed to answer the question “can an attacker learn someone's typing features?”. In this experiment we check if the level of feedback to the attacker can help him or her imitate legitimate users. By giving attackers three levels of feedback.

Yes/no feedback. Similar to a real system where an attacker is being accepted or not.

Score feedback. A system where an attacker gets more feedback than needed, the distance score.

Full feedback. The attacker gets hold of the template, and can see which features are wrong.

If attackers can improve their distance score as a genuine user with the full feedback, then we can conclude that it is possible to learn someone's typing features. However if there are a small or no improvement then we can conclude that it is very difficult to imitate someone's typing features.

The attack groups are not going to see how the victims type their password, which makes the attacker's task a bit more difficult. But we want to create an attack scenario where attackers get their hands on a user's authentication data, which is the username, password, and typing reference template. If the attackers are capable of imitating the users typing without even watching the victim typing, we can safely conclude that it is possible to imitate someone's typing features.

6.2.1 Experiment design

We want to check if the feedback given to the attackers enables them to imitate users. We therefore have them test three different programs, which return different amounts of information to the attackers. By varying the amount of information, we are able to check whether or not attackers are able to learn to imitate users. We also need to check different passwords, so that we are not using a password that is very easy to forge.

Three programs:

1. Accepted/Fail feedback.
2. A score feedback.
3. Further information, a score feedback and graphs that visualize the differences.

The three programs use the same distance metric, which is the distance metric tested in Section 7.4. We decided to use this distance metric since it performs well, and it is one of the easiest distance metrics to implement. The first program is similar to a real system, where users are either accepted or denied access. The second program will print the distance metric score to the attacker, and the attacker can use this score to improve his or her imitation of the victim. An attacker is unable to know where he or she is wrong, the third program will give the attacker this information. The third program returns the same score as in program two, but it also outputs a graph that visualize the victims typing and his or her own typing. With this graph an attacker can see where his or her typing differentiates from the victims typing. The graph displays both duration and latency, since the distance metrics takes both features into account.

Three victims typed three different passwords thirty times each, and these thirty typings were used to create a template for each password, resulting in nine different templates. This means that we will have one unique template for each password program combination. We have nine password program combinations, and each of the three programs will be used in combination with the three different passwords. This means that each password will be used three times, with three different templates. Such that a password template combination is never used twice, see Table 4 for the experiment design.

We wanted a shorter password for the imitation experiment than we had for the authentication experiment, since a ten character long password takes long time to learn than an eight character long password. We used a random password generator [39] to create the three passwords:

1. jamu9reS
2. bruf9Tr2
3. p7eneZuh

Our three users typed these password thirty times. We checked that the three users

Table 4: Experiment design

Attack group: Victim	A	B	C
I	P1 Prog3	P2 Prog2	P3 Prog1
II	P3 Prog2	P1 Prog1	P2 Prog3
III	P2 Prog1	P3 Prog3	P1 Prog2

P1, P2, and P3 are the three different passwords. Prog1, Prog2 and Prog3 are the three different programs. In the table we see that attack group A will first try to imitate victim I's password one with program three. Then Victim II's password three with program two. And finally victim III's password two with program one.

had different typing characteristics by comparing their templates. If we did not check this we could risk having users with very similar typing characteristics. This means that the attackers could learn the general typing characteristics of the users, and they only need to do small modifications to imitate the other users. We also made sure that the attack groups have different victims for each password and program, and different password for each program. We also designed the experiment so that the groups use the programs in different order.

These precautions should make us able to see if the extra information really helps attackers.

6.2.2 Technical details

We want to answer the question “can an attacker learn someone's typing features?”. The imitation experiment is designed to answer this. In the authentication experiment we discovered that there were similarities between the users typing features, and also sufficient differences to authenticate users. There are also similarities between the three participants acting as victims in the imitation experiment. Figure 13 describe the enrolled values for the three participants acting as victim in the imitation experiment. There are three graphs from each participant, the mean values, the mean + one standard deviation, and mean - one standard deviation. We see that the enrolled average values are closer than one standard deviation for some of the typing features. Since users share some typing features the attackers will automatically get a lower score when they have typed the password a few times. There are hesitations and other irregularities in the typing the first few times, from the authentication experiment we know that users get a fairly consistent typing after only a few typings. Therefore relatively few password typings are needed to see if there is a learning curve or not. It is also the fact that users will get bored, and less willing to concentrate on the imitation if we have many password typings.

We wanted to see if attackers can learn someone's typing features. When someone is doing the same task several times, the time needed to perform the task will decrease. People are *learning by doing*, and this can be expressed mathematically with learning curves. Learning curves are also known as *progress curves*, *experience curves*, or *improvement curves*, see [40, 41]. Learning curves has been used to describe costs, performance, time, production, etc. It has been discovered that the direct labour work to produce one unit decrease at a uniform rate. We do not require a control group since program one is the control, while program two and three are the treatment programs. This rate is called learning rate, and is the actual decline per doubling of production. The learning



Figure 4: This program returns Accept or Fail. This program is similar to a real authentication system, where users type their username and password. We are only checking the password, the password in this picture is “*jamu9reS*”, and the user has been accepted.

curve varies from one product to another, and a change in personnel, process, or product disrupts the learning curve.

We will compare the learning curves from the three programs. Since the first program’s feedback to the attacker is whether or not they are accepted as the victim, this program will be used as control program. If the other two programs which provide the attackers with more information do not give better results or learning rate than program one, it might not be possible to learn someone’s typing features.

In the experiment design it is very important that the attacker’s learning curve is disrupted. If the learning curve is not disrupted then the order of the program gets important, and we are unable to see if it is possible to learn someone’s typing features.

The participants acting as attackers was placed into three groups, and the three groups had three different experiment setups. The order of the program was different, which password tested in which program was also different, and the victim was different for each program. This means that for each new program the attackers tested, they had a new password and victim to imitate. It also means that they could not use information about the password and or victim they might have learned from the previous programs. So if one attack group start with program three, they can not use the information about the victim or password in the subsequent programs. If the learning curve theory is correct, then there should be a new learning curve for each program. Since we change the victim and password for each program, we should be able to see if program three and two give better results than program one, in terms of learning curve and false acceptances.

Since we had a limited number of participants in the imitation experiment we needed an experiment design that does not require many participants. We decided to go for a time-series design, but since we are afraid that the program order does count we could not set the baseline before the treatments. And it is of no use comparing another password, since peoples typing features in static keystroke dynamics are strongly connected with the typed text. We therefore decided to go for a design similar to the alternating

treatment design, which means that we try different treatments in a time-series. We do not require a control group since program one is the control, while program two and three are the treatment programs. Instead of alternate between two treatments, we have three groups that tested the three programs in different order. One group could test program three, then program two and finally program one. Another group tested program one, then program three and finally program two.

We had three participants acting as victims in the imitation experiments, the three victims typed three passwords thirty times. This resulted in a total of 270 typings, we calculated the reference profile for each password/victim combination in R , which is a free statistical program, see [42]. We removed outliers that where 1.5 standard deviations above mean or 2.5 standard deviations below mean, as described in Section 7.1.12. Although the upper and lower limits are probably not optimal, we believe it is better to remove outliers from the standard deviation, than risking that some standard deviation values are abnormally large. Although features with a high standard deviation can and probably should be removed in a real system, it could give bad results in our experiments since we are using a global threshold. A password with fewer features than other passwords with the same threshold would automatically return a better distance score, which makes it easier to imitate the specific user.

We created three programs in C# for the imitation experiment. They had similar design as the program in the authentication experiment, it always shows the password, the input password characters are shownen as *, it has a login button like real login programs, it shows the written password to users so that they can see what they typed, see Section 6.1.2 for the authentication program design. The user feedback is however different, the first program gives a *Accepted / Fail* feedback to the attacker, see Figure 4. The second program returns the distance metric score, see Figure 5. And the third program visualize the typing and return the same score as program two, see Figure 6.



Figure 5: This program returns a score. It is similar to imitation program one, but it returns the score instead of an “Accepted/Fail”. This means that attackers will get a little more information about how their typing is compared with the victim’s typing.

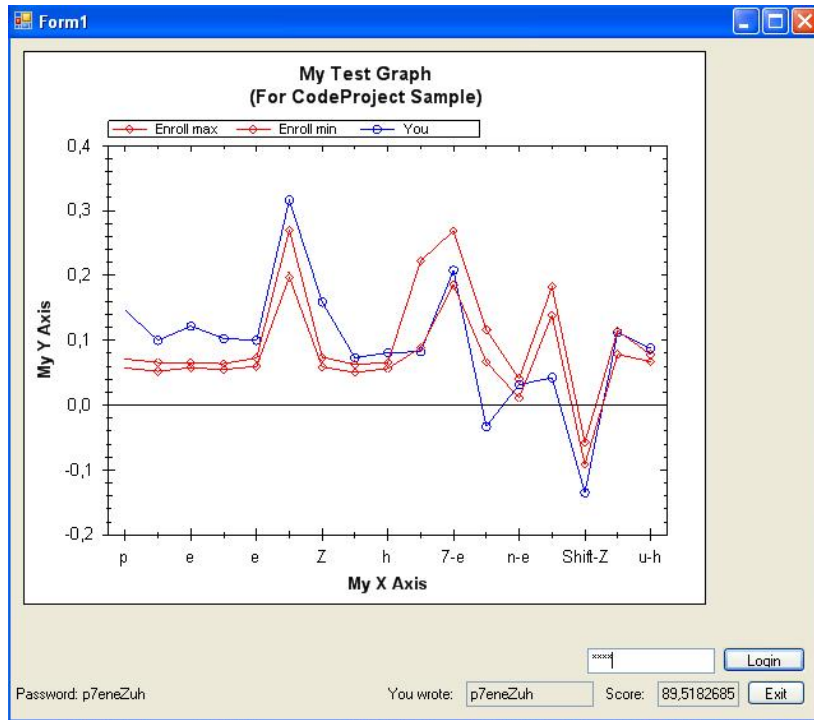


Figure 6: This program returns the same score as imitation program two, but it also visualize the victim's typing and the attackers typing in a graph. The graph's left side is the durations, and the right side is the latencies. The victims typing is visualized as two graphs, the upper graph is one standard deviation above mean, and the lower is one standard deviation below mean, points with a small gap between the victim's graphs are more important than points where there is a large gap. We instantly see that the victim is holding the keys too long.

6.2.3 Participants

We had three participants that were used as victims in the imitation experiment, these three participants needed to type three passwords thirty times, a total of ninety password typings for each participant. We created nine templates from these password typings. The three participants are good typists, older than 25 and all three are connected to HIG.

We also had nine participants acting as attackers in the imitation experiment. These participants were mainly men in their twenties, but we also had a few female participants. Most of the participants are very good typists, with a few exceptions.

7 Analysis

The majority of our participants in the authentication experiment are very good typists, with a few inexperienced typists. 21 participants were participating in the authentication experiments, some produced less than 20 typing samples and some produced more than 100 typing samples. The participants produced about 2000 typing samples, with an average of 95 typing samples for each participant. 16.9 % of the password typing were typed wrong, which means that the False Rejection Rate should be higher than our results in this chapter, since we do not use these mistypings in our calculations.

7.1 Distance metrics tried in the authentication experiment

We needed a verification system based on keystroke dynamics in the imitation experiment, we therefore conducted the authentication experiment. The authentication experiment was designed to find a good distance metric. It was designed to capture the typing features, so we could test several distance metrics and compare the results between them. A good distance metric has a small intra class distance and a large inter class distance. It should also be possible to use it in the imitation experiment. We used the Equal Error Rate (ERR) to see which distance metrics that had a good performance.

The tested password was “p5uxAc6emm” and consist of 10 characters and 11 keystrokes, 11 keystrokes means that we had 21 features to compare. The templates was generated with the 12 first password typings, and we removed outliers that were 1.5 standard deviations above and 2.5 standard deviations below mean, as described in Section 7.1.12. The standard deviations and average values were recalculated, and outliers were removed again. This was done until no further outliers could be removed.

7.1.1 Euclidean distance metric

The Euclidean distance metric is very common, and we tried it in our experiment.

$$D(X, Y) = \sqrt{\sum_{i=1}^n (X_i - Y_i)^2} \quad (7.1)$$

X is the authentication vector, Y is the stored reference vector.

This approach resulted in an EER of 43%.

7.1.2 Manhattan city block distance metric

The Manhattan city block distance metric is also widely used, see [43, 8]. We tried it in our experiment.

$$D(X, Y) = \sum_{i=1}^n |X_i - Y_i| \quad (7.2)$$

X is the authentication vector, Y is the stored reference vector.

This approach was a bit better than Euclidean, with an EER of 41.5%.

7.1.3 Euclidean distance metric with standard deviation

We tried to improve the Euclidean distance metric by dividing with the standard deviation.

$$D(X, Y) = \sqrt{\sum_{i=1}^n \frac{(X_i - Y_i)^2}{\sigma_i}} \quad (7.3)$$

X is the authentication vector, Y is the stored reference vector and σ is the standard deviation for each feature i .

