Keystroke Dynamics on a Device with Touch Screen

Uno Andre Johansen

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2012

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

Keystroke Dynamics on a Device with Touch Screen

Uno Andre Johansen

2012/06/20

Abstract

Keystroke Dynamics has been heavily researched over many years. Despite the large activity there are few real world implementations using Keystroke Dynamics as an authentication mechanism. The change in how internet banks are accessed, from using personal computers to using smart phones, in combination with the increasing burden for people to remember many passwords, has increased the need for stronger or enhanced authentication mechanisms.

We look at the suitability of using keystroke dynamics as an additional feature to enhance security of authentication, when using a smart phone having a touch screen as input method. Both performance issues and security issues are investigated. We are going in depth on keystroke data analysis, where we look at why some methods perform better than others. The change in security, as a result of change in the physical/environmental factors are also addressed.

The results show that keystroke dynamics on a smart phone are more resilient against certain attacks than keystroke dynamics on a personal computer. We proved this by building a device capable of imitate someone's typing characteristics, and explained why the same task is difficult against keystroke dynamics on a smart phone. Further, we got good results from using new features available from a touch screen, and we also improved the performance of some detectors using a technique that adapts the standard deviation to skewed distributions.

Acknowledgements

I want to thank my girl friend for her support and understanding during my work on this thesis, which is not a small contribution considering the time used and my sometimes absent presence. I also want to thank Patrick Bours, my supervisor, for help getting participants, discussions and as a great source of knowledge and inspiration, I also want to thank Pål Erik Endrerud for pointing me to relevant stores to buy electronic components and for suggesting to use a Arduino board to control my imitator. And last but not least all participants for completing the experiments, without that effort there would not be any results.

Uno Andre Johansen, 20th June 2012

Contents

List of Figures

List of Tables

1 Introduction

1.1 Topic

A crucial factor in computer security is to only let in those we want to grant access and we also need to know the identity of every user for the access control system to work effectively. Username and password have been and still are the main method to gain access to computers or data systems. We give the username as a claim on who we are and present the password to prove that we really are who we claim to be. The process of verifying ones identity is known as authentication. The above described authentication is secure if and only if the password is secret and shared only between the one system and the one user. However, there are more methods of authentication and they are usually sorted in three categories. (1) something you know, like passwords, pin codes and pass phrases, (2) something you have e.g. device for generating one-time passwords, a cellular phone or a key-card,(3) something you are, which is properties of your body, either physiological features or behavioral characteristics. By combining factors from one or more groups we get multi-factor authentication which have a potential to be more difficult to break and is sometimes referred to as strong authentication. Authentication may also be grouped into continuous and static. Static is the authentication described above where one has to prove his identity before entering a system. In continuous authentication the identity is continuously verified after a user has entered the system and during use of the system.

Something you are, also called biometric, is one way of authenticating a user. Biometrics [1] are divided into two major groups, behavioral and physiological features. Physiological characteristics are measurements of various body parts e.g. fingerprint, palm print, shape of face, retina, iris, pattern of blood veins. Behavioral characteristics refer to how we use our body e.g. how we walk, how we speak, how we write our signature and the way we type on a keyboard. How we type on a keyboard is known as keystroke dynamics, which most most often use timing information to decide who is typing. By measure when a key is pressed and when it is released it is possible to detect pattern that can be used to recognize people.

This work is about using keystroke dynamics to enhance the security of the authentication mechanism when logging into your internet bank account from a mobile device. Specific, using keystroke dynamics only on the username which is the persons social security number.

1.2 Keywords

Computer Security, Authentication, Behavioral Biometrics, Static Authentication, Mobile, Touch screen.

1.3 Problem description

Passwords have for many years been the main technique to verify ones identity. If considering only one system, user ID and password are still a cost effective and sufficient method to authen-

ticate a user. But the world has evolved beyond the point that a user only accesses one system. Another trend is that more and more systems are bundled or accessed through a common infrastructure, the world wide web. Today more people than ever before depend on the internet to do many common tasks like, accessing ones bank account, communicating with each other using e.g e-mail or social networks, buying stuff from web shops and communicating with government e.g tax purposes or social services. It has great consequences if a criminal get unauthorized access, e.g. ones bank account could be cleared, identity could be stolen and misused in all sorts of crime. Apart from these personal hazards there is also a system perspective. Weak authentication systems become a vulnerability to strong security systems because they share users. This happens because people have cognitive limits, we are not able to remember many hard to guess passwords consisting of random characters. When we are supposed to maintain such passwords for several systems we tend to take shortcuts like writing down passwords, using memory aids or some system to make us remember. It is not difficult to see that low security systems now can be used to extract information that can be used to attack high security systems. More of this can be found in [2].

As we have seen increased dependency of online services make people take dangerous short cuts. At the same time the increased concentration of lucrative to exploit systems on the internet attracts criminals. Hence both the threat and vulnerability of authentication have increased, and there is a need for methods to reduce the risk for being exploited. Some internet banks which require username and password in their authentication process, has realized this increasing threat and do require the user to enter more information. Additional information may be a one-time password from a key-card or a one-time password received as an SMS message on the cell phone. If a criminal manages to steal a users password, he must have access to the users cell phone as well to gain access to his bank accounts. However, the development continues in the cell phone area too, nowadays cell phones are computers running e.g. linux(android) and the banks provide the users with mobil bank solutions. A cell phone is both easy to "borrow" and to steal. One possible attack could be to "borrow" the cell phone and plant a key-logger.

As we have seen the problem arise from three areas. First, the increased dependency of online services makes internet more attractive to criminals, who want to find vulnerabilities and exploit those to break systems and get money. Second, people need to remember more passwords. The password should be difficult to guess, something that make them worse to remember. Further passwords should not be used on different services. The solution is often to use same password at several locations or write down passwords on a piece of paper. Third, mobile computing (smart phone) makes us use the same device to authenticate and to be that "something we have". The problem is how we can mitigate these increased threats and vulnerabilities.

1.4 Justification, motivation and benefits

If a criminal or someone else who would like to see you suffer get hold of your password they could clear your bank account, embarrass you in your social networks, misuse your identity to criminal affairs and make a lot of trouble for you.

As a system owner or employer, you are vulnerable to how your employees manages their passwords. If they use e.g memory aids to generate passwords and these rules are guessed it has

an huge impact on the security of the systems these people have access to. Authentication is more secure when including more elements to it, on the other side more elements are disturbing for the users. Beside the cost of implementation and operation of more elements, increased complexity also leads to work at customer support. Keystroke dynamics as this extra element is cheap to implement because it do not need special hardware. Keystroke dynamics is also unobtrusive, and beyond the initial training period most people will not notice it. If keyboard dynamics has proper performance it can be used in many situations to improve the security of authentication.

1.5 Research questions

There has been a lot of research on static authentication using keystroke dynamics, and some are reporting promising results. Most research is done using an ordinary keyboard as input device. The trend is that more people are using smart phones or other touch screen devices to access their bank accounts and other on-line services. Crawford[3] recommend further research on new keystroke characteristics for mobile devices. e.g. finger pressure. At this point we are not aware of any research comparing keystroke dynamics on a standard keyboard with keystroke dynamics on touch screens. This is necessary to utilize existing research on this emerging computing platform. We also suspect that keystroke dynamics on a standard keyboard has vulnerabilities that are much less on a touch screen device. Before one implements a security mechanism in a new way or in a different area, there are two important questions that need to be answered. One obvious is, will the mechanism still work. The other question is easy to forget but are equally important, are there changes in threats and vulnerabilities. We refer to these two factors as suitability as a security mechanism.

The main research question; How is the suitability of keystroke dynamics on a touch screen compared to keystroke dynamics on a standard keyboard. To answer this question the following sub questions need to be answered:

1. How does the performance of keystroke dynamics compare, when using exact same features on both platforms?
2. How will new features(pressure, size, movement and position) from a touch screen affect the performance?
3. Is it easier to imitate someone's typing rhythm on one platform compared to the other?

1.6 Planned contributions

Beside answering the research questions there must be done research on detector methods. To be able to answer question 1, we must remove the distance measure itself as an confounding variable. Through that work we will go beyond comparison of methods and achieve knowledge of why some statistical anomaly detectors performs better that other. Thus the goal is to explain why we gets the results more than maximizing performance.

We also collect a data set, which will be available for later research.