This improved the results with an EER of 33.3%, but if we use the original standard deviation, and only improve the average values by removing outliers then we get an EER of 32.2%.

7.1.4 Manhattan city block distance metric with standard deviation

We also tried to improve the Manhattan city block distance by dividing with the standard deviation, as done in [6]. They got an EER of 8% with a global threshold, and EER of 4% with individual threshold.

$$D(X, Y) = \sum_{i=1}^n \frac{|X_i - Y_i|}{\sigma_i} \quad (7.4)$$

X is the authentication vector, Y is the stored reference vector and σ is the standard deviation for each feature i .

This gave an EER of 27%, but we could reduce it to 23.5% if we only removed outliers from the average values and kept the original standard deviation values. We also tried to improve this further by dividing with the variance instead of the standard deviation, the variance is σ^2 , this gave worse results, with an EER of 38.2% when we used the original standard deviation. Our results of 27% and 23.5% are far worse than 8%, but we had a shorter password.

7.1.5 Mahalanobidistance distance metric

The Mahalanobi distance metric is the Euclidean divided by the variance, instead of the standard deviation as done in Equation 7.3.

$$D(X, Y) = \sqrt{\sum_{i=1}^n \frac{(X_i - Y_i)^2}{\sigma_i^2}} \quad (7.5)$$

X is the authentication vector, Y is the stored reference vector and σ is the standard deviation for each feature i . This gave an EER of 31% , but if we use the original standard deviation values before we remove outliers and only improve the average values, we managed to get an EER of 26.5% . Not as good as the Manhattan with standard deviation in Equation 7.4, but second best. We also tested without taking the square root of the sum, this gave the same result, but it was very threshold sensitive.

7.1.6 Cosine correlation coefficient

We tried to use the Cosine correlation coefficient also known as the interior angle θ as a distance metric, see [44].

$$\theta = \cos^{-1}\left(\frac{X \cdot Y}{\|X\| \|Y\|}\right) \quad (7.6)$$

X is the authentication vector, Y is the stored reference vector.

This distance metric was not suited in the imitation experiment. It had an EER of 44.6%, which is worse than the Euclidian distance metric, and it was therefore not accurate enough for further usage.

7.1.7 The normalized minimum distance classifier

We tried the normalized minimum distance classifier as used in [20] with good results.

$$D(X, Y) = \frac{(X-Y)^t(X-Y)}{\|X\|\|Y\|}, \text{ equals:}$$

$$D(X, Y) = \frac{\|(X-Y)\|^2}{\|X\|\|Y\|}, \text{ equals:} \quad (7.7)$$

$$D(X, Y) = \frac{\sum_{i=1}^n (X_i - Y_i)^2}{\sqrt{\sum_{i=1}^n (X_i)^2} * \sqrt{\sum_{i=1}^n (Y_i)^2}}$$

Where all norms are Euclidean. This gave us an EER of 39.9%, which is better than Euclidean but not good enough for our imitation experiment. The strange thing is that the authors in [20] got very good results with the normalized minimum distance classifier, with a False Rejection Rate of 8.1% and a False Acceptance Rate of 2.8%. They did not publish their EER, but it should be somewhere between FAR and FRR. There are some differences between our authentication experiment and their experiment, we had 21 participants and they had 14 valid users and 25 invalid users. Our participants shared a password of ten characters and they had the participant's names as password with lengths between 11 and 17 characters. They calculated the reference template with 30 typing samples, we used 12. They updated the template weekly with the last 30 typing samples as the new reference template, we did not update our templates. They had 539 legitimate login attempts, we had 1468. They had 768 attack attempts, we had 32133. They used a shuffling technique to improve their reference template, see Section 6.1.2. We used statistics which was proved better than the shuffling technique in [28]

7.1.8 Maximum and minimum distance metric

We also tried to use the maximum and minimum distance metrics.

$$D(X, Y) = \max(|X_i - Y_i|), i = 1, 2 \dots n \quad (7.8)$$

$$D(X, Y) = \min(|X_i - Y_i|), i = 1, 2 \dots n \quad (7.9)$$

X is the authentication vector, Y is the stored reference vector.

The maximum distance metric gave an EER of 41.5%. And the minimum distance metric was slightly better with an EER of 38.5%.

7.1.9 Distance metric with probability of letter

We tried to use the probabilities for each letter in written English texts, with the distance metric:

$$D(X, Y) = \sum_{i=1}^n \frac{|X_i - Y_i|}{P(L_i)} \quad (7.10)$$

Where X is the test vector, Y is the template, $P(L_i)$ is the probability of letter i.

We calculated the probability based on occurrences of letters in the books "A Doll's House" [45], and "The Adventures Of Sherlock Holmes" [46]. With an online ligature counter [47]. The probabilities for the letters in our password "p5uxAc6emm" is in Table 5, the rest of the letters are in Appendix A. This gave an EER of 45.8%. The numbers 5 and 6 were rarely used in the books, so we gave them the same value as x. The shift key

Table 5: Letter probability

Letter:	Occurrences	Probability
p	7893	1.1%
u	16647	2.3%
x	646	0.1%
a	42740	6.0%
c	11785	1.6%
e	64486	9.0%
m	13844	1.9%

The probabilities of the letters in our password. The occurrences is the number of times the letter occurred in the books “*A Doll’s House*” [45], and “*The Adventures Of Sherlock Holmes*” [46]. The probability is the percentage of occurrences divided by total number of letters in the books.

was not counted for, so we tried to give it the value 0.3%, which is a bit low. All these values can be adjusted to improve the results.

We modified this, and tried to multiply instead of divide:

$$D(X, Y) = \sum_{i=1}^n |X_i - Y_i| * P(L_i) \quad (7.11)$$

This gave better results and an EER of 35.2%.

We also tried modifying this further by dividing with the standard deviation.

$$D(X, Y) = \sum_{i=1}^n \frac{|X_i - Y_i| * P(L_i)}{\sigma_i} \quad (7.12)$$

This resulted in a great improvement, to an EER of 26.5%, but with the original standard deviation we got an EER of 24.7%.

7.1.10 Interval metric

We tried a distance metric that counts the number of timings that are outside an interval of Z standard deviations above and below mean. We increase a counter by one for each feature that is outside the accepted interval, when all features in the freshly typed password are checked we know how many features that had timings outside the accepted interval. If no feature timings are outside the interval, it means that the freshly typed password is probably typed by the correct user. In the distance metric we divided by the total number of features and got a score between zero and one, where zero is best and one is the worst. The total number of features is the number of characters multiplied by two and subtracted by one, where the shift key is counted as a character. All characters have a duration, but the last character does not have a latency value. Our password “*p5uxAc6emm*” had a total of 21 features.

When no outliers are removed, we got an EER of 32% with $Z = 1$, and an EER of 26% with $Z = 2$, see Table 6.

In Section 7.1.12 we could sometimes improve the results by using the original standard deviations, and only improving the average values through outlier removal. We removed outliers from the mean that were 1.5 standard deviations above and 2.5 standard deviations below. This technique got EER of 26% and 25%, when $z = 1$ and $z = 2$. An improvement from when no outliers are removed, especially when z is one, this is

Table 6: EER for the interval metric

Above/Below	$z = 1$	$z = 2$
None	32%	26%
Avg. only	26%	25%
1.5/2.5	29%	22%
1.5/1.5	29%	22%
3/3	31%	25.3%
2.5/2.5	32.5%	24%

The EER for the interval metric with two different z values and various outlier removals. Avg. only is when only the average values are improved and the original standard deviations are used. The average values are improved by removing outliers 1.5 standard deviations above and 2.5 standard deviations below mean. The other outlier removal techniques improve both standard deviation and average values.

logical since $z = 1$ has a narrower accepted interval than $z = 2$, and therefore needs a more accurate mean. We also tested several values for the outlier removal. We removed outliers 1.5 standard deviations above and below, and got an EER of 30% and 27% when $z = 1$ and $Z = 2$. We removed outliers 3 standard deviations above and below, and got an EER of 32% and 26% when $z = 1$ and $z = 2$. In Table 6 is the EER when outliers are removed 1.5 standard deviations above and 2.5 standard deviations below mean.

In Table 6 we see that $z = 2$ generally performs better than $z = 1$, for all the tested outlier removals. We also tested $z = 3$, but that had worse results. When we removed outliers 1.5 standard deviations above and 2.5 standard deviations below we got an EER of 22% when $z = 2$, and an EER of 26% when $z = 3$.

The interval metric performed best when z was two, and we improved both standard deviation and average values in the reference template. We got the best results when we removed outliers that were 1.5 standard deviations above and 2.5 standard deviations below mean, but equally good results was achieved when outliers are removed 1.5 standard deviations above and below mean. The problem with the interval metric is that it is more difficult to find the correct threshold, it is very coarse and a slight difference will give a very different result. However, it was the best distance metric in the authentication experiment with an EER of 22%.

7.1.11 Pearson correlation coefficient

We tried to use the Pearson correlation coefficient as done in, [44]. But that resulted in an EER of 59.9%, which is very bad. The equation we used was:

$$D(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2 * \sum_{i=1}^n (Y_i - \bar{Y})^2}} \quad (7.13)$$

The Pearson correlation coefficient is widely used to measure association between two vectors, but it seems like we had too few features or the participants typing was too similar for this distance metric.

7.1.12 Remove outliers

An outlier is defined in [48] as “a point, which varies sufficiently from other points such that it appears to be generated by a different process from the one governing the other points”. In keystroke dynamics outliers are timings that differs from the rest of the timings

for a specific feature.

Distance metrics that does not take standard deviation or variance into account greatly benefit from removing outliers as described earlier. However, distance metric that does take standard deviation or variance into account could actually perform worse when outliers are removed. This is because the variance is reduced. These distance metrics will in some cases perform better with no outliers removed, than with some outliers removed. Nevertheless, we can improve the EER by calculating the standard deviation or variance before outliers are removed, and then remove outliers. And using the new average and old standard deviation as the template.

We used the Manhattan city block distance metric divided by the standard deviation in our calculations. When we improved both average values and standard deviation values in the template by removing outliers, as proposed by the authors in [28]. We first calculate the standard deviation values and average values for all features in the enrolled password. In our authentication experiment, we used the 12 first passwords as the enrolled passwords. This means that each feature had 12 timings that we could calculate the standard deviation and average on. When the average and standard deviation were calculated, we could see if any points were outside the accepted boundaries. The accepted boundaries were N number of standard deviations above and below mean. Points outside these accepted boundaries were removed, and the standard deviation and average values were recalculated. With the new standard deviation values and average values, we checked if any points had fallen outside the new boundaries. This was performed until all remaining points were inside the accepted boundaries.

The EER was 27% when we removed outliers 1.5 standard deviations above and 2.5 standard deviations below mean, as described above. When we remove outliers 1.5 standard deviations above and below mean the EER is 27.1%. When we removed outliers 2.5 standard deviations above and below mean, we get an EER of 23.9%, and 23.5% when we removed outliers 5 standard deviations above and below mean. The same result when no outliers are removed which is an EER of 23.5%, see Table 7. We see that the best results are achieved when no outliers are removed, when both standard deviation values and average values are improved.

These results could be improved by using the original standard deviation values, and only improving the average values by outlier removal. We got an EER of 23% when we removed values that were 1.5 standard deviations above and 2.5 standard deviations below. Which is slightly better than when no outliers are removed, see Table 7. It is important to remember that we stop the removing of outliers if the standard deviation value gets below 0.001, without this we could risk that almost all points are removed in extreme cases.

We also tested which of latency and duration did perform worse when the outliers are removed. We first tried to remove outliers above 1.5 standard deviations, and below 2.5 standard deviations. The EER for duration was 27.4%, and the EER for latency was 35.3%. We see that the latency does perform worse than the duration, and a distance metric that only uses duration is actually better than one that uses both latency and duration when outliers are removed. When no outliers are removed, only duration timings give an EER of 28.2% and only latency timings give an EER of 30.9%. This means that the duration actually benefits from removing outliers, and the latency works best when no outliers are removed.

Table 7: Outlier removal effects on EER

Above/Below	SD or AVG	EER
None	None	23.5%
1.5/2.5	Both	27%
1.5/1.5	Both	27.1%
2.5/2.5	Both	23.9%
5/5	Both	23.5%
1.5/2.5	AVG	23%
2.5/2.5	AVG	23.5%
1.5/1.5	AVG	23%

The EER is calculated with various outlier removal techniques. The distance metric is the Manhattan City Block distance metric with standard deviation, from Section 7.1.4. SD is standard deviation values in the template and AVG is the average values in the template. We improved both standard deviation values and average values in most of the tests, but we also tried to keep the original standard deviation values and only improve the average values. Above/Below is the accepted boundaries in N number of standard deviations above/below mean.

When distance metrics with standard deviation are used, it is sometimes best to not remove any outliers, see the ROC curves in Figure 7. The green curve is when we improve the average values and standard deviations by removing outliers 1.5 standard deviations above and 2.5 standard deviations below mean. The red curve is when the original standard deviation is used. And the black curve is when no outliers are removed. Distance metrics without standard deviation will normally benefit from removing outliers, like in Figure 8 where we use the Euclidean distance metric. The green curve is when we remove outliers, and the red curve is when no outliers are removed. It is easy to see that the Euclidean benefits from removing outliers, the same is true for the Manhattan city block distance without standard deviation.