We also provide a custom made device to imitate someone's typing pattern on a standard keyboard. This is done to support answers to question 3.

2 Methodology

How is the suitability of keystroke dynamics on a touch screen? Such big question is unrealistic to get an answer to in research limited to a few months, therefore the main research question is moderated with "*compared to keystroke dynamics on a standard keyboard*". Suitability is not a common unit of measure and need to be specified. One part of suitability is related to how well it works i.e., the performance. Another part of suitability is related to the security i.e., will the physical change in environment introduce or remove threats and vulnerabilities. Thus, we divide the question into three more manageable parts related to performance and security.

2.1 Performance

To measure performance we need data on both platforms i.e., the smart phone and the computer with standard keyboard. The data collected should be comparable so we need to control variables. Further, data should be collected in a realistic way to ensure that it contains a normal variation. e.g., to control how people write will certainly improve performance, but we will not measure what we intend to. Just think if one was told to write her signature holding the pen with only two fingertips. Then all do it the same way, but none will be able to write his genuine signature. Sometimes control is to make sure the situation is as normal as possible. The details on how data is collected is described under the experiment description in section 5.1.

Detector performance is influenced by many factors e.g., test subjects group composition, number of participants, learning, size of data sets and how the chosen detector fits the nature of the collected data. To get control of these factors we need to analyze the data carefully. The collected data alone is not enough to ensure that the data is representative for the population, and not a subset that give results that cannot be replicated i.e., poor validity. By using a external data set as reference we will detect if there are something wrong in our data sets. Both descriptive and inferential analysis of the data is conducted in chapter 6 Analysis.

The results from section 5.1 and chapter 6 are used in chapter 7 discussion, to answer the questions: How does the performance of keystroke dynamics compare when using the exact same features on both platforms and how will the new features from a touch screen affect the performance?

2.2 Security

The major threat against keystroke dynamics is imitation. Previous research claims that it is difficult to imitate someone's typing rhythm. We will show that it is easy to imitate someone's typing rhythm by building a device. How such device is built and used is described in section 5.2. We analyze data from the device in chapter 6 analysis. We also has focus on security in out literature study in section4.4. In chapter 7 we use results from analysis and literature study to answer the question: Is it easier to imitate someone's typing rhythm on one platform compared to the other?

3 Biometrics and Authentication

This chapter will give a brief introduction to biometrics and authentication, and readers already familiar with these subjects may skip to the next chapter on keystroke dynamics. However, some central terms used in this thesis are introduced here.

3.1 Biometrics

Biometrics is in this context about the human body and it's characteristics that make it possible to recognize individual persons. In a human to human relation we easily recognize people by their face, shape of body, their voice and even on the sound of their walk. Use of biometrics is also known in forensic investigation to identify people on the scene of a crime. In that setting use of e.g. a fingerprint, DNA and footprint are useful. When you go to the postoffice to pick up a delivery you have to sign a paper to get the package, if the receptionist does not know you he also may ask for identification, and by looking at the picture of you and your previous signature he verifies that you actually are who you claim to be. We can group biometric features into two groups [1]:

1. Physiological which is e.g. face, fingerprint, eye(iris and retina) and hand (shape or blood veins).
2. Behavioral which is e.g. signature, voice, gait and keystroke dynamics.

The focus of this report is keystroke dynamics which you can read more about in chapter 4, Keystroke Dynamics. A more detailed overview of biometrics can be found in [1, 4, 5].

Another aspect is how the biometrics are used. In the forensic investigation it is about identification and at the postoffice it was about verification of identity.

1. **Identification**. Identification is about recognizing a single individual among a group of people. To be successful in identification we need a database or register containing all relevant individuals. Then we decide on the identity by picking the one which is most similar to the features we are testing. If the database does not contain all relevant people, we should have a secondary measure to avoid using the least mismatching identity in the database.
2. **Authentication**. In authentication we have an claimed identity and the task is to verify the claimed identity. The prerequisite is that we must have access to a previous sample, that we trust, to compare against. In the above example the postoffice receptionist verified identity by looking at e.g. a driver licence to compare picture and signature, and accept it if close enough.

Human to human relation is easy to use to visualize the concepts. For the rest of this report we will only consider Human - machine relation, and how a user can authenticate against a machine.

3.2 Authentication

"*Authentication is the binding of an identity to a subject*"[6](p309). We know this principle from several areas. A subject may be a spy who carries a red rose, as a agreed token, to prove his identity for another spy. Products are commonly imprinted with brand name, products which are subjects to imitations have more difficult to fake imprints e.g. a hologram on software DVD's to prove authenticity. Web pages may prove its genuineness by providing a certificate from a third party. In a machine to machine communication e.g. between banks, it is essential that the machine knows it is communicating with the computer it is supposed to and not an impostor who wants to steal money. From the examples we can see there are two parties involved when doing authentication, one to prove his identity and the other to verify the others identity. In some situations where the authentication is vulnerable to attacks we mitigate the threats by doing the authentication in certain ways, also known as authentication protocols.

The scope of this thesis is the part of access control where a human proves his identity towards a computer system, or the computer system verifies the identity of the human. There is a distinction, usually the human present a proof of identity, but the computer system might as well detect proofs of identity without the users knowledge about it, as we will see later. The person that want access to the computer system is referred to as a claimant. When authenticating the computer system verifies the proofs against previous stored information and decides if the claimant in an impostor or a genuine user. The authentication between a human and a machine is divided into two groups, static and continuous. Static authentication takes place in before a session where an impostor should be denied access to the system. Continuous authentication takes place during a session and is intended to detect and reject intruders who gain access to computers after a genuine user has started a session and is not present/forgot to log out. Static and continuous authentication can be combined. It is also possible to do authentication at time intervals during a session, this is referred to as periodic authentication.

The proofs, also called authenticators, used to verify someone's identity are commonly grouped into four categories[7]. We combine the last two into one, since both are about biometrics - who we are. [6] describes an additional category *where the entity is*, e.g. in front of a specific terminal.

- Knowledge. Something we know. e.g. Password, PIN or pass-phrases.
- Objects. Something we possess. e.g. a token, code-sheet or a device to generate passwords.
- Who we are. Who we are is further dividen into two parts. Behavioral characteristics from how we use our body. e.g. walk, talk or type on a keyboard. And physiological characteristics. e.g. fingerprint, retina, iris and shape of face.

O'Gorman [8] compared the authenticators in the list above. The main issues are discussed in the next subsections.

3.2.1 Knowledge

Secret knowledge are security by obscurity. The user knows it, but adversaries do not. Knowledge is limited by our cognitive abilities. Humans forget, thus it is not possible to authenticate. Users often mitigate this by choosing easy to remember passwords, which is weak because they may be easy to guess to. Every time a password is used it is exposed and may be compromised. It is

difficult to detect if a password is compromised. Because of this we must assume that the security of a password is reduced every time it is used. If a user reuse the same password on several systems it lower the security in those systems. To authenticate by knowledge is inconvenient to users. When a password is compromised it is easy to change and the cost of implementing is low.

3.2.2 Objects

Objects e.g. a device for generating passwords are very difficult to misuse without access. Thus the defence is the same as with passwords, keep it close. The costs are higher than for a password system because each user needs a physical device. It is also less convenient for a user to bring a device all the time. However when lost, it is detected, and the user can take the corrective actions. The cost of replacement are higher than for passwords.

3.2.3 Physiological

Not all physiological features are unique in a population, or even present among all individuals. A finger, eye or even a hand may be lost due to accident, war or sickness. Some may also be difficult to persuade people to use because it is obtrusive e.g. retina scan. If a physiological authenticator is compromised it cannot be replaced. It is expensive to use because we need specialized hardware to detect the features.

3.2.4 Behavioral

The uniqueness is lower than for physiological characteristics. As for physiological features, not all individuals in a populations got all features. E.g. a mute person can not speak and if you have no hands it is difficult to type on a keyboard. Some features are not dependent of any special device, it may be transparent so users don't need to know it is used, and it is also generally cheaper to implement than physiological biometrics. This is certainly the case for both gait used as continuous authentication on a smart phone, and for keystroke dynamics. In either case you do not need additional hardware. The behavioral characteristics are easy to reveal, but hard to forge [9, 10]. Later in this thesis we will show a easy way to attack keystroke dynamics.

3.3 Biometric system for static authentication

The unobtrusive characteristics of behavioral biometrics is attractive in an authentication setting. In a world with increasing threats, and where complexity makes people take dangerous shortcuts, we need ways to strengthen security without additional burden on the users. Behavioral biometrics has potential to realize this goal. Figure 1 shows the main building blocks and phases of a biometric system for static authentication.

3.3.1 Phases

Before authentication can take place, the system need to learn how to recognize a user. This learning takes place in the **enrollment** phase, where the system collects reference signatures[1] to build a template. In behavioral biometrics it is normal to use multiple reference signatures to build such a template. Reference signatures should be collected in a controlled setting to be sure

[1] "*Signature*" is used as a generic expression for proof of identity that is used in one authentication try. This is not restricted to a written signature but can also be a scan of fingerprint or the keystrokes from typing the username. In some literature the word sample is used.

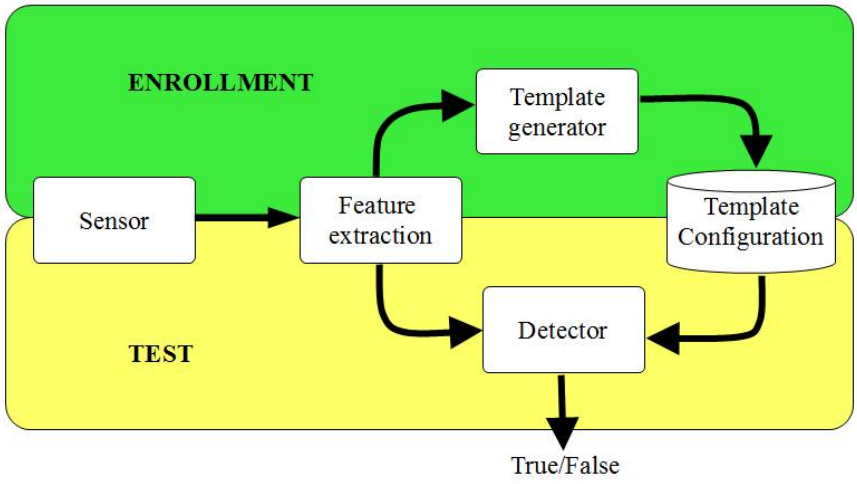

Figure 1: A block diagram showing the main components of biometric system for static authentication. The system is shown in two phases, the initial enrollment phase and in the test phase. The diagram is a simplified version from several sources, e.g. [11].

that it is the genuine user entering the data, and not an impostor. The template are stored in a database, linked to the user id.

The authentication takes place in the **test** phase, where the user provides his id, usually a username and provides his biometric signature. The detector retrieves the template based on the given id, and then compares the given signature against the template. If they are close enough, normally below a threshold, the user is accepted, if not the user are denied access. The threshold may reside inside the template if it is individual, or be a global setting.

The phases could be mixed. Because our behavior change can over time, the detector performance may benefit from adapting to this change, i.e. from updating the template with the new input.

3.3.2 Sensor

Sensors are used to capture biometric features. e.g. capturing a fingerprint can be done by taking a picture or generating a picture by capacitive measures, there could and should be built-in security measures to ensure the finger measured is alive. Voice is captured with a microphone, at given sampling intervals and the result are stored in a sound file. The timing of keystroke dynamics are available in most environments and need no additional sensors, while e.g. to measure pressure we sometimes need extra sensors.

3.3.3 Feature extraction

We need a stable and reliable way of comparing the template against new signatures. There are factors that introduce noise which makes this process difficult. Feature extraction is about extracting certain elements from the raw information that are usable to recognize people. A fingerprint is the pattern of ridges and valleys on the surface of the fingertip. Information of these features, e.g. relative position and angle, must be extracted from the image. This initial

phase of feature extraction, preprocessing, removes noise and other irrelevant information.

The next step is to select an adequate number of features to use in the system. In systems where we have a closed set of potential users, the feature selection can be optimized to distinguish between these users. In an open system, or systems where potential impostors are not known in advance, removing features is difficult. However, there may exist methods to select what features that should be used on a individual users basis.

3.3.4 Template builder

Templates can be generated from several reference signatures. An increasing number of reference signatures usually improves the quality of the template, while at the same time user acceptance decreases. Statistical measures like e.g. count, mean and standard deviation are often used in templates, pattern recognition structures like e.g. covariance matrixes or neural networks are also be used.

Both physical and behavioral features may change over time, due to aging or improved skills because of training. One can handle such change by implementing a template adaption mechanism.

3.3.5 Detector

The task of an detector is to compare a stored template against a user signature. The comparing can be done by calculating a score or distance between the template and the signature, and then use a threshold to decide to accept or reject the authentication try. Some special cases of authentication exists, where the group of user are controlled by external factors. Then a classification scheme can be used. Then the signatures are compared against all available templates and the id from the most similar template are compared against the login-try id. If it is equal the user are accepted as the valid user.

3.3.6 Data storage

The system needs a way to securely store templates and other settings, like thresholds. If an impostor has knowledge about the template it is much simpler to exploit the system, thus the data storage should be protected in a similar manner as a password database.

3.3.7 Performance

An authentication system can produce four results where two of them are errors.

1. An impostor tries to authenticate and is denied. This is fine.
2. A genuine user tries to authenticate and is accepted. This is fine too.
3. An impostor tries to authenticate and are accepted. This is in most cases the most serious error and is quantified by the FMR(false match rate). Let N_I be total authentication tries by an impostor and E_I be the number of accepted logins by an impostor. Then $FMR = E_I/N_I$.
4. A genuine user tries to authenticate and is denied access. This error is quantified by the FNMR(false non match rate). Let N_I be total authentication tries by a genuine user and E_I be the number of rejected logins by a genuine user. Then $FNMR = E_I/N_I$.

Ideally we should remove both FMR and FNMR. Gaines et.al. [12] claims that it is impossible to remove both FMR and FNMR, because we cannot reduce one without increasing the other. However, this is only true if we assume that it is not possible to separate users completely. And we will also see from empirical studies that it is possible to reduce both. See figure 2 for a visualization of the relation between FMR and FNMR.

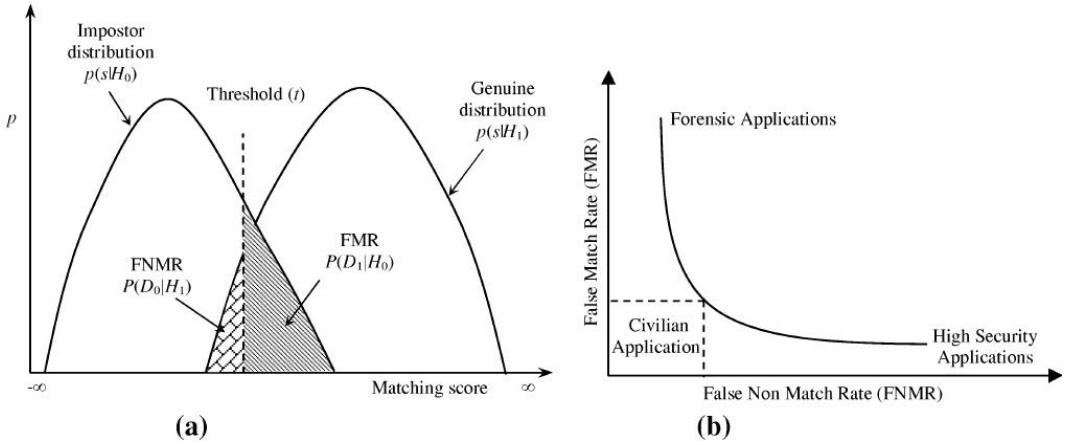

Figure 2: From [4],where (a)show the error rates at a given threshold over the distribution of genuine users and impostors. (b)show the relation between FMR and FNMR at different thresholds. The latter is known as a detection error trade-off(DET) curve. DET curves are sometimes confused with receiver operation characteristic(ROC)curve where the FNMR axis is replaced with the genuine match rate(GMR=1-FNMR).