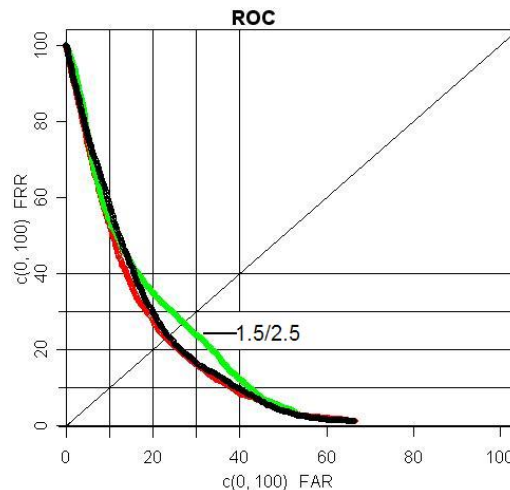


Figure 7: The ROC curve for various outlier removals with the Manhattan city block distance metric divided by the standard deviation. Green is when we remove outliers that are 1.5 standard deviations above and 2.5 standard deviations below mean and use the new standard deviations. Red is when we keep the original standard deviations. And black is when no outliers are removed.

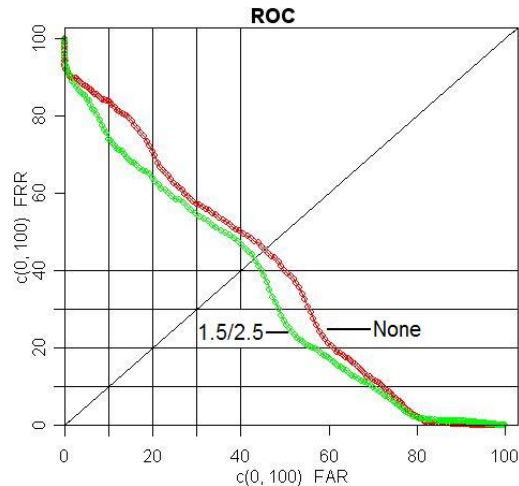


Figure 8: The ROC curve for various outlier removals with the Euclidean distance metric. Green is when we remove outliers that are 1.5 standard deviations above and 2.5 standard deviations below mean, and red is when no outliers are removed.

If we use the standard deviation in a distance metric it seems like it is a good idea to not remove any outliers, but this approach does not guarantee good performance. In Figure 9 is a template for the password “*p7eneZuh*”, which we used in the imitation experiment. In this figure the black curve is the enrolled average value, the upper blue curve is one standard deviation added with the average value, the lower blue curve is one standard deviation subtracted from the average value. In this graph point nine on the x scale is very strange. The standard deviation is very large, and the average value is much higher than where the majority of the points are. Although there were thirty password typings during enrolment, only two large values were needed to make *h*'s duration (point nine) in the reference profile wrong (latency between *p-7* (10), *7-e* (11) and *e-shift* (14) are also wrong (point 10, 11 and 14)). When outliers 1.5 standard deviations above and 2.5 standard deviations below mean are removed, the reference profile looks very different, and much more correct. Figure 10 visualize the timings after outliers have been removed.

We discovered this problem when we tried to use the reference profile before outliers were removed, and it then became fairly easy to be authenticated as this user. It is a very simple reason for this, since we divide by the standard deviation in the distance metric, and four features had very large standard deviations. This means that the distance metric had four less features to check. However, two points were better in Figure 9 than in Figure 10. Too many timings were removed from the latency between *z-u* (16) and *u-h* (17), the standard deviation values are very small and the legitimate user will find it difficult to be authenticated. With a global threshold, both situations are bad. Fewer features means a smaller result from the distance metric, and with a global threshold it was easier to imitate this user. Too small standard deviations will generate a high result from the distance metric, and the legitimate user will find it difficult to be accepted by the system.

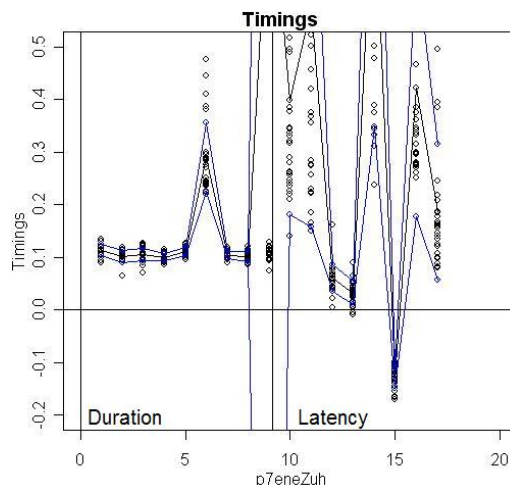


Figure 9: The reference profile for one participant with the password “p7eneZuh”, no outliers are removed. The duration timings are first, and then the latencies.

7.1.13 Overlap

It is easier to distinguish users on unique features, than general features. We therefore tried to calculate the overlap of all enrolment samples, by comparing two and two enrolled templates. We had 21 participants for this test, giving $\binom{21}{2} = 210$ pairs to compare. And for each pair E_i, E_j where $i \neq j$, we calculate the overlap. The overlap is where the participant i interval overlaps participants j 's interval. The participant's interval is the values between a maximum and minimum value. Where the maximum and minimum value is enrolled mean + one standard deviation, and enrolled mean - one standard deviation. This is visualized in Figure 11, where O is the overlap for one feature between two participants. We used the formula in Equation 7.14 to calculate the overlap values for all the pairs.

$$O = \max(0, \min(Y1_i, Y2_i) - \max(X1_i, X2_i)) \quad (7.14)$$

Where the O is the overlap. $X1$ and $Y1$ is the minimum and maximum values for participant one. $X2$ and $Y2$ is the minimum and maximum values for participant two. i is the feature.

When we calculated the overlap for all participant pairs and added together the overlaps, we created a graph where the sum for each feature is drawn, see Figure 12. The duration is the 11 first points, and the latency is the 10 last points. We see that the duration values are relatively small compared to the latency values, except for the shift-key duration, shift-key latency, e-m latency and m-m latency.

The overlap graph is very similar to the standard deviation graph, which is logical since a large standard deviation is more likely to give a large overlap.

We tried to use the overlap in a distance metric, we used the Manhattan distance and divided with the overlap.

$$D(X, Y) = \sum_{i=1}^n \frac{|X_i - Y_i|}{O_i} \quad (7.15)$$

Where the O_i is the overlap for feature i .

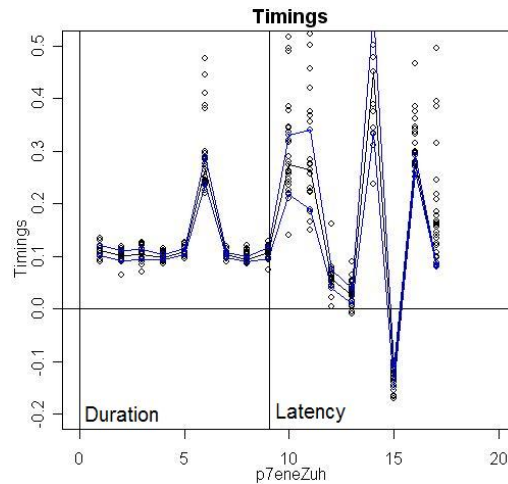


Figure 10: The reference profile for one participant with the password “p7eneZuh”, outliers that are 1.5 standard deviations above and 2.5 standard deviations below are removed. The duration timings are first, and then the latencies.

This gave us an EER of 28%. We improved the values further by removing features that had high overlap values. By removing seven features that had the largest values we got an EER of 21%. By removing the ten largest values we got a small improvement, but not significantly smaller. Removing a few features with the largest overlap seems like a good idea, this can be done with other distance metrics as well. We tried to remove seven features from the Manhattan City Block distance with outliers removed from 1.5 standard deviations above and 2.5 standard deviations below, the EER was improved from 27% to 24.5%. Features with a large overlap will generally have a large standard deviation, this means that if features with a large standard deviation are removed, the distance metric will probably be more accurate.

When we removed seven values, only latency features was removed. Since the standard deviation graph is so similar to the overlap, this also means that these seven features also have the highest standard deviation values.

We believe that people are more consistent on the duration than on the latency when they type a new text, since they have to learn the password. This tells us that an adapting mechanism could greatly improve our results.

7.2 Imitation Experiment

7.2.1 Threshold

In a verification system based on keystroke dynamic, users are being compared against a stored reference template for the specific user. If the distance score returned from the distance metric is smaller than a threshold the person typing is accepted as the user, and if the distance score is higher the person typing is denied access to the system. We are using a global threshold in our experiments, but the studied literature suggests that an individual threshold will give the best results, see [6].

A global threshold means that all users have the same threshold, while an individual threshold is when the threshold is adjusted to fit a specific user. It is easy to choose an individual threshold to get the desired False Rejection Rate (FRR). A global threshold on

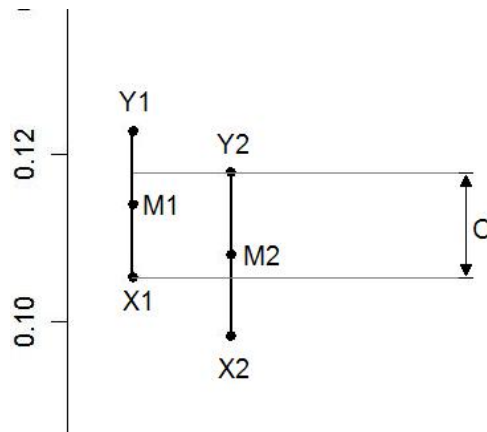


Figure 11: Overlap between two participants, for the “a”’s duration in the password “*jamu9reS*”. M1 is participant one’s mean, M2 is participant two’s mean. X1 and Y1 is participant one’s minimum and maximum value. X2 and Y2 is participant two’s minimum and maximum value. The overlap is O.

the other hand is more difficult, since a specific threshold can be perfect for one user, and useless for another user. If all users were using the same password we could find the best global threshold as we did in the authentication experiment to find the EER. But the users should have different passwords, and we can not compare other legitimate user’s password typing against each other, since they have different passwords. It is of course possible to find the FRR for the system, by using 2/3 of the enrolment typings as a template, and last 1/3 as login attempts. And then choose a threshold with an acceptable FRR. The problem is that some password/user combinations are easier to imitate than other combinations, and they should have a lower threshold or another password. But it is probably very difficult to discover these situations.

In Figure 13 it is possible to see the typing from the three participants acting as victim in the imitation experiment. There are three graphs from each participant, the mean values, the mean + one standard deviation, and mean - one standard deviation.

We have tested several threshold values in the authentication experiment. It is important to remember that the Manhattan City Block distance metric with standard deviation, which we used in this experiment will get different scores based on the standard deviation value. A large standard deviation will return a small score, and a small standard deviation will return a large score. This is also true for other distance metrics which take standard deviation into account. Outlier removal will obviously decrease the standard deviation, and the distance score will therefore be higher if outliers are removed. This however does not imply that the distance metric perform worse, it means that the distance scores from both legitimate and impostor attempts will be larger, and a higher threshold should therefore be used.

In the authentication experiment we tested which threshold was best suited for the imitation experiment. We removed outliers that was 1.5 standard deviation above and 2.5 standard deviations below mean, as described in Section 7.1.12. We got an Equal Error Rate (EER) of 27.4%, with a threshold of 50.6. We did however choose a threshold of 50, which is much easier to explain to the participant. With a threshold of 50 we got a False Rejection Rate (FRR) of 28.2%, and a False Acceptance Rate (FAR) of 26%. If

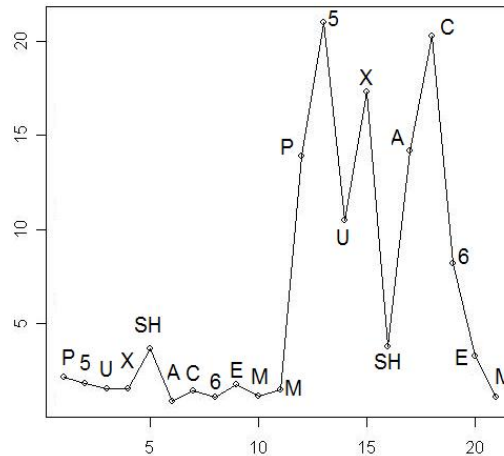


Figure 12: All overlaps are added together, and the sum for each feature is plotted in this graph. The password is “p5uxAc6emm”. The first 11 features are duration and the last ten features are latencies.

we did not remove any outliers, a threshold of 50 would result in a FRR of 0.7% and a FAR of 78%. This means that almost all impostor attempts are accepted. The correct threshold when no outliers are removed is 27. We should probably have chosen a smaller threshold than 50, since there are different password lengths in the two experiments. In the authentication experiment we had 10 characters, and 8 characters in the imitation experiment. All passwords had one upper character, which means that we had 11 and 9 keystrokes respectively in the authentication experiment and imitation experiment. In the authentication experiment we could compare 21 features for each password, in the imitation experiment we had 17 features. A threshold of 50 means that we had a threshold average of 2.38 for each feature, with 17 features that would give a threshold of 40.46, but we decided to use a threshold of 50, since we could not verify this theory and the threshold is not used in our analysis.

In the imitation experiment we used a threshold of 50 in program one, which means that the distance score returned from the distance metric had to be below 50, if program one should accept the typing as the legitimate user. In the other two programs the distance score is returned, and we explain the score results to the participants. We calculated FRR and FAR for each of the three passwords, based on the typing from the three victim participants. We calculated the FRR by using the first 20 typings as a template for each user, and the last ten typings as legitimate login attempts. We calculated the FAR by comparing all typing samples from the other two participants against the same template as in the FRR calculation. In Figure 14, is the ROC curves created from the FRR and FAR values. The three colours red, blue and black are the three different participants and the squares are the FAR and FRR values when the threshold is 50. We see that the red participant has an FAR of zero and an FRR of 0, 10 and 20% when the threshold is 50. Some of the user/password combination has an EER of 0%, and the worst EER is 10%. We get these good results since we only have ten legitimate login attempts and 60 impostor attempts for each password/user combination. But it is clear that a global threshold can not be perfect for everybody, and we see that one participant has an FRR of 0% and a FAR of almost 40% with this global threshold. This participant should have

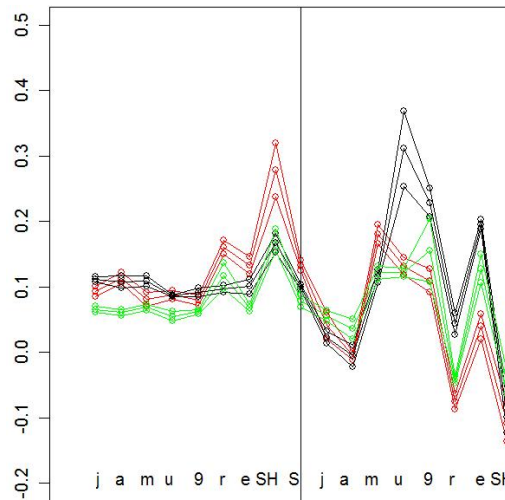


Figure 13: Enrolled user templates for the three participants acting as victims. There are three graphs for each user, mean + one standard deviation, mean and mean - one standard deviation. This graph shows the template for the password jamu9reS. We can see that there are similarities between the participants, but the templates are definitely different.

a smaller threshold, and the participant with 20% FRR and 0% FAR should have a larger threshold.

Individual thresholds could be found by saying that all users should have an FRR of 10 - 20%. In this example we would get an FAR of less than 2% if all users had an individual threshold where the FRR was 20%. But if we had decided the threshold with an FRR of 10%, one user would get an FAR of 25%. Finding the perfect individual threshold is difficult, but it is definitely possible to find a user specific threshold, which is better than the global threshold. We decided to use a global threshold since we do not need the extra accuracy in our imitation experiment, we are only interested in the scores returned from the three programs. In addition, the “*accepted/fail*” answer to the participant based on a threshold is not interesting in the analysis.

We have also discovered that the distance score returned from the distance metric also depends on the password, and not only the user. In the imitation experiment we discovered that password three had a higher FAR than the first password, except for one participant. Participant one had a FRR of 0% for all three passwords, and a FAR of 1.6, 23 and 37% for the three passwords. Participant two had a FRR and FAR of 0% for the first password, and a FRR of 10 for the two other passwords, participant two also had a FAR of 12 and 25% for password two and three. Participant three had a FAR of 0% for all three passwords, and a FRR of 20, 10 and 0% for the three passwords. All these results are with a threshold of 50, password one was “*jamu9reS*”, password two was “*bruf9Tr2*”, and password three was “*p7eneZuh*”. The average FRR was 5.5%, and the average FAR was 11%.

7.2.2 Outliers in the imitation experiment

The attackers in the imitation experiment had several strange distance scores, and an attacker could have a distance score of 100, 120, 256, and then down to 112. In this example we see that 256 is much larger than the surrounding values. This value is actually

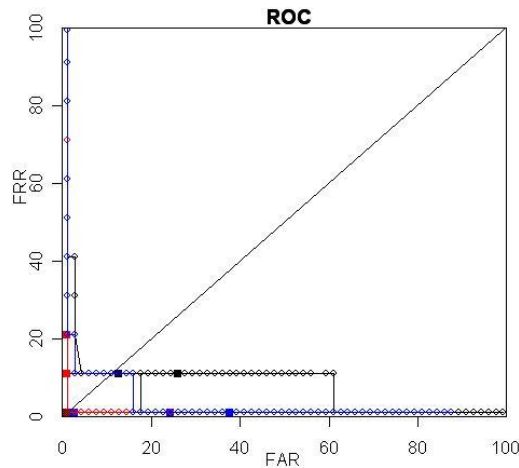


Figure 14: ROC curves from the three participants acting as victims in the imitation experiment. One curve for each password/user combination, and one colour for each participant. The squares are FRR and FAR for a threshold of 50. It is easy to see that a global threshold is not perfect for all users in a keystroke dynamic verification system.

three standard deviations away from the average of all distance scores from the specific attacker/password combination. The authors in [49] suggest that these outliers should be examined carefully, and they believe that automatic removal of outliers is unwise. In addition, they wrote that outliers should generally only be removed if they are caused by recording errors or in setting up the apparatus.

We had one attacker participant who had much higher distance scores than the other attacks, and the participant seemed stressed. Perhaps the participant was afraid of getting a bad score, or the participant had an appointment. The participant had a very high distance score with program two and the password “*jamu9reS*”, but the other two programs and passwords distance scores were also high, compared to the other attacker participants. We therefore believe that the removal of this participant will result in more precise results, since the participant was stressed and not really acting as an attacker.

7.2.3 Learning curves

Learning curve theory originates from observations that workers in manufacturing plants become more efficient as they produce more units, see [40, 41]. The time needed to create a new item decrease with a percentage every time the production is doubled. The percentage is called the learning rate, a learning rate of 80% means that if it takes 100 minutes to create unit number 50, it takes 80 minutes to create unit number 100.

Learning curves has been used to describe production rates, energy efficiency, cost efficiency, labour work, quality improvement, and many other areas where “*learning by doing*” can be applied.

We are using learning curves to check if it is possible to learn someone's typing features, by having three programs with different feedback levels. We are in this setting interested in checking if the program with most feedback has the smallest learning rate. A small learning rate means that the distance score will be reduced fast, and a large learning rate means that the distance score will be reduced slowly if reduced at all. A learning rate below 100% means that there is a reduction, and above means that there

is an increase. If the program with most feedback has a smaller learning rate than the other programs, it means that it is probably possible to learn someone's typing features. See Equation 7.16.

$$\text{Score}(x) = S_0 x^{(\log LR)/(\log 2)} \quad (7.16)$$

Where LR is the learning rate in percentage. S_0 is the distance score from the first password typings. And x is the password typing number.

With the original typings We use the original password typings from each attacker, with various number of typings from each attacker, and the same number of typings from each program. We calculated the learning rates for all attacker/program combinations. Program one got an average learning rate of 88.8%, program two got an average learning rate of 99.8% and program three got an average learning rate of 85.1%. Program three has a lower learning rate than program one and two, which means that attackers typing will be closer to the victims typing after n password typings. See Figure 15 for the learning curves from these learning rates. If attackers using program one, two and three all started with a distance score of 100. Then after 200 password typings, attackers with program one would have a distance score of 40, 48 with program two, and a distance score of 29 with program three. The learning curves indicate that the distance scores will become zero after N typings, but there must for obvious reasons exist a limit higher than zero. Since not even the legitimate user will get zero distance score.

The three attack groups performed differently, attack group A had an average learning rate of 89.2%, 90.2% and 81.3% for program one, two and three. Attack group B had 83.2%, 82.6% and 87.8% for program one, two and three. Attack group C had 94.0%, 99.6% and 86.3%. We see that attack group B performed worst with program three, but the other two groups performed best with program three.

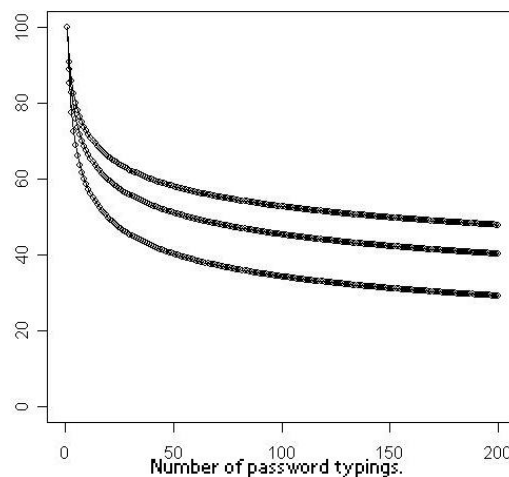


Figure 15: Learning curves for the three programs with the original values, when the five first attacker typings removed. The highest typing is program two, the lowest curve is program three. This means that attackers with program three will get a lower distance score after n password typings, than if they were using program one or two.

Although it seems like program three performs better than program one, the learning rates from the three programs are not different enough to be conclusive. We calculated learning rates for the attackers against the three different passwords, and learning rates from attackers against the three different victims. These learning rates between the different passwords and d between the different victims were more different than the learning rates from the different programs. It is therefore safer to conclude that both victim and password influence the attackers learning rate more than the program. However, none of these learning rates are different enough to be conclusive.

A Student's t test determines whether a statistically significant difference exists between two means, see [36]. We performed a Student's t test on the learning rates from the three different programs, and the learning rates could be from the same mean. The hypothesis H_0 stands, see [50, 51].

When we remove the five first typings When we remove the five first password typings from each attacker, we remove most of the typings where attackers learn the password. It is in these five typings that most of the hesitations occur. We calculated the learning rates for all attacker/program combinations, and program one gets an average learning rate of 96.6%, program two gets an average learning rate of 96% and program three gets an average learning rate of 94%. Program three has a slightly lower learning rate than program one and two, which means that attackers typing will be closer to the victims typing after n password typings. See Figure 16 for the learning curves from these learning rates. If attackers using program one, two and three all started with a distance score of 100. Then after 200 password typings attackers with program one would have a distance score of 77, 73 with program two, and a distance score of 62 with program three.

The three attack groups performed differently, attack group A had an average learning rate of 98.3%, 97.8% and 95.6% for program one, two and three. Attack group B had 96.4%, 90.3% and 93.6% for program one, two and three. Attack group C had 95.1%, 99.9% and 92.7%. We see that all three groups performed better with program three than program one, but Attack group C performed worst with program two. It is important to remember that attack group C had one attacker who performed badly, see Section 7.2.2.

Although it seems like program three performs better than program one, the learning rates from the three programs are not different enough to be conclusive. In fact the learning rates from the three different password and victims are slightly more different than the programs. A Student's t test determines whether a statistically significant difference exists between two means, see [36]. We performed a Student's t test on the learning rates from the three different programs, and the learning rates could be from the same mean. The hypothesis H_0 stands, see [50, 51]

Removed one participant We removed the participant that was stressed, and had a very high distance score, see Section 7.2.2. The rest of the dataset was similar to the original dataset, but group C had two participants, while group A and B are original. We calculated the learning rates for all attacker/program combinations, and

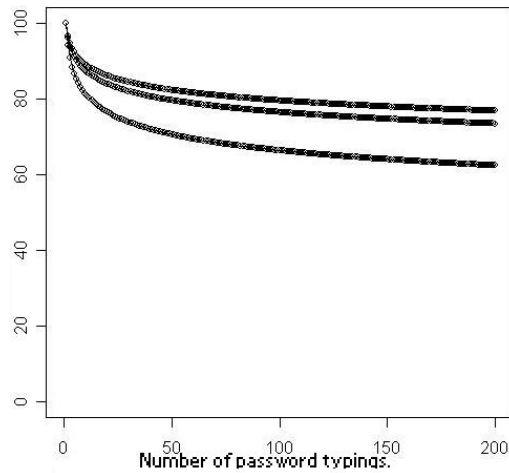


Figure 16: The learning curves for the three programs when the five first attacker typings removed. The highest typing is program one, the lowest curve is program three. This means that attackers with program three will get a lower distance score after n password typings, than if they were using program one or two.

program one gets an average learning rate of 89.7%, program two gets an average learning rate of 91.5% and program three gets an average learning rate of 85.0%. Program three has a lower learning rate than program one and two, which means that attackers typing will be closer to the victims typing after n password typings. See Figure 17 for the learning curves from these learning rates. If attackers using program one, two and three all started with a distance score of 100. Then after 200 password typings attackers with program one would have a distance score of 44, 51 with program two, and a distance score of 29 with program three. The three attack groups performed differently, attack group A had an average learning rate of 89.2%, 90.2% and 81.3% for program one, two and three. Attack group B had 83.2%, 82.6% and 87.8% for program one, two and three. Attack group C had 100.3%, 106.8% and 86.3%. It is important to remember that attack group C has two attackers, while attack group one and two had three attackers.