FMR and FNMR are errors on algorithm level, on system level we have the "equivalents": false acceptance rate(FAR) where system accepts an impostor and false rejection rate(FRR) where system rejects an genuine user. To explain the relation between FAR/FMR and FNMR/FRR, we need two more error rates related to acquisition. Failure to capture(FTC) is when system cannot capture biometrics. Failure to enroll (FTE) is a FTC during enrollment. Because a higher FTC result in a better template and signature it result in a lower FMR and FNMR. We can also write the relation like $\mathrm{FRR} = (1-\mathrm{FTC})\mathrm{FNMR}+\mathrm{FTC}$ and $\mathrm{FAR} = (1-\mathrm{FTC})\mathrm{FMR}$. FRR and FAR are often confused with FNMR and FMR, but in some experiment the FTC is zero and the distinction is irrelevant.

One performance measure using FNMR and FMR are the equal error rate(EER). To find EER the threshold is adjusted to a value where FNMR=FMR, then EER = FNMR = FMR.

4 Keystroke Dynamics

A person can be recognized on the way he is typing on a keyboard [12, 7, 13]. The features and methods used are known as keystroke dynamics and is a subfield of behavioral biometrics. Biometric features are collected from a keyboard e.g., a number pad on a ATM machine, a virtual keyboard on a touch screen, a keyboard on a cell phone, the old fashion keyboard attached to a desktop computer or maybe in the future a virtual hologram keyboard projected in front of the user.

The next sections provide an introduction to keystroke dynamics, previous research and findings that are particular relevant to the research questions in this thesis.

4.1 Introduction

Keystroke dynamics is about collecting typing features and to analyze the features so that the result are useful for the intended purpose. After an overview of the features, analyzing methods and applications, we look into the history, recent research and results relevant to the research questions.

4.1.1 Features

The most commonly used feature in keystroke dynamics is timing information. Timing information can be collected from a special timer or device, from a hook within the operating system or by event handlers which pick up when a key is pressed and/or released. A signature of length n gives us a set of consecutive keystrokes(K), $K = \{k_1, k_2, ..., k_n\}$, a corresponding set of time stamps when keys are pressed,$D = \{d_1, d_2, ..., d_n\}$ and a set of time stamps for when keys are released $U = \{u_1, u_2, ..., u_n\}$. Where $_n$ is the total number of keys in the sequence. From the raw data, four timing features can be extracted.

Duration(DU) The duration of time from a key is pressed to it is released $DU = \{du_1, ..., du_n\}$, where each value $du_i = u_i - d_i$.

Latency(UD) The duration of time from one key is released to the next key is pressed. If a key is pressed before the previous key is released this feature will have negative values. $UD = \{ud_1, ..., ud_{n-1}\}$, where each value$ud_i = d_{i+1} - u_i$

UU The latency from one key is released to the next key is released. This feature is a aggregate of the latency and the duration of the following key. $UU = \{uu_1, ..., uu_{n-1}\}$, where each value $uu_i = u_{i+1} - u_i$. This latency is referred to as UU or latency(UU), never as latency.

DD The latency between two consecutive keys pressed. This feature is the sum of the duration of the first key and the latency. $DD = \{dd_1, ..., dd_{n-1}\}$, where each value $dd_i = d_{i+1} - d_i$. This latency is referred to as DD or latency(DD), never as latency.

It is also possible to extract other timing information like e.g., time it takes to write a word, three letters(trigraph) or two letters(digraph). [12] used digraphs, explained as the time it takes to type two successive characters, where the values range from 75ms to several seconds when using professional typists. Digraph differ from the latencies explained above in that they are timing between specific pairs of characters successively typed, that may or may not occur in the text e.g., the digraph $Digraph_{oe} = e_{released} - o_{pressed}$. However, in [12] it is not clear what is measured. We assume it is $Digraph_{oe} = e_{pressed} - o_{pressed}$. The background for this assumption is that the distinction between duration and latency is first pointed out in [14], as we are aware of, further is it not very likely that early research as [12, 15, 16, 7, 13, 17] had access to both key-press and key-release without being specific about which latency were used.

Other features are finger pressure[18, 19, 20, 21], position on key, size of key surface touched by finger. All these features may change during the duration a key is pressed and be a source for another feature, finger movement. For every key pressed we then have a vector $k_i = \{P, S, L\}$, where $P(pressure) = \{p_1, .., p_n\}$ and $L(position) = \{X = \{x_1, ..., x_n\}, Y = \{y_1, ..., y_n\}\}$ and $S(size) = \{s_1, ..., s_n\}$. Until recent years specialized keyboards has been needed to use these features. Today these features are easily available on devices using a touch screen or touch pad as input.

4.1.2 Applications

Research on keystroke dynamics has most commonly been targeted against user authentication. In user authentication we want to verify that it is the genuine user that accesses the computer system. Such authentication can be done once when entering the system(i.e., static authentication), or during use of the system (i.e., continuous authentication).

Keystroke dynamics can also be used to decide the emotions of a user. Epp et al[22] shows that it is possible to accurately determine two levels of seven emotional states(confidence, hesitance, nervousness, relaxation, sadness, and tired). The emotions can be used as an additional feature to build context sensitive systems.

We focus only on static authentication in this thesis.

4.1.3 Methods

In any form of human authentication the computer system need to "know" the user, or possess knowledge on how to authenticate someone by other means. Thus, before keystroke dynamics can be effectively used, we must store away one or more user signatures or only selected properties from the signatures.

We refer to the stored signatures as a *template*. The process of collecting the signatures we refer to as *enrollment*. Enrollment can be done before the user is allowed to use the system, or gradually and transparent during use of the system. The template may change over time to adapt to the change in a users typing rhythm over time(i.e., learning). When a template is collected it can be used in future authentication attempts to compare against the new signature. Based on a score or a distance measure the user is either allowed or denied access to the system. The score or distance is calculated by a *detector*

Detectors are implemented using different techniques e.g., descriptive statistics, inferential statistics or neural networks. The different methods has different strengths and weaknesses. To

maximize a detectors performance we may need to limit the number or features or do preprocessing on the features. A *threshold* is set on system level or per user. The threshold is the limit we use to decide if an authentication attempt is from a genuine user or from an impostor.

4.2 History

The idea of recognizing someone on their typing rhythm is old. Already in the age of telegraphs someone was able to recognize who was transmitting from the speed, rhythm and maybe common errors that were made. In 1980 Gaines et al.[12] investigated to what extent such typing signatures was present when typing on a computer keyboard, and also if such data could be used as a basis for user authentication. They conducted an experiment using six professional secretaries. Each typist should write almost 1000 words and they were asked to repeat the task after four months. However, not all participants completed all words. They recorded the time it took to type a pair(digraph) of successively typed letters. The timing information were recorded in a 1 ms resolution. Timing distributions curves showed large tails, so they transformed the data and achieved a more normal-distribution look on the distributions. The transformation was done by removing outliers exceeding 500 milliseconds, and by log transforming the data. On the transformed data they used student t-tests to verify that typing patterns was consistent over time, and thus usable in an authentication setting. They further developed an authentication procedure by analyzing which features that best discriminated the typists. Digraphs which occurs ten times or more were included in the analysis. Because of this, one participant was excluded due to too few words, resulting in a total of 11 test samples and 55 unique authentication tests, were 50 are impostor attempts. When using all digraphs they got no primary errors and 2 of 5 secondary errors(40%). Further work with the digraphs, using only right hand digraphs gave a perfect result with no errors. They managed to achieve the same performance using only five digraphs, in, io, no, on and(ul, il or ly). The good result are inspiring for further work, but keep in mind that this is not a realistic authentication situation. They used a homogenous small group and had quite few samples. The sample size were big and not suited for authentication. The fact that the participants were skilled touch typists and a mix of right and left handed probably also contributed to the good result. It is not strange that by picking 5 of 87 feature manage to separate 6 users with 2 samples each. The good question are, how can we identify those few features in advance for another group of people? Another result is that the change in typing was small between sessions of four months.

In the following years two US. patents were filed. In 1986 Garcia[16] got an US. patent on a personal identification apparatus. Garcia use the average of the time delay between successive input operations to build a template for each user. Authentication trials are then statistically tested against the claimed user template by using Mahalanobis distance. When using two thresholds 50 and 100, where values below 50 give access to the system and values above 100 gives access, and values in between means that user has to type his name again, he achieve a average FMR$<$0.01% and a FNMR$<$50%. Garcia claim that using an individuals own name, perform training before making a template and remove outliers will improve performance. However, keep in mind that no information on experiment set up or method used is given. Another US. patent were approved in 1989. "Method and apparatus for verifying an individual's identity"[15] by Young and Hammon.