Although it seems like program three performs better than program one, the learning rates from the three programs are not different enough to be conclusive. In fact the learning rates from the three different password and victims are slightly more different than the programs. A Student's t test determines whether a statistically significant difference exists between two means, see [36]. We performed a Student's t test on the learning rates from the three different programs, and the learning rates could be from the same mean. The hypothesis H_0 stands, see [50, 51].

The learning curves can indicate that it is possible to learn someone's typing characteristics, since program three had a "quicker" learning curve than program one. Attackers using program three would most of the time get a smaller distance score after some typings than with program one, but the Student's t test has shown that this is not necessarily true, and the H_0 hypothesis can not be rejected. We have also conducted other statistical analysis of the authentication experiment.

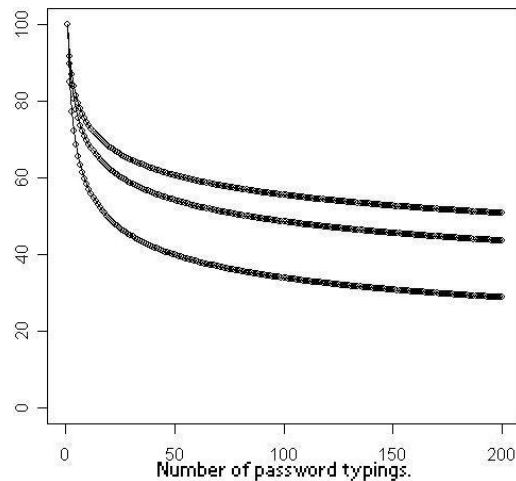


Figure 17: Learning curves for the three programs with eight attackers, one attacker is removed, and the rest of the dataset is original values. The highest curve is program two, the lowest curve is program three. This means that attackers with program three will get a lower distance score after n password typings, than if they were using program one or two.

7.2.4 Descriptive statistics

Before statistical analysis can begin, we have to describe the data we are performing the analysis on. Like finding points of central tendency, variability, and which variables are related.

We had nine participants acting as attackers in the imitation experiment, and the nine attackers were put in three different groups, and the groups tried the programs in different order. Some of the attackers were more eager to imitate the victims than others, and the eager attackers tried more often than the less eager attackers. This means that some of the attackers contributed with more data than other attackers. The analysis will probably depend more on the eager attackers since they contributed with more data, this is not necessarily bad, since the eager attackers are probably more motivated and a real attacker would be very motivated. The attackers also contributed with various numbers of distance scores from the three programs, like 21 from program one, 20 from program two and 25 from program three. We analyse on the same number of distance scores from all three programs per attacker, we would in this example take all 20 distance scores from program two, and the 20 first scores from program one and two. Another attacker could have 22 distance scores from all programs, and some had fewer, the least eager attacker had 13 distance scores, and the attacker with most distance scores had 26.

With all nine attackers and equal number of distance scores from the three programs we had 169 distance scores from each program, a total of 507 distance scores from all three programs. We had 169 distance scores from each password, and 169 distance scores against each victim. The distance scores have a median of 82, and a mean of 110. The mode is 63, which is the most frequently occurring score, see [36]. The distance score has a range of 595, which is the difference between the lowest and highest value. The first quartile is 58, and the third quartile is 128, the first quartile is where 25% of the score values are below, and the third quartile is where 75% of the values are below. The interquartile range is $128 - 58 = 70$, this means that 70 is the range for the middle 50% of

Table 8: Correlation table.

	Number	Score	Program	Password	Victim	Attacker
Number	1	-0.27	0	0	0	0.1
Score	-0.27	1	-0.03	-0.2	0.3	0.3
Program	0	-0.03	1	-0.03	-0.03	0
Password	0	-0.2	-0.03	1	-0.03	0
Victim	0	0.3	-0.03	-0.03	1	0
Attacker	0.1	0.3	0	0	0	1

The correlation between the various variables in the imitation experiment.

the score values. The standard deviation is 85, and the variance is 7221 for the distance score of all three programs. The distribution of the distance scores are positively skewed, see Section 7.2.5.

In the regression analysis it is important that the independent variables are uncorrelated with each other, see Section 7.2.6. Table 8 shows the correlation table for the data, where number, program, password, victim and attacker are the independent variables, and score is the dependent variable.

In the correlation table 8, there is a very small correlation between program, password and victim. This is caused by the various number of distance scores from the attackers, and there is no correlation between these variables when we have an equal number of distance scores from the attackers. The correlation between program and score is slightly bigger than the correlation between program, password and victim. The independent variable that is least correlated with the score is the program, this is probably due to the fact that the three programs have a high distance score variation. The correlation table shows that the experiment could have benefited from a fixed number of scores from each participant, but this could increase the number scores from unmotivated attackers. And the correlation is so small between the independent variables that they are practically uncorrelated. This is also the case with the program and score, there is a negative almost nonexistent correlation between them. Negative means that program two and three will have a lower score than program one, but this correlation is almost nonexistent, and we certainly can not reject the H_0 hypothesis.

7.2.5 Distribution

We created histograms for all three programs, where all distance score distributions were positively skewed. Positively skewed means that the peak is to the left of the midpoint and there is a long tail to the right, where the peak is the interval with most number of scores, see [52, 36, 50]. See the Figures 18, 19 and 20, for the histograms and distributions. In the histograms it is easy to see that program three has more scores between zero and 50, than both program one and two. Since 50 was the threshold, much more attackers got accepted as the victim with program three, than with the other two programs.

When we analysed the original data, with the same amount of scores from all three programs, program three had the lowest mean, but program two had the highest mean and lowest median, see Table 9. There is a great difference between mean and median for program two, this is due to some extreme values, especially from one participant.

If we remove the attacker with the high distance scores, the same participant as in

Table 9: Min, median, mean and max from program one, two and three.

Program:	Min	Median	Mean	Max
One	30.3	93	107.9	406.4
Two	38.9	77.1	120.3	623.7
Three	28.5	81.3	101.7	388.0

The minimum, maximum, median and mean score for the three programs. With the same number of scores from the three programs, but different number of scores from the participants.

Table 10: Min, median, mean and max when we remove one participant.

Program:	Min	Median	Mean	Max
One	30.3	85.6	98.1	406.4
Two	38.9	73.0	85.4	261.0
Three	28.5	75.4	88.4	335.2

The minimum, maximum, median and mean score for the three programs. When we remove one attacker with high distance scores.

Section 7.2.2, the mean and median will be much more similar for program two, but also for the other two programs, see Table 10.

We also tried to remove the first five distance scores from each imitation attempt, with all attacker participants. See Table 11 for the min, max, median and mean values.

7.2.6 Regression analysis

In linear regression analysis we assume that the relationship between the depended variable Y and the independent variable X follow a straight line. And the relationship can be expressed with the formula for a straight line, see [49].

$$Y = \beta_0 + \beta_1 X + \epsilon \quad (7.17)$$

Where Y is the depended variable, X is the independent variable, the password typing number in our experiment. β_0 is the start point when X is zero, β_1 is the growth parameter, how much is the line increased or decreased for every X . ϵ is the random element, and is due to variation or measurement errors.

One of the first analyses we perform is analysis of variance (ANOVA), which answer the question of how much of the variation of the data has been explained by the regression line. We used the original dataset with equal amount of data from the three pro-

Table 11: Min, median, mean and max when we remove the first five distance scores.

Program:	Min	Median	Mean	Max
One	30.3	86.7	95.3	318.1
Two	38.9	73.8	111.4	484.9
Three	28.5	73.2	87.9	256.6

The minimum, maximum, median and mean score for the three programs. When we remove five distance scores from each imitation attempt, with all attack participants.

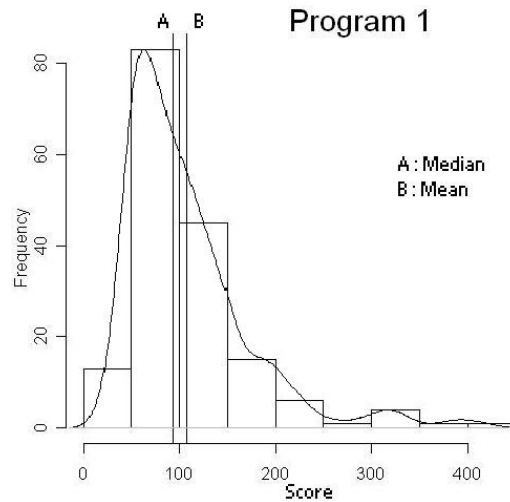


Figure 18: Histogram for program one, when all attack participants are included. A is the median, B is the mean. The distribution is positively skewed.

Table 12: Analysis of Variance Table.

	Df	Sum Sq	Mean Sq	F value	Pr
Number	1	260975	260975	111	2.2e-16
Program	2	30545	15273	7	0.0016
Victim	2	378934	189467	81	2.2e-16
Attacker	8	1683352	210419	90	2.2e-16
Password	2	146772	73386	31	1.7e-13
Residuals	491	1153385	2349		

Analysis of the variance table. The score is the depended variable Y, and the independent variables are program, password, victim, attacker and number. We see that all the independent variables influence the score with a relatively high probability.

grams, and various number of distance scores from the attacker participants. We have the score as the depended variable Y, and the independent variables are program, password, victim, attacker and number, see Table 12 for the ANOVA table.

From the ANOVA table it seems like the program does influence the score. But the residual sum of squares in the table shows that the actual observations do not lie on the regression line, if all observations would lie on the regression line then the residual sum of squares would be zero. This is obviously not possible, but 1153385 could indicate that the regression line is unsuited.

If a regression analysis should be accurate it has to comply with some conditions. The regression should be a linear line, absence of multicollinearity, the errors (residuals) should be independent, have a constant variance and follow a normal distribution. See [49, 52].

Absence of multicollinearity means that variation of the different independent variables should be independent of each other. If one independent variable is adjusted, the other independent variables should be unaffected. This can be checked with a Variance Inflation Factor (VIF) test. If one of the independent variables score more than 10, it

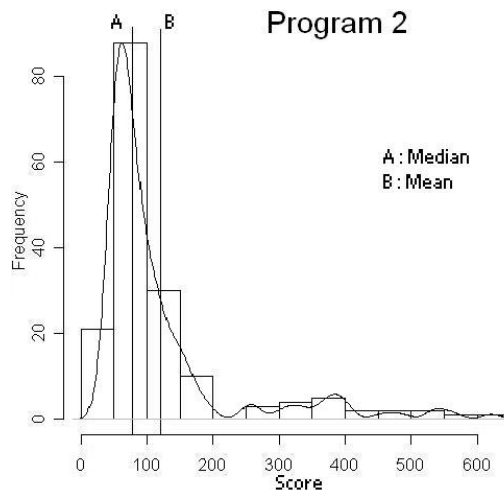


Figure 19: Histogram for program two, when all attack participants are included. A is the median, B is the mean. The distribution is positively skewed.

means that there is a problem with the dataset. The VIF test gave none of the independent variables a score of more than 1.13, which means that there is no multicollinearity problems in the dataset.

A residual plot shows us that the constant variance condition is a problem, see Figure 21.

We need a constant residual variance to achieve a reliable regression analysis, and we therefore had to adjust the regression model. This is done by transforming the depended variable Y , we tried square root transformation and natural logarithmic transformation. The natural logarithmic transformation gave the best results, a constant residual variance, see Figure 22. The ANOVA analysis for the new regression model with $\ln(Y)$ as the depended variable showed that the program still influence the score, and the mean residual square was close to zero. The VIF test was almost equal to the previous VIF test, this means that the transformation of Y to $\ln(Y)$ resulted in a better regression analysis. We are therefore using the transformed regression analysis in the rest of the regression analysis.

The residuals are due to the “*central limit theorem*” normal distributed in large populations. We have 507 samples in the data and that is according to [52] large enough. But we had to check the residuals distribution, which were almost perfectly normal distributed with the transformed regression model, and both mean and median were equally to 0. We also tried to see if the original regression model had a normal distributed, and it was actually left skewed. If the residuals had not followed a normal distribution, the hypothesis testing could have been wrong. We could risk ending up with the wrong conclusion. But with the transformed regression model the residuals were normal distributed, and the hypothesis testing should be correct.

All independent variables that influence the depended variable should be in the model. Other “*unknown*” variables that influence the depended variable Y should be in the random element ϵ . We have checked the residuals against the independent variables in the model, and they seem to be uncorrelated. It is difficult to test this condition, but if we identify all the independent variables that can influence the depended variable, and in-

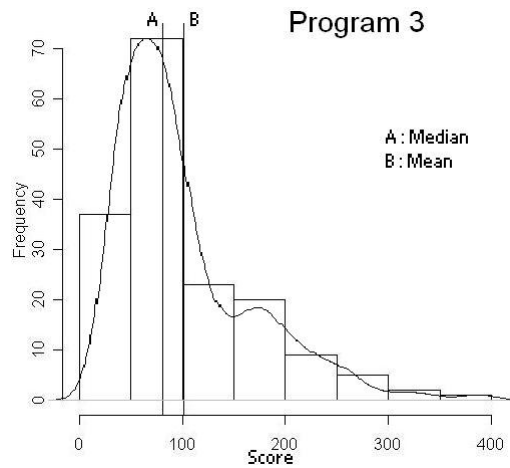


Figure 20: Histogram for program three, when all attack participants are included. A is the median, B is the mean. The distribution is positively skewed.