They introduced a vague concept of continuous authentication. Various features are combined into an n-dimensional vector. Such features could be time between successive keys pressed, time to write more that two words, key pressure or any combination of such. They suggest to use the Euclidean distance to compare two such vectors, or a Euclidean distance normalized in respect to variance and with individual weighted features. They provide no evidence or indication of performance of the invention, but still claims that typing patterns are as unique as a persons fingerprint.

After 1990 the amount of research have increased every year. Joyce and Gupta[7] give a good overview of research prior to 1990. Peacock[23] are summing up trends and challenges in 2004. In recent years Shanmugapriya and Padmavathi have conducted a survey[24] and Crawford[3] has written a review covering the research activity. The latest paper survey found and also the most comprehensive is by Karnan et al.[11] in 2010.

4.3 Recent research

We have not been able to track down specifications from systems that use keystroke dynamics in the authentication process. However we note that there exist a couple of systems that claim to be using keystroke dynamics as authentication method. However, there have been comprehensive research in the area which we will explore in the next sections.

4.3.1 General

In the secondary research the focus performance tend obscure other important finding. The performance are just numbers that are not comparable even with equal unit of measure. To be comparable several criteria must be met. The group of test subjects must be large and representative for the population. The same signature should be used and collected in a similar manner. The same amount of signatures gathered with equal time intervals, due to the learning curve. Compare of performance within the same research is a very useful measure to show that certain techniques are successful or not. Karnan et. al.[11] have a good review of methods used in feature extraction/selection and in detector implementations. In table 3 there is a overview of literature in keystroke dynamics on mobile devices. In this thesis we focus on findings and results found in the next sections.

4.3.2 Findings

Adaption mechanism Adaption mechanism is not a new idea. Already in 1990 [13] used that idea, when they weekly updated the template. An adaption mechanism were also used in [25]. Lee et. al.[26] Improved the average performance by using an adaption mechanism, but for some users it had negative effect. [27] found that an adaption mechanism increase the performance only when it is used with a conditional update where most of the features must be within a certain distance.

Features Loy et al.[18] compared latency and pressure, latency resulted in a EER 1.5% better than the pressure did. However, the combination improved EER by another 3.2%. That combination of different features outperforms single vectors is supported by [28, 29].

Individual parameters Hocquet et. al.[30] used information in the training data to calculate

Study	Type	Keyboard Style	Metrics			Classifier	Study Size	Error Rates (%)		
			Inter-Key	Hold Time	Other			FAR	FRR	EER
Clarke *et. al.*, 2002	Static	Numeric	✔			Neural network	16	1.3%	0%	–
Clarke & Furnell, 2007	Static	Thumb	✔	✔		Neural network	32	–	–	12.8%
Karatzouni & Clarke, 2007	Static, pseudo-dynamic	Numeric	✔	✔		Neural network	50	–	–	12.2%
Buchoux & Clarke, 2008	Static	Numeric	✔			Neural network, statistical	20	0%	2.5%	–
Saevanee & Bhattarakosol, 2009	Static	Soft	✔	✔	✔	Neural network	10	–	–	9%
Zahid *et. al.* , 2009	Static, dynamic	Numeric	✔	✔	✔	Neural network, statistical	25	2.07%	0%	–
Campisi *et. al.*, 2009	Static	Numeric	✔	✔		Statistical	30	–	–	14.46%

Figure 3: The figure show table II from Crawford[3], and list literature on mobile device keystroke dynamics.

both threshold and the weights on each method used in the fusion. They found that personalized parameters increased the performance of keystroke dynamics.

Multiple detectors Hocquet et al.[31] have experimental result showing that three different ways of combining classifiers have a better performance than each classifier alone. The classifiers they used are (1) average and standard deviation, (2) typing rhythm and (3)order of timing information, the last like in [32]. Even with a result EER=1.75% they got high maximum FAR and FRR. This indicate that the errors belong to few users, that have an unstable way of typing.

Outliers [12] removed outliers above 500 milliseconds(digraph) while [7] removed outliers exceeding three times the standard deviation. Mahar et al.[17] showed that there are no uniform variance across all digraph latencies for a single user. They also verified that a single threshold for outliers is not appropriate when differentiating between users.

Preprocessing Yu and Cho[33] improved performance from FRR=15.78% to 3.54% by using feature subset selection. They achieved this using the genetic algorithm(GA) in a wrapper configuration using one class support vector machine (1-SVM) for evaluation.

Montalvao Filho and Freire [34] achieved improved performance when using a single memoryless non-linear mapping of time intervals on the keystroke data. The work is based on the belief that timing information is log normal distributed while most algorithms assume normal distribution. The performance gain varied between different detectors.

Public data set Killourhy and Maxion made a keystroke benchmark database[35] public available and did a comparative study[36] on 14 novelty detectors presented in literature. Another database is made public on the internet by [34].

Signature length [13] Shows that misclassification increase steeply when signatures are around

10 and shorter. Ord et al.[37] considered 11 digits to be too much for a user to remember, thus impractical in real use.

Sound Nguyen et. al [38] use indirect detection of timing and pressure via sound recording. This support my imitator, another easy way to capture someone's typing characteristics. A bio-matrix with independent component analysis(ICA) were used to extract the features Key hold, latency and pressure. Fast artificial Neural Network Library(FANN) were used for classification. Considering that they used indirect measures the results are impressing, FAR 4.12% , FRR 5.55%.

Template Joyce and Gupta[7] found that eight signatures was sufficient to form the template. They used first name, last name, username and password to form a signature.

Threshold [7]Set individual threshold by calculating distance between template and each signature the template is based upon. Then calculate meanμ and the standard deviationσ of these distances. Threshold$= \mu + 3\dot{\sigma}$. A similar technique is used by [39].

Timing resolution Killourhy and Maxion [40] simulated various clock resolution, by using data collected at a 0.2millisecond resolution. The effect was small in the normal range [0.1ms, 20ms] of clock resolutions.

Training Bleha et al. [13] and [7] both uncover that familiar strings are best suited in keystroke dynamics. [27] also confirm a significant improvement when familiar signatures are used.

Unskilled typists Hwang et al.[41] propose a method to improve the data quality of timing data itself; uniqueness and consistency. They introduce artificial rhythms to improve uniqueness and tempo ques to improve consistency. Thus, this is a strategy a user can use himself to increase the effect of keystroke dynamics. By typing the signature according to music familiar for himself, stability of typing will increase, while by making n pauses at chosen location in the signature. The latter one is however easily exploited by shoulder surfing. In [42] they used this method on a mobile phone in an empirical experiment and reduced error from 13% to 4%. The method showed to be most effective for unskilled typists, which normally has been a challenge in keystroke dynamics.

4.4 Results related to research questions

The results from use of adoption mechanisms, the effect of training, and the effect of familiar signatures tells us that we need to check our data for learning effects. We also use a 11 digit number which may be difficult to remember, and we need to consider the effect on our results. However, in a real situation here in Norway, that is probably not an issue because the social security number is frequently used in authentication.

We use the public data set[35] as a reference data set, both for quality assurance of our methods and to verify properties of out data sets.

The possibility to use sound to capture keystroke features is used in our security discussion. In the next section more research on security displayed.

4.4.1 Security

A Smart phones is a full-fledged computing platform in respect to functionality, however and as usual the cost is security and privacy challenges. Cai and Chen[43] used the motion detection sensor on a smart phone as a side channel to build a touch logger(key logger). They could correctly infer more that 70% of keystrokes. This is a threat to keystroke dynamics too, because it is easier reveal the typing rhythm than it is to infer what key is pressed.

Schlegel et.al.[44] developed a stealthy and context-aware sound trojan for smart phones. It is capable of detecting situations where e.g. credit card information is spoken. This information can then be extracted locally on the phone and transmitted silently to adversaries. The trojan can be set up to start automatically when a phone call starts. Thus it is hard to discover.

Cai et.al. [45] illustrate the vulnerabilities of a smart phone with built in sensors and propose a framework for protection against exploitation of the sensors. The most alarming is that the operating system itself did not contain any mechanism to solve this problem.

Software attacks[46] against keystroke dynamics and defences [47] also exists, but is out of scope for this thesis. However, some make assumptions like in [48] "*We assume that a biometric is not reproduceable. Hence it is unique to an individual, but even more importantly, one should not be able to artificially generate a "device" with sufficient characteristics to pass a biometric verification of a user.*". If we combine this with claims that shoulder surfing is not a threat for authentication systems using keystroke dynamics [24], then it become dangerous. If shoulder surfing using a audio recorder or a video camera the typing rhythm is easily revealed. As Schneier[49] say biometrics are not secrets. Vuagnoux and Pasini[50] prove this too when they show that it is possible to recover 95% of keystrokes at a range of 20 meters, even through a wall. They tested 12 different keyboards and they were all vulnerable to at least one of the four attacks they describes.

Rundhaug [10] investigated a humans ability to learn someone else's typing pattern. It is not easy for an attacker to imitate someone's typing characteristics. Hence they still recommend that template should be protected with equally strength as passwords.

Already in 2005 Clarke [51] found that mobile devices were widely used to access on-line services. At the same time the user awareness were low and most users did not care to enable available security mechanisms. Many users (83%) think use of biometric is a good idea and voice verification were the most preferred by users. A year later Kowalski and Goldstein [52] confirm the same low awareness of security functionality on mobile. The user acceptance for use of biometrics to increase security are high, but we must not forget that biometrics are not secrets[49] and if they are lost they can not be replaced.

5 Experiment Description

Recall the main research question. Is keystroke dynamics using a touch screen as input feasible. Others have already found that keystroke dynamics are feasible using a standard keyboard as input, however not as a stand alone authentication method. Thus, we only need to (1)compare how keystroke dynamics performs using a touch screen as input compared to performance using a standard keyboard as input, and (2)compare intrinsic vulnerabilities and strengths between the two platforms.

To answer the first question we conduct a *Data Collection Experiment*. Data from the experiment is used in our *Analysis* where we find suitable detectors, decide on methods to use and present our results.

To answer the second question we conduct an *Imitator Experiment*. The results from this experiment is used in a *Security Discussion*.

5.1 Data Collection Experiment

We want to simulate the situation where a person access an online service using ones social security number as user identification, e.g. ones bank account. This may happen at any time a person wish and has access to a computer. Most people have access to a computer at home or at work, while the smart phone is brought all the time. Thus, the experiment must have the same availability. The availability is solved by developing two data registration programs, one for Microsoft Windows OS and one for Android OS. The information about the experiment and programs were distributed from a web page dedicated for this experiment. In addition a system for collecting and preparing data for analysis are developed.

5.1.1 Execution details

In a normal situation one types ones user id only once when logging into e.g a online banking service. Such sessions does not occur many times a day. To simulate such activity it will take too long and the participants would probably be dropping off. On the other side, collecting many samples over short time would be more acceptable to the participants and probably yield much better performance due to lover variance in the typing rhythm. The performance would not be real, because they are based on data more stable than in a normal situation.

A balanced solution were chosen. Data are collected over 20 session, in each session 3 signatures are given. In total 60 signatures are collected from each participant. The text to type are available in advance of each try, and must be remembered. When start typing the text disappear. If the signature have typos, it is rejected and, it must be retyped. By using this approach the natural learning curve are preserved, giving natural variance in data. The data are influenced by a bigger register emotional states and possible interruptions.

The text typed is supposed to simulate ones social security number. Using a persons social security number in such experiment would be too intrusive and potential harmful. Instead all

users are given the same fictive social security number. The constitution day of Norway, 17th of may 1814, are used for the date part. A randomly selected number,02293 , are used for the last digits. The signature is then 17051402293. Selecting the same number for all participants have up sides and down sides. Each participant may be used both as impostor and as a legitimate user. However, the participants have the state of mind as a legitimate user all the time and they are not familiar with the number used as they normally are with their own social security number.

Next section describe the technical setup, and the measures taken to make the two environments as equal as possible.

5.1.2 Technical details

Three different modules are developed to handle collection of experiment data. One module is a program to register signatures from a smart phone, one do the same from a personal computer and the third is a module for collecting session data and assembling then into separate file for each participant.

Smart phone

The purpose of the smart phone program is to collect experiment data from a touch screen. The program is developed using Eclipse IDE for Java Developers, Indigo service release 1, with the Android SDK plug-in. The Android plug-in allows to develop software for several versions of the android platform. Our software utilize the Android 2.3.3 platform.

The program consist of three screens, the main menu form, the registration screen and the experiment screen. Initially the menu screen only contains a button for registration after registration it also contains one button for each registered user. In the registration screen we fill in a name, email address, phone number, gender, preferred hand, typing skills and age. By pressing the user-button on main menu screen the user are transferred to the experiment screen.

Virtual keyboards may be configurable and to avoid different setups among the participants we developed a custom keyboard and input field. This way it is not possible to change anything by configuring the smart phone itself. To avoid further differences between participants and between the PC experiment and the smart phone experiment we made the key layout, see figure 4, the same as the number pad section of a standard keyboard, we locked the screen in a portrait orientation, we implemented multi touch, and we changed the standard key event mechanism. Multi touch make it possible to press next key before releasing the previous. The normal event mechanism for a button is to abort action when finger leaves the button area, but still are in contact with the screen. To avoid unnecessary aborts we listen to event on the background instead and calculate what key is hit. We ignore when fingers hit outside any buttons, and start the key action upon hitting a key and ends the key action when the same finger leaves the screen surface again.

When starting a session a message is shown, see figure 4. The purpose of message is to repeat the explanation of the number to make it easier to remember. The user must press the ok button to start the session. Before typing start, also after one try and the next, the number is shown in the input field. When start typing, the number disappear. If a longer number or a wrong number is typed the input field show "ERROR" and the user must press the button CLR to retry. When three tries are correctly typed, the data collected are sent to server and the user are returned to

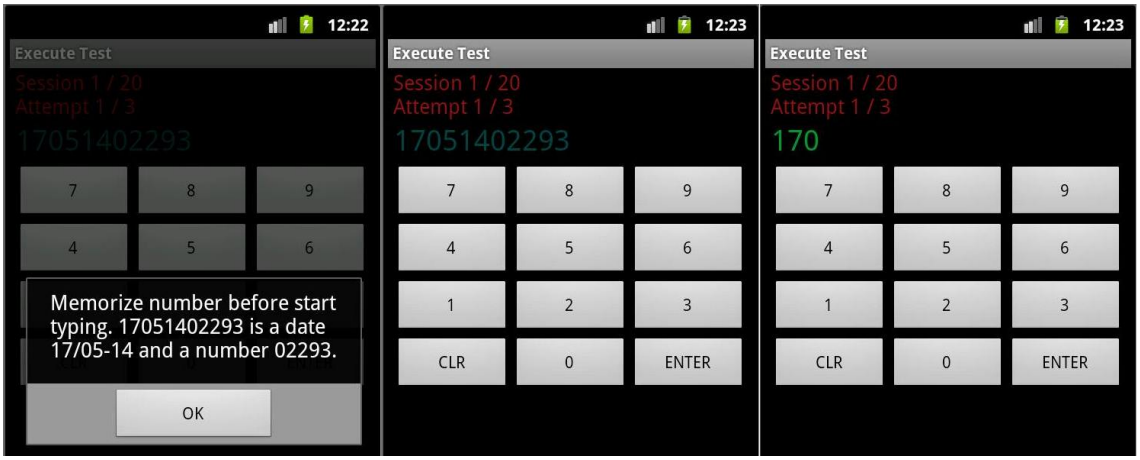

Figure 4: Experiment screens on a smart phone

the main menu screen.

From the main menu screen the user may choose to register as a participant or start start a new session when already registered. More than one user may register using the same phone. It is not possible to start a new session before the delay, 30 minutes, from the previous session has expired, see figure 5.

The setup is downloaded from the server upon registration. Setup include the information message, number to type, number of sessions and number of tries in each session. A user can not complete the experiment from multiple phones. Details about data collected is found in the appendices Experiment data.

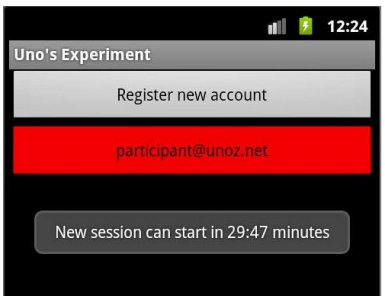

Figure 5: A session is done, and user has tried to start next session before delay has expired.