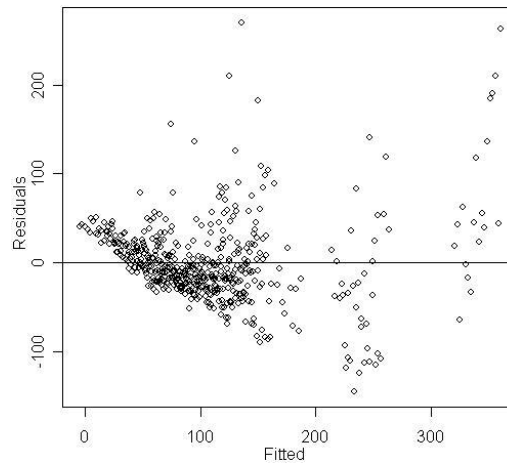


Figure 21: Residuals plot, the residuals are widening to the right. Showing that there is a nonconstant variance.

clude them in the regression analysis, the condition should be satisfied. We can however never be entirely sure that all possible independent variables X that influence Y are identified. Violation of this condition does not mean that the regression analysis is wrong, most regression analysis violate this condition to some degree. Since we have included five independent variables we should be relatively safe.

The last condition is that the regression should be a linear curve, we know that the distance scores were reduced quickly in the beginning when the attacker was learning the password, and it was reduced slowly in the end. The actual distance score looked like the learning curves we analysed in Section 7.2.3. The distance score does therefore not follow a linear curve, it would have reached negative distance score if it had been linear. It is impossible to get a negative distance score, and the transformed model is better, since it follows the exponential curve. Formula for the exponential curve is in Equation 7.18.

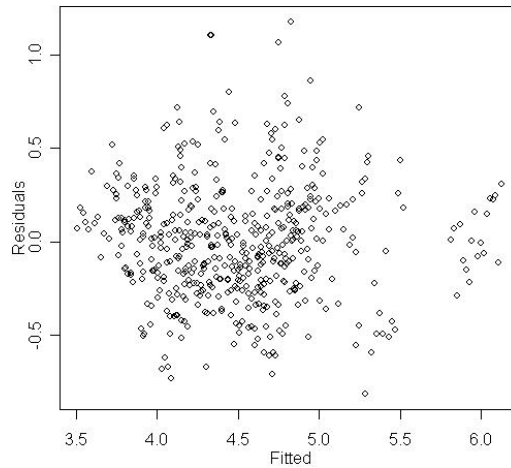


Figure 22: Adjusted residuals plot, we changed the regression model, the depended variable is now $\ln(Y)$, the independent variables are the same as before. The residual plot now shows that there is a constant variance.

$$Y = \beta_0 e^{\beta_1 X} + \epsilon \quad (7.18)$$

Where Y is the depended variable, X is the independent variable, the password typing number in our experiment. β_0 is the start point when X is zero, β_1 is the growth parameter. ϵ is the random element, and is due to variation or measurement errors.

To find the exponential regression we simply transform the depended variable Y to $\ln(Y)$, exactly as we have done in the transformation. Although the regression curve is exponential, the regression analysis performed is linear. This means that the linear regression will be correct, and the condition is good. The F-statistics shows that the regression curve has a good fit to the observed values, the assumption of an exponential model is sound.

The regression analysis of the transformed model can be seen in Table 13. Since Y is transformed to $\ln(Y)$, the growth parameter describe the change in percentage instead of absolute change. This means that the distance score is reduced by 1.8% every time an attacker tries to enter the password. The program, victim, attacker and password are either zero or one. Program three reduce the distance score by 9.7% from program one. For growth parameters bigger than 0.2 we need to use a formula, example is for password two. If the victim is using password two the distance score is reduced by $100 * (e^{-0.260} - 1) = 22.9\%$, see [52]. In this analysis it is highly probable that program three reduce the distance score, compared with program one. The regression analysis shows that it is 99.5% certain that program three will reduce the distance score. If we remove the attacker with very high values there is a larger difference between program three and one, even program two is outside a 95% confidence interval. If we remove the five first typings, it is 99.9% certain that program three will reduce the distance score. It is actually less certain that the distance score will be reduced by the number of tries. With the original data in the regression table we see that victim, password and some attackers influence the distance more than the program.

When we are testing hypothesis there is always a risk of making the wrong conclu-

Table 13: Regression analysis.

	Estimate	Std. Error	t value	Pr(> t)
Intercept	4.722	0.063	75.34	< 2e-16
Number	-0.018	0.002	-7.64	1.17e-13
Program 2	-0.048	0.034	-1.39	0.165
Program 3	-0.097	0.034	-2.83	0.005
Victim 2	0.141	0.034	4.12	4.55e-05
Victim 3	0.518	0.034	15.09	< 2e-16
Attacker 2	0.037	0.066	0.56	0.573
Attacker 3	-0.249	0.063	-3.97	8.43e-05
Attacker 4	-0.366	0.062	-5.88	7.60e-09
Attacker 5	0.142	0.070	2.03	0.043
Attacker 6	0.174	0.067	2.61	0.009
Attacker 7	-0.113	0.064	-1.76	0.079
Attacker 8	-0.383	0.062	-6.19	1.30e-09
Attacker 9	0.953	0.065	14.68	< 2e-16
Password 2	-0.260	0.034	-7.57	1.90e-13
Password 3	-0.276	0.034	-8.02	7.76e-15

Regression analysis table of the transformed regression model. Where Y is transformed to $\ln(Y)$. This regression analysis shows that program three has a lower score than program one. Since Y is transformed to $\ln(Y)$ the growth parameter describe the percentage change in Y for every unit X is increased, see [52].

sion, we can falsely reject H_0 when H_0 is true, or we can falsely keep H_0 when it is wrong. The first is a type one fail, and the second is a type two fail. Both type fails are bad, and we can never be 100 % sure that we make the correct choice. In this example we can say that it is more than 99.5% certain that program three reduce the distance score compared to program one. We have checked that the analysis complies with the regression analysis conditions, and the regression analysis should therefore be valid. This means that we have to reject H_0 , and can conclude that it is possible to learn someone's typing characteristics. Our findings indicate that it is difficult to imitate someone's typing characteristics, since some of the attackers were unable to improve their imitation with the additional feedback, and the difference between program one and program three is relatively small.

8 Discussion

Most of the discussion is already done in the previous chapters, but we have summarised and discussed further some important aspect of the thesis.

8.1 Authentication experiment

8.1.1 Password

We had one password for all 21 participants in the authentication experiment, the password was randomly generated with a password generator, see [39]. We choose the password “*p5uxAc6emm*”, since it utilise most of the keyboard partitions and a ten characters long password was the best trade-off between classification and usability, see Section 3.1.

Almost all our participants reported that the password was too long and difficult to remember. The authors of [34] also discovered that random passwords are hard to remember, they suggested using pass phrases instead. Pass phrases are created by the users, our participants needed to learn the random generated password. Although the password was constantly displayed to the participants, our findings indicate that secure passwords generated by an authority can be difficult to remember. This means that users will write down the password, with all the extra security issues involved. There is nothing wrong with writing down the password, as long as the user adequately protects the paper; this is for obvious reasons a security hazard.

Since brute-force attacks are mitigated with keystroke dynamics, it is possible to use shorter memory friendly passwords. Shorter passwords are also more user friendly, and this is one of the reasons why we choose to have eight characters long passwords in the imitation experiment.

8.1.2 High EER

We got a relatively high Equal Error Rate (EER) in the authentication experiment, the best suited distance metric for the imitation experiment had an EER of 23%. This is higher than most if not all published reports on keystroke dynamics. There are several reasons why we got a high EER in comparison with the studied literature.

We had a tested text (password) of ten characters, this is short compared to other reports, like in [53] where the tested text was 683 characters long.

We did not have any adapting function in our experiment. An adapting function is a function than adapts the user template to changes in the users typing characteristics. An adapting function should improve our results, especially since the experiment was conducted over a relatively long period of time.

Our participants ran the experiment on their own computer at home or work, this means that the timing accuracy will be different between the participants. The timing accuracy should be much smaller than 1 ms for all participants, but this cannot be verified. A low timing accuracy produce less accurate decision results, according to [53, 18].

We had 12 enrolment samples in the experiment. A higher number of enrolment samples produced better results, but we needed more than 25 enrolment samples before we got a significant (3.5%) reduction of the EER.

It is possible to improve the distance metrics further, we tried to divide with the Root Mean Square (RMS) instead of the standard deviation in the Manhattan City Block distance metric in Section 7.1.4, see [44, 54]. This resulted in an EER of 20.5%, when we removed outliers 1.5 standard deviations above and 2.5 standard deviations below. The RMS is connected with the mean and standard deviation, $X_{rms}^2 = \bar{X}^2 + \sigma_x^2$. This means that the RMS will always be greater than or equal to the mean. With 25 enrolment samples, the EER was reduced to 19%. This means that it is possible to improve our results in the authentication experiment.

The main reason for this high EER is probably that the experiment was conducted in an uncontrollable environment, that is on the participants own computers. The participants could be distracted, a high CPU load can reduce the timing accuracy, the participants could use different keyboards if they ran the experiment on several computers, they could try to write in a “funny” way for their own amusement, and there is always the risk that other than the participants tried the program. We had informed our participants that it was important that only the correct person used the program, but we could not control this. Although there are risks involved in an uncontrolled experiment environment, we believe that the results will be more realistic if the experiment is run on the participant’s computer without any supervision. Since this better imitates a real keystroke dynamic verification system.

8.1.3 Misspellings

Although we received more than 2000 password typings from our participants, about 340 (16.9%) of these password typings where misspellings or aborted password typings. Misspellings could occur anywhere in the password, but the aborted passwords where normally aborted in the beginning.

Most of these 340 password typings could be classified as misspellings, since it seems that the participants knew when they made a spelling mistake and often aborted the password typing by pressing the “enter” button. One example is a participant entering “p”, “6”, “back” and “enter”. The participant discovered that he or she had written “6” instead of “5”, and then tried to erase the mistake with “back space”. We did not deal with backspace in our experiment, and the participants knew that the program would return “password fail” instead of a score when they used backspace. They quickly learned that instead of fixing the misspelling, they could press “enter” or “login” and start from scratch.

If we had handled backspace as described in Section 2.6.3, the percentage of misspelled password would most likely be much lower. A real system therefore needs to handle backspaces.

8.1.4 Timing accuracy

We know from previous reports on keystroke dynamics that the timing accuracy is important. The authors in [29] state that the timing accuracy should be better than one ms. This was also the reason why we did not use a Java applet to capture the keystroke typings. Java applets have a timing accuracy of 10 ms or worse in some systems, like Microsoft. Our program had a better timing accuracy than 1 ms, but since it run on different computers there could be different timing accuracies. It could be as low as one μ s, but it would be worse on a slow system with many processes running. We had no way of knowing what the real timing accuracy was on the participants systems. Variation in

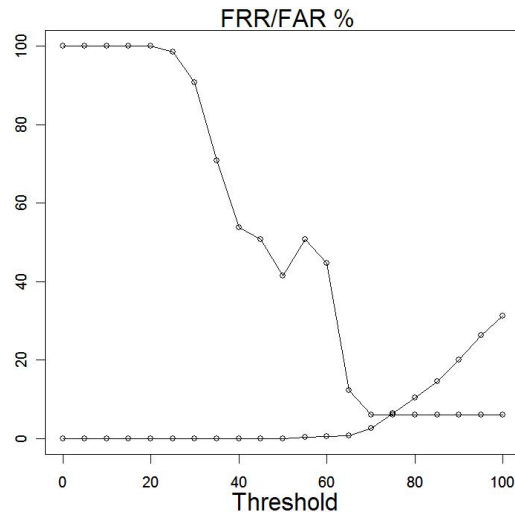


Figure 23: FAR and FRR with an adapting function for one participant in the authentication experiment. The EER is 6% and the graph looks strange due to outlier removal.

timing accuracy could however contribute to our relatively high EER, since some reference templates could be generated with a low timing accuracy. However, we believe that timing accuracy in our experiment was good enough and that timing accuracy had little to do with our high EER.

8.1.5 Adapting function

The studied literature suggest that an adapting function should be implemented in an authentication system based on keystroke dynamics, see [22, 18]. This is needed to adapt the user's reference profile to his or her changes in typing characteristics, since the typing characteristics changes over time. We did not need an adapting function in the imitation experiment and our focus was therefore on distance metrics without an adapting function. The authors in [18] reduced their EER by 50% when they used an adapting function. Our results in the authentication experiment should also improve, but we do not know to which extent. We tested an adapting function for one of the participants in the authentication experiment, we used the Manhattan City Block distance divided by the standard deviation, see Section 7.1.4. We removed outliers that were 1.5 standard deviations above and 2.5 standard deviations below mean. The participant had an EER of 9% without the adapting function, and the EER was reduced to 6% with the adapting function.

The tested adapting function replaces the oldest typing in the reference template each time the user's login attempt is accepted. The mean and standard deviations are recalculated each time a new login attempt is accepted. Figure 23 shows the FAR and FRR rates for several thresholds, the FRR graph is a bit strange for some thresholds, it is not logically that there is an lower FRR with a threshold of 50 than with a threshold of 55, the opposite is logic. This is probably caused by problems with outlier removal, where the standard deviation gets too small for some features and therefore makes the subsequent login attempts less likely to succeed.

The FAR rate is calculated with the new user template for each threshold. This means that the FRR is first calculated for a specific threshold, then the resulting template is

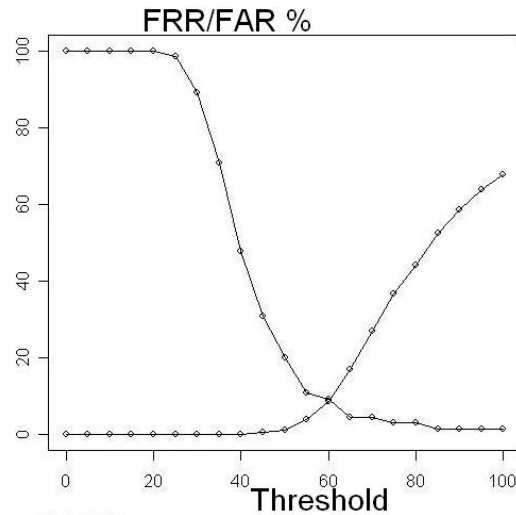


Figure 24: FAR and FRR without an adapting function for one participant in the authentication experiment. The EER is 9%.

used to calculate the FAR for the specific threshold. This means that the participant's last password typings will probably be the template in the higher thresholds. Since the last typings are probably always accepted with the high thresholds. This makes the FAR curve smooth, but the EER could be inaccurate and we can not conclude that an adapting function will improve the results from this test, when we tested the adapting function on another participant we actually got an higher EER with the adapting function than without. The difference was larger than 10%. A more accurate test would be to take the time when the passwords were typed into account, so that almost all the generated templates would have impostor attempts. Figure 24 shows the same participant as in Figure 23 without an adapting function.

What we can conclude with is that individual thresholds will reduce the EER, the participant in Figure 24 had an EER of 9% with an individual threshold. If he or she had used a global threshold, the FRR would have been around 20% for this participant, with the global threshold that gave the lowest EER with all participants. This is the same for all participants, there is normally a threshold that results in a lower EER than the global threshold. Choosing this threshold in a real authentication system is not as straight forward as in our experiment, since users have different passwords and we cannot calculate the FAR.

8.2 Imitation experiment

8.2.1 Passwords

We had a ten characters long password in the authentication experiment. The participants reported that the password was too long and hard to remember, as discussed in Section 8.1.1. We therefore decided to have shorter passwords in the imitation experiment. We choose to have eight characters long passwords. Passwords shorter than eight characters are considered insecure, see [34]. As in the authentication experiment, we wanted realistic passwords, and therefore created the passwords with a random password generator [39]. This replicates a normal password distribution situation, where

users receive a random password from their IT department.

The three passwords were:

1. *jamu9reS*
2. *bruf9Tr2*
3. *p7eneZuh*

The three passwords are of equal length, and have the same number of keystrokes. There are nine keystrokes in each password, since there is one uppercase letter in all three passwords. It was very important to have the same number of keystrokes, since we are comparing the distance returned from the distance metric. Nine keystrokes means that we can compare 17 features in the distance metric, nine duration timings and eight latency timings. Each compared feature in the distance metric will produce some distance, fewer features will therefore produce a smaller total distance. An important criterion is that the passwords must utilize most of the keyboard partitions, this is also one of the reasons why we decided to use these three passwords.

In the analysis, we discovered that there were some differences between the three passwords, password one was harder to imitate than password two and three. Password two and three were almost equally difficult to imitate. We do not know why password two and three were easier to imitate. It seems like the design criteria for a secure password is different from a secure keystroke dynamics password, future scientific work is needed to find the design criteria for secure keystroke dynamics password.

8.2.2 Distance metric

We tested several distance metrics in the authentication experiment, some performed acceptable and some had a very bad performance. The worst distance metrics tested would have been outperformed by a random decision system returning “*true or false*”. The best distance metrics was our own Overlap distance metric in Section 7.1.13, when we removed features with the large overlap. The Overlap distance metric was not suited in the imitation experiment, we could not calculate the overlap values between the participants and the distance metric can be seen as cheating, since it needs the legitimate and attackers typing before calculating the overlap. It is therefore not a usable distance metric with keystroke dynamics.

The Interval distance metric did also perform well, with an EER as low as 22% it was the second best distance metric, see Section 7.1.10. The Interval distance metric can be used in keystroke dynamic systems, but in our authentication experiment we discovered that the returned distance was quite coarse and an increase or decrease in the threshold had a great impact on the FAR and FRR values. This is probably not an issue in a real authentication system based on keystroke dynamics. But in our imitation experiment, we needed a distance metric that had a high resolution scale, since we were comparing distances from attackers and needed to distinguish the attacker's performance. The attackers would generally have a distance close to the maximum distance, and that would make it difficult to analyse which program the attackers performed best with.

We decided to use the Manhattan City Block distance divided by the standard deviation, since it performed acceptable and it has a very high resolution scale. We should therefore be able to discover differences between attackers, programs, victims, and passwords. During analysis, we discovered that the chosen distance metric was accurate

enough for our needs. Since we did not use the Interval distance metric we do not know if this can be used in a larger experiment. We believe that it is possible to modify the Interval distance metric to be suitable, perhaps by adding more scoring values than zero or one. Like increasing the counter with one if the feature is outside a small accepted interval, increase with two if it is outside a larger accepted interval and increase with three if it outside a large accepted interval. This would add more accuracy to the Interval distance metric and it would probably be accurate enough. However, as we discovered, the Manhattan City Block distance divided by standard deviation is good enough for a larger experiment.

8.2.3 Victim test order

In the imitation experiment, the attackers tried to imitate the three different victims in the same order. This gave the possibility that attackers would get better at imitating others each time. The assumption was that attackers would improve with each imitation attempt. If this was the case, we could risk getting the wrong results and we would need to take precautions in the analysis. We believed that it was possible to check if the victims order matters, by comparing attackers' learning curves against the different victims and statistical analysis. If the order does matter, the attackers learning curves should be steeper against victim two and three, and the regression analysis should indicate that victim three is easier to imitate than victim one.

We discovered that attackers learning curves were steeper against victim one than against victim three. The average learning parameter against victim one and two was 86%, the learning parameter was 92% against victim three. With the five first typings removed, the learning parameter for victim one, two and three was 96%, 93% and 97%. Attackers had a steeper learning curve against victim one and two than against victim three. The regression analysis shows that it was more difficult for the attackers to imitate victim three than victim one. These tests indicate that the order of the victims is unimportant. This is difficult to prove with only three victims, a bigger experiment is needed to verify this. However, it seems to be the case in our experiment. These findings could also mean that one victim is enough in further experiment, since the attackers seems unable to improve their learning process in this short experiment. Different passwords are still needed, the learning curve will probably not be disrupted if the same password is used.

9 Further work

We tested many distance metrics in this thesis, and various outlier removal techniques for each distance metric. Some of the tested distance metrics had good performance in the studied literature, but poor performance in our experiment. Although, some had good performance in both the studied literature and in our experiment, like the interval metric. We created the overlap distance metric, which was among the best distance metrics, but it requires all users to have the same password in order to calculate the overlap values. The Manhattan City Block distance metric with standard deviation was also one of the best, and it is easy to implement in a real system. We later discovered that it could be better to use RMS instead of standard deviation in this distance metric. It is obvious that future experiments are needed to find better distance metrics.

Intrusion Detection Systems (IDS) can be evaluated against tests datasets, like the Darpa dataset [55]. There are no similar datasets in keystroke dynamics. Scientist must create their own test data when they want to evaluate a new distance metric. It is therefore almost impossible to compare distance metrics based on their published results. It is important to have a large dataset of keystroke typings, which can be used to evaluate different keystroke dynamic verification systems. A large experiment with many participants that is conducted over a long period of time is needed to create the dataset. With data for both static and dynamic authentication. This dataset means that scientist can evaluate their distance metrics against other published distance metrics. In addition, it enables customers to compare various systems before they invest.

Adapting functions adjust the template when the users typing changes and is needed in a real authentication system, since users rarely change their password and their typing characteristics will change. We tested an adapting function in our experiment, but we discovered that it does not guarantee better results. We also discovered that it was difficult to get the correct EER. The template is important for the FAR, and the FAR is calculated against the same template for the higher threshold values. Since most of the typings will be accepted with a high threshold, and the template will therefore be generated on the last password typings. A better solution would be to use the time when the password was typed, so almost all generated templates have impostor attempts. An experiment is needed to test different adapting functions, and find the correct way of testing systems with adapting functions. The test data set should include time in order to test adapting functions.

We discovered that it is possible to learn someone's typing characteristics, but a large experiment with more attackers and many imitation attempts is needed to confirm our findings. This large experiment should also discover how easy it is to perform the proposed attack. With a large number of attackers, it is possible to have a control group that only use a program with little feedback, similar to program one. And a group that use a program with full feedback, like program three. With many participants the typing characteristics and imitation skills should be equally distributed between the two groups. This should of course be tested in the experiment, possibly after the attacks have been conducted. It should also be investigated to see if an attacker that manages to imitate

someone, is able to imitate other sentences from the same victim. Since this is an attack scenario, where a user changes his or her password before an attacker learned the old password typing characteristics.

10 Conclusion

In the authentication experiment, we got worse results than the studied literature with the same distance metrics. Some of the distance metrics that had shown a good performance before, had a poor performance in our experiment. For example the normalized minimum distance metric in [20], which had an EER lower than 5% in their experiment and an EER of 40% in our experiment.

We discovered that the Manhattan City Block distance divided by standard deviation performs well in a keystroke dynamics system. We also discovered that the interval distance metric is well suited in an authentication system, but it was too coarse for our imitation experiment. An authentication system decides whether or not it is the correct user, we tested the differences between attackers and that requires a finer scale. We did propose an adjustment of the interval distance metric to get a finer scale, this can probably be used in further experiments.

Long secure random generated passwords are user unfriendly, users will write such passwords down in order to remember. With keystroke dynamics it is possible to use shorter and more user friendly passwords. Since the password security will be increased with keystroke dynamics and brute-force attacks will be mitigated. Brute-force attacks will be mitigated since it would take several seconds to test one password, the same password has to be tested several times, since it could be the correct password and wrong typing characteristics.

In this thesis we wanted to answer the question “*can attackers learn someone's typing characteristics?*”. By giving attackers more feedback on where their typing was different from the victim, they managed to adjust their typing to be closer to the victims typing. Attackers can therefore learn someone's typing characteristics, but it is not alarmingly easy for an attacker to do so. In fact, some of the attackers were unable to improve their imitation with more feedback. Our results indicate that not only the password must be stored securely, also the typing characteristics must be protected with equally strong defences. Since an attacker with the proper tool and determination can learn legitimate users typing characteristics.

We cannot conclude that keystroke dynamics is insecure, the fact that this attack was relatively difficult for our attackers can indicate that keystroke dynamics is a very secure authentication method when combined with a password. Larger experiments are needed to test how resistant keystroke dynamics is against the proposed attack

Bibliography

- [1] Ilonen, J. 2003. Keystroke dynamics. In *Advanced Topics in Information Processing - Lecture*.
- [2] Lassmann, G. 2002. Some results on robustness, security and usability of biometric systems. *IEEE International conference on multimedia and expo*, 2(1), 577–579.
- [3] Galbally-Herrero, J., Fierrez-Aguilar, J., Rodriguez-Gonzalez, J. D., Alonso-Fernandez, F., Ortega-Garcia, J., & Tapaidor, M. 2006. On the vulnerability of fingerprint verification systems to fake fingerprints attacks. *IEEE International Carrihan Conference on Security Technology*, 130–136.
- [4] Gunetti, D. & Picardi, C. 2005. Keystroke analysis of free text. *ACM Transactions on Information and System Security*, 8(3), 312–347.
- [5] Cardot, H., Hocquet, S., & Ramel, J.-Y. 2005. Fusion of methods for keystroke dynamic authentication. *Fourth IEEE Workshop on Automatic Identification Advanced Technologies*.
- [6] Boechat, G. C., Ferreira, J. C., & Carvalho, E. C. B. 2006. Using the keystrokes dynamic for systems of personal security. *Transactions on engineering, computing and technology. Enformatika*, 18(1), 200–205.
- [7] Haider, S., Abbas, A., & Zaidi, A. K. 2000. A multi-technique approach for user identification through keystroke dynamics. *IEEE International Conference on Systems, Man, and Cybernetics*, 1336–1341.
- [8] Joyce, R. & Gupta, G. 1990. Identity authentication based on keystroke latencies. *Commun. ACM*, 33(2), 168–176.
- [9] Ren, J. 2006. Multi-order standard deviation based distance metrics and its application in handwritten chinese character recognition. *IEEE International Conference on Pattern Recognition, ICPR2006*, 1–13.
- [10] Bours, P. A. H. 2006. Authentication course, imt 4721. *Gjovik University College*.
- [11] Lau, E., Liu, X., Xiau, C., & Yu, X. 2004. Enhanced user authentication through keystroke biometrics. *MIT, 6.857: Computer and Network Security*.
- [12] Bleha, S. A. & Obaidat, M. S. 1991. Dimensionality reduction and feature extraction applications in identifying computer users. *IEEE Transactions on Systems, Man and Cybernetics*, 21(2), 452–456.
- [13] Peacock, A., Ke, X., & Wilkerson, M. 2004. Typing patterns: A key to user identification. *IEEE Security and Privacy*, 2(5), 40–47.
- [14] BioPassword Inc. <http://www.biopassword.com>, last visited 29.6.2007.