PC

The purpose of the PC program is to collect experiment data from the number pad section of a keyboard. The PC program is developed in Microsoft Visual Studio 2010 professional as a windows forms application using C# and Microsoft.NET Framework Version 4.

The PC program consist of three forms, the startup form, the registration form and the experiment form. The startup form have a link to the registration form, a link "*forgot password*" and login fields. In the registration form we fill in a name, email address, phone number, gender, preferred hand, typing skills, age and a password. When registered one can enter email address and select the link "*forgot password*". The password is then sent by email to the user. To login the user enter email address and the password, and we are transferred to the experiment form 6.

In the experiment form, an ordinary input field is used to enter data, see figure 6. However to have control over the environment there are limitations. Only key codes in the range {96, ...,105} and 13 are accepted. These key codes are from the number pad section of the keyboard. Other key codes will result in ERROR status and the user have to press clear button, or delete on keyboard to start restart typing. The number typed are evaluated when the ENTER key is released.

After each login, setup data are downloaded from server. Setup include the information message, number to type, number of sessions, sessions left and number of tries in each session. Thus, a user may user several computers to complete the test. Details about data collected is found in the appendices Experiment data.

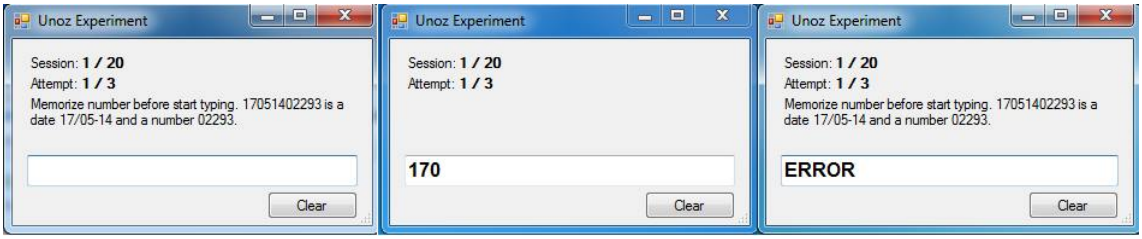

Figure 6: Experiment forms on a PC

Data collection

The goal of the data collection are to collect the individual session files, assemble them into one file per user containing all sessions. The content of these files is described in the appendices Experiment data. The data collection module consist of four server scripts written in PHP and a small windows program for downloading and assembling the session files into one file per user. The four files on the server are (1) register.php to register new users, (2) login.php to handle user logins, (3) ForgotPassword.php and DeliverData.php. The windows program is developed in C# and Microsoft .NET framework Version 4.

5.1.3 Participants

The volunteers are mainly students and employees at the school. A total of 42 persons downloaded program and registered in the experiment. 7 of these registered in both the pc and in the smart phone part of the experiment. 23 of the 46 registered persons had completed at least one

part of the experiment, 19 completed the pc experiment, 10 competed the smart phone experiment, and 6 completed both. All participants are right handed, except two in the pc experiment that are left handed. All completing participants are male. The youngest participant are 15 years old, the oldest are 46 years old, and average are 26($\mu = 26.43, \sigma = 8.57$).

All participants was given the same information before entering the experiment, and all information were available on a dedicated web-page during the experiment. Upon registration an automatically generated e-mail was sent to the participant containing a link to the experiment web-page. Information given are a brief overview on authentication, biometrics and keystroke dynamics, and details about the experiment and the motivation for doing it. No information or encouragement were given on training before entering the experiment nor any suggestion to maintain a stable rhythm of typing.

Some small prizes were announced to motivate people to join the experiment.

5.1.4 Ethical considerations

The volunteers were well informed before joining the experiment. The information include overview of topic, experiment and practical issues. Topic covers authentication, biometric and keystroke dynamics. Experiment information is both details about execution of the experiment, what data are collected, the goal with the research and how the data are used. Practical issues is about who are the data processing unit, contact information, that the participation is voluntary and the option to stop at any time and have all data deleted, and that the data are fully anonymous after collection phase and before assembly and use.

Active consent were given by downloading program after reading the agreement. The volunteers selected time and place for reading agreement and registering themselves, and without any external supervision or pressure.

In the experiment a 11 digit number was used to collect keystroke dynamics data. The number is not likely to be used by anyone outside the experiment. The data collected is only usable to authenticate someone using the specific number, and is useless to authenticate anyone in any other situation and far less useful to identify someone. The risk for misuse is therefore considered non existent.

5.2 Imitator Experiment

As seen, work is done to find out if a human can imitate ones typing characteristics[10]. They found that some learning is possible, but not a easy task. We know from several areas that machines are more accurate that people and want to find out how easy it is to build a device that are capable to imitate a humans typing characteristics.

The imitator experiment entails to construct a device, the ImitatorUno, capable to imitate a persons typing rhythm. The controller card used in our device is named Arduino Uno, thus we name our device ImitatorUno. This section describe the building process of the ImitatorUno, the software and and how it is used to deliver data to the data collection experiment. The results from this section is then used in the Security Discussion.

5.2.1 ImitatorUno

Coarse measures of an a keyboard was done. It takes a little less than 60 gram to press a key, and about 30 gram to keep it down after pressed. Further the key must be pressed about 2 millimeter. This work can easily be done with a push solenoid. The distance between center of one key to the center of the next key where measured to 1.9 centimeter, thus the diameter of the solenoid must be less than 1.9 centimeter. When an electrically generated field collapses a power surge is released. By applying power to any coil such field is generated, and the field collapses when power is removed. This is deliberately done in cars to make spark plugs spark. In the ImitatorOne we do not want this effect because it may harm other electronic components and may significantly reduce lifetime of switches used. To remove the voltage spike we add a diode in parallel with the solenoid. The micro relay used for switching power to the solenoid also contain a coil and need the same protection. The Arduino Uno[53] is a microcontroller board based on the ATmega328 and are well suited as a controller board for the ImitatorUno. The digital output are used to control power switching. To lower the load on the digital output a dalington transistor is used between the output and the micro relay. The diagram in figure ImitatorUno, diagram show the control of one solenoid.

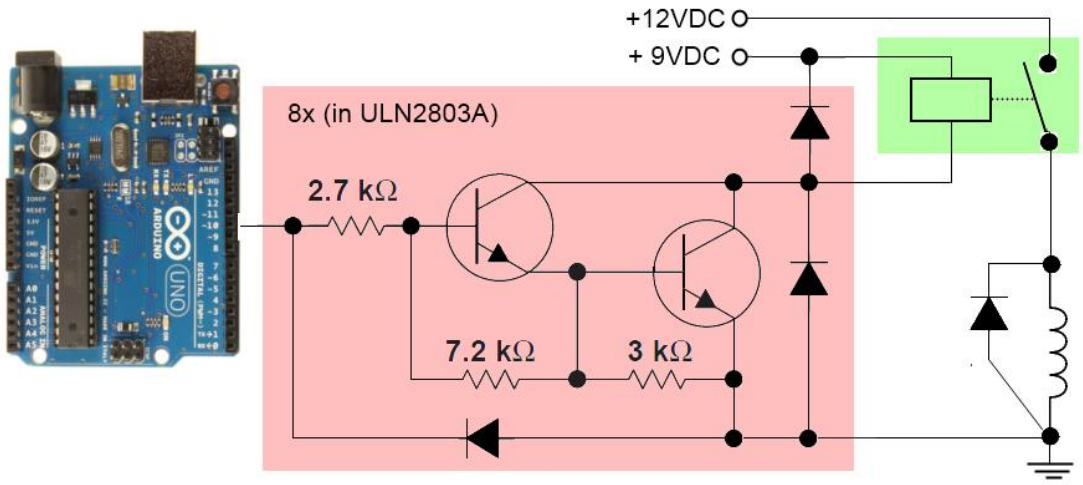

Figure 7: ImitatorUno is built using Arduino Uno as controller board, ULN2803AN to drive the micro relays, the relays switch power to the solenoid and diode prevent a power surge when power to a solenoid id turned off. The +9VDC is connected to the Vin pin on the controller board. Therefor one can choose to provide this power from the USB cable or use a power supply connected to the ImitatorUno. +12V DC must be provided to operate the solenoids.