- [15] Deepnet Security. <http://www.deepnetsecurity.com>, last visited 29.6.2007.
- [16] Kotani, K. & Horii, K. 2005. Evaluation on a keystroke authentication system by keying force incorporated with temporal characteristics of keystroke dynamics. *International Journal on Man-Machine Studies*, 24(4), 289–302.
- [17] Eltahir, W. E., Salami, M. J. E., Ismail, A. F., & Lai, W. 2004. Dynamic keystroke analysis using ar model. *IEEE International Conference on Industrial Technology*, 3(1), 1555–1560.
- [18] Araujo, F. C. L., Sucupira, L. H. R., Lizarraga, M. G., Ling, L. L., & Yabu-Uti, J. B. R. 2005. User authentication through typing biometrics features. *IEEE Transactions on Signal Processing*, 53(2), 851–855.
- [19] Robinson, J., v.W. Liang, Chambers, J., & MacKenzie, C. 1998. Computer user verification using login string keystroke dynamics. *IEEE Transactions on Systems, Man and Cybernetics*, 28(2), 236–241.
- [20] Bleha, S., Slivinsky, C., & Hussien, B. 1990. Computer-access security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12), 1217–1222.
- [21] Song, D. X., Wagner, D., & Tian, X. 2001. Timing analysis of keystrokes and timing attacks. *10th USENIX Security Symposium, Washington*, 1(1), 337–352.
- [22] Monroe, F., Reiter, M. K., & Wetzel, S. 1999. Password hardening based on keystroke dynamics. In *CCS '99: Proceedings of the 6th ACM conference on Computer and communications security*, 73–82, New York, NY, USA. ACM Press.
- [23] Obaidat, M. S. & Sadoun, B. 2005. Keystroke dynamics based authentication. <http://www.cse.msu.edu/~cse891/Sect601/textbook/10.pdf>, last visited 29.11.2006.
- [24] Magalhães, S. T., Revett, K., & Santos, H. M. D. 2005. Password secured sites - stepping forward with keystroke dynamics. *IEEE Proceedings of the International Conference on Next Generation Web Services Practices*.
- [25] Monroe, F. & Rubin, A. D. 2000. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems*, 16(4), 351–359.
- [26] Sheng, Y., Phoha, V. V., & Rovnyak, S. M. 2005. A parallel decision tree-based method for user authentication based on keystroke patterns. *IEEE Transactions on System, Man and Cybernetics*, 35(4), 826–833.
- [27] Wong, F., Supian, A., & Ismail, A. 2001. Enhanced user authentication through typing biometrics with artificial neural networks and k-nearest neighbor algorithm. *IEEE Conference on Signals, Systems and Computers, 2001.*, 2, 911–915.
- [28] Monroe, F. & Rubin, A. 1997. Authentication via keystroke dynamics. In *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*, 48–56, New York, NY, USA. ACM Press.

- [29] Jin, L., Ke, X., Manuel, R., & Wilkerson, M. Keystroke dynamics: A software-based biometric solution. *6.857 Computer and Network Security Massachusetts Institute of Technology*.
- [30] Clarke, N. L. & Furnell, S. M. 2006. Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security*, 6(1), 1–14.
- [31] Fu, K., Sit, E., Smith, K., & Feamster, N. Aug 2001. Dos and don'ts of client authentication on the web. In *Proceedings of the 10th USENIX Security Symposium*.
- [32] Hocquet, S., Jean-Yves, Ramel, & Cardot, H. 2005. Fusion of methods for keystroke dynamic authentication. *Fourth IEEE Workshop on Automatic Identification Advanced Technologies*, 224–229.
- [33] Lin, D.-T. 1997. Computer-access authentication with neural network based keystroke identity verification. *International Conference on Neural Networks*, 1, 174–178.
- [34] Yan, J., Blackwell, A., Anderson, R., & Grant, A. 2004. Password memorability and security: Empirical results. *IEEE Security and Privacy*, 2(5), 25–31.
- [35] de Ru, W. G. & Eloff, J. H. P. 1997. Enhanced password authentication through fuzzy logic. *IEEE Expert: Intelligent Systems and Their Applications*, 12(6), 38–45.
- [36] Leedy, P. D. & Ormrod, J. E. 2005. *Practical Research, Planning and Design 8th edition*. Pearson Education, New Jersey, USA.
- [37] Norsk samfunnsvitenskapelig datatjeneste. <http://www.nsd.uib.no/personvern/index.cfm>, last visited 29.6.2007.
- [38] Lov om behandling av personopplysninger(LOV-2000-04-14-31). <http://www.lovdatab.no>, last visited 29.6.2007.
- [39] PC Tools Password Generator. <http://www.pctools.com/guides/password/>, last visited 29.6.2007.
- [40] Loerch, A. G. 1999. Incorporating learning curve costs in acquisition strategy optimization. *Naval Research Logistics*, 46(3), 255–271.
- [41] Nemet, G. F. 2006. Beyond the learning curve: factors influencing cost reductions in photovoltaics. *Energy Policy*, 34(17), 3218–3232.
- [42] The R Project for Statistical Computing. <http://www.r-project.org/>, last visited 29.6.2007.
- [43] Coltell, O., Badfa, J. M., & Torres, G. 1999. Biometric identification system based on keyboard filtering. *IEEE International Carnahan Conference on Security Technology*, 203–209.
- [44] Sturn, A. Clustser analysis for large scale gene expression studies. Master's thesis, Institute for Biomedical Engineering, Graz University of Technology, Austria, 2000.
- [45] Ibsen, H. *A doll's house*. Project Gutenberg.

- [46] Doyle, S. A. C. *The adventures of Sherlock Holmes*. Project Gutenberg.
- [47] Volcano Kit. <http://www.volcanokit.com/volcanokit3/ligcounter/index.php>, last visited 29.6.2007.
- [48] Vandana, J. P., Vijayalakshmi, A., Jaideep, V., & Nabil, A. R. 2005. Collusion set detection through outlier discovery. *IEEE International Conference on Intelligence and Security Informatics, ISI2005*, 1–13.
- [49] Draper, N. R. & Smith, H. 1998. *Applied Regression Analysis*. John Wiley & Sons, Inc., USA.
- [50] Løvås, G. G. 1999. *Statistikk - for universiteter og høyskoler*. Universitetsforlaget AS, Oslo, Norway.
- [51] Edwards, A. L. 1976. *An Introduction to Linear Regression and Correlation*. W. H. Freeman and Company, USA.
- [52] Thrane, C. 2003. *Regresjonsanalyse i praksis*. HøyskoleForlaget, Kristiansand, Norway.
- [53] Bergadano, F., Gunetti, D., & Picardi, C. 2002. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security*, 5(4), 367–397.
- [54] Woodard, D. L., Faltemier, T. C., Yan, P., Flynn, P. J., & Bowyer, K. W. 2006. A comparison of 3d biometric modalities. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, 57, Washington, DC, USA. IEEE Computer Society.
- [55] Lincoln Laboratory. <http://www.ll.mit.edu/>, last visited 29.6.2007.

A Table with letter probability

Table 14: Letter probability

Letter	Occurrences	Probability
e	64486	9.0%
t	46982	6.6%
a	42740	6.0%
o	42644	6.0%
n	35088	4.9%
h	33551	4.7%
s	32643	4.6%
i	32221	4.5%
r	30608	4.3%
d	22185	3.1%
l	21424	3.0%
u	16647	2.3%
m	13844	1.9%
w	12635	1.8%
c	11785	1.6%
y	11748	1.6%
f	10494	1.5%
g	9762	1.4%
p	7893	1.1%
b	7123	1.0%
v	5424	0.8%
I	4920	0.7%
k	4566	0.6%
H	1729	0.2%
T	1547	0.2%
N	1231	0.2%
M	1083	0.2%
A	1014	0.1%
W	1008	0.1%
S	917	0.1%
Y	750	0.1%
B	657	0.1%
x	646	0.1%
L	593	0.1%
q	493	0.1%
C	445	0.1%
O	423	0.1%
j	418	0.1%
R	417	0.1%
D	352	0.0%
E	336	0.0%
K	244	0.0%

P	230	0.0%
F	208	0.0%
G	207	0.0%
z	152	0.0%
J	126	0.0%
V	93	0.0%
0	82	0.0%
1	62	0.0%
U	54	0.0%
2	38	0.0%
8	37	0.0%
4	22	0.0%
Q	22	0.0%
7	18	0.0%
5	16	0.0%
3	15	0.0%
9	15	0.0%
6	14	0.0%
X	4	0.0%
Z	2	0.0%

The probabilities and occurrences of letters in the books “*A Doll’s House*” [45], and “*The Adventures Of Sherlock Holmes*” [46].

B Letter from NSD

Norsk samfunnsvitenskapelig datatjeneste AS
NORWEGIAN SOCIAL SCIENCE DATA SERVICES

Patric Bours
Institutt for informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 GJØVIK

Harald Hørlagres gate 29
N-5007 Bergen
Norway
Tel: +47-55 58 21 17
Fax: +47-55 58 56 50
nsd@nsd.uib.no
www.nsd.uib.no
Org nr: 985 321 884

NSD

Vår dato: 14.02.2007 Vår ref: 16077/KS Deres dato: Deres ref:

KVITTERING PÅ MELDING OM BEHANDLING AV PERSONOPPLYSNINGER

Vi viser til melding om behandling av personopplysninger, mottatt 09.01.2007. Meldingen gjelder prosjektet:

16077	<i>Keystroke Dynamics</i>
Behandlingsansvarlig	<i>Høgskolen i Gjøvik, ved institusjonens øverste leder</i>
Daglig ansvarlig	<i>Patric Bours</i>
Student	<i>Fred Erlend N. Rundhaug</i>

Personvernombudet har vurdert prosjektet og finner at behandlingen av personopplysninger er meldepliktig i henhold til personopplysningsloven § 31. Behandlingen tilfredsstiller kravene i personopplysningsloven.

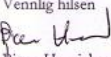
Personvernombudets vurdering forutsetter at prosjektet gjennomføres i tråd med opplysningene gitt i meldeskjemaet, korrespondanse med ombudet, eventuelle kommentarer samt personopplysningsloven/-helseregisterloven med forskrifter. Behandlingen av personopplysninger kan settes i gang.


Det gjøres oppmerksom på at det skal gis ny melding dersom behandlingen endres i forhold til de opplysninger som ligger til grunn for personvernombudets vurdering. Endringsmeldinger gis via et eget skjema, <http://www.nsd.uib.no/personvern/endingsskjema>. Det skal også gis melding etter tre år dersom prosjektet fortsatt pågår. Meldinger skal skje skriftlig til ombudet.

Personvernombudet har lagt ut opplysninger om prosjektet i en offentlig database, <http://www.nsd.uib.no/personvern/database/>

Personvernombudet vil ved prosjektets avslutning, 30.03.2007 rette en henvendelse angående status for behandlingen av personopplysninger.

Vennlig hilsen


Bjørn Henriksen


Katrine Utaaker Segadal

Kontaktperson: Katrine Utaaker Segadal tlf: 55 58 35 42

Vedlegg: Prosjektvurdering

✓ Kopi: Fred Erlend N. Rundhaug, Kalkveien 11, H0711, 2818 GJØVIK

Avdelingskontorer / District Offices:
OSLO: NSD, Universitetet i Oslo, Postboks 1055 Blindern, 0316 Oslo. Tel: +47-22 85 52 11. nsd@uio.no
TRONDHEIM: NSD, Norges teknisk-naturvitenskapelige universitet, 7491 Trondheim. Tel: +47-73 59 19 07. kyyre.svana@svt.ntnu.no
TROMSØ: NSD, SVF, Universitetet i Tromsø, 9037 Tromsø. Tel: +47-77 64 43 36. nsdmas@svt.uit.no

Figure 25: The letter from NSD [37], where they accepted our experiment.