The imitator can easily be extended to handle a full keyboard. A normal keyboard has less than 128 keys, thus we need only eight digital outputs to control a keyboard. One output line is used for clock and seven output lines($2^7 = 128$) are used for decoding/demultiplexing. The decoding circuits are placed between the Arduino digital outputs and the ULN2803AN circuits. 74LS138 may be used for decoding and SN74LS75 may be used to keep the output states. If we

forget to turn off a solenoid it may overheat. This can easily be avoided using another digital output as enable signal. By setting enable to low all solenoids should be turned off, thus overriding the decoding logic.

The fully assembled ImitatorUno is shown in ImitatorUno, box.

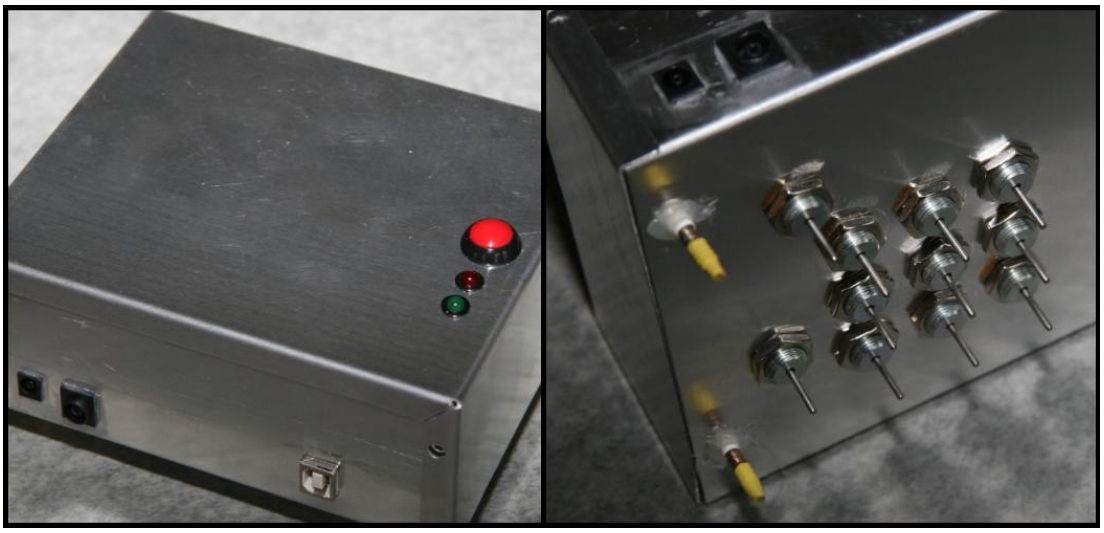

Figure 8: On the top we see one button and two status lights, the function of these are controlled by software. The inputs on the side of the box are 9V DC input, 12V DC input and a USB connector for programming. Underneath we see 11 pins that push the keys 0 to 9 + ENTER. The legs are adjustable and make it possible to use on different keyboards.

5.2.2 Software

The software for the ImitatorUno controller board is written in C++ using Arduino development environment v1. The full code are found in appendix ImitatorUno Software.

The software has two modes of operation, the single signature mode and the experiment mode. In the single signature mode one press on the button make the imitator to type the signature once. In experiment mode the ImitatorUno types all signatures required for a complete experiment. Constants must be set in the software according to the experiment, sessionCount$= 20$, sessionDelay$= 60 \cdot 30$, and tryCount$= 3$. The signatures is specified in three arrays delays[], values[] and actions[]. The delays array is a list of delays in milliseconds, the value array is a list of keys and the action array is a list of actions where 1 is press and 0 is release. The values must be set according to the person one want to imitate.

5.2.3 Data

29 imitator users are created, 10 based on the smart phone users and 19 based the PC users. All imitator users are PC users because they use the PC software to type the signatures.

The imitator typing rhythm is the average of the 10 first signatures from each user. As we have seen signature i of length n consist of a sequence of keys(K) and corresponding key presses(D) and key releases(U), $S_i = \{K, D_i, U_i\}$, where $K = \{k_1, k_2, ...k_{n-1}, k_n\}$ and $D_i = \{d_1, d_2, ...d_{n-1}, d_n\}$ and $U_i = \{u_1, u_2, ...u_{n-1}, u_n\}$. Let the average of the 10 first signatures $\{S_1, ..., S_9\}$ be denoted μS. Then $\mu S = \{K, \mu D, \mu U\}$.

The average values can not be used directly in the ImitatorUno software of two reasons, first the values in μD and μU are relative to the start of signatures instead of delays from last action, second a key may be pressed before the previous is released. Thus conversion is needed. A flexible structure is achieved by converting μS into delays[], values[] and actions[] like this:

1. Set delays = $\{\mu D, \mu U\}$.
2. Set actions = {(set of n ones), (set of n zeroes)}
3. Set keys = $\{K, K\}$
4. Order delays, actions and keys based on delays.
5. Calculate delays. Let delays=$\{d_0, ..., d_{n-1}\}$. Iterate from second to last delay in delays and calculate $d_i = d_i - d_{i-1}$

6 Analysis

In this chapter we start with explaining central terms used later in the analysis. Then we analyze the data sets followed by a description of the methods we are using. The chapters concludes with a summary.

6.1 Notation and Definitions

6.1.1 Definitions

Data set When we talk about data sets we refer to data sets containing keystroke data from multiple users. Since we work with data from several sources separately we do not want to mix them into a global set containing all data. Instead we work with different *named* data sets, described in next subsection, separately.

The structure of all our data sets are the same, and they are only containing feature values. We use the named data set PD to describe the structure. A single feature value is identified by $PD_{u,s,f}$, where $_u$ is the user, $_s$ is the signature, and $_f$ is the feature. All feature values in PD can be expressed using two important sub sets. $PD_u = \{S_1, ..., S_n\}$ where S_i is the i^{th} signature vector for user $_u$ and $_n$ is the number of signatures and also the dimension of the feature vector, and $PD_u = \{F_1, ..., F_d\}$ where F_i is the i^{th} feature vector for user $_u$ and $_d$ the number of feature vectors and also the dimension of the signature vector.

The **signature vector**, $S = \{s_1, ..., s_d\}$ where s_i is the i^{th} element of the signature. The **feature vector**, $F = \{f_1, ..., f_n\}$ where f_i is the i^{th} element of the feature vector. The $_d$ and the $_n$ is already explained.

Observe that when $S_i(_k)$ refer to the $^{th}_k$ element of signature vector $_i$, and $F_j(_l)$ refer to the $^{th}_l$ element of feature vector $_j$, then $S_i(_k) = F_k(_i)$. Thus the signature vectors is the rows in a matrix and the feature vectors are the columns.

Feature vector A feature vector is a list of features. This is the only vector we may refer to as only vector. For a detailed description see the definition of Data Set.

Interval Interval is a easy way to express a set of numbers. We use it in two forms, $(a, b) = \{x \in Z \mid a < x < b\}$, and $[a, b] = \{x \in Z \mid a \leq x \leq b\}$. The interval may also be used with real numbers, then Z is replaced with an R in the definitions, We can deduce from the context whether Z or R is used. We may also use the MatLab notation to express a set of integers, $a : b = [a, b]$ or $a : s : b$ where s is step, for example $1 : 2 : 10 = \{1, 3, 5, 7, 9\}$.

Signature vector A signature vector is a list of features also known as a sample. The signature vector is thus a collection of features from one authentication try. In our case, all features collected when typing 17051402293 one time. We may refer to the signature vector as the signature. For a detailed description see the definition of Data Set.

σ_H and σ_L The spread of values in a keystroke dynamics data set often deviates much from a normal distribution. We often observe distributions where the interval $(\mu - \sigma, \mu + \sigma)$ is outside the interval of the data set, see figure 9. In normal distributed data the same interval should be inside the interval of the data and cover about 68% of the values. It is easy to see that the standard deviation is inadequate in such distributions. We have developed an easy way to adapt the standard deviation to skewness in the data, where we calculate two standard deviations, one for the lower values in the data set and one for the higher values in the data set, see also section 6.2.2.

Let $D = \{d_1, ..., d_n\}$ be the data vector and the median $\tilde{d} = median(D)$. We then we split D into $H = \{x \in D \mid x > \tilde{d}\}$ and $L = \{x \in D \mid x < \tilde{d}\}$. Let l_i be the i^{th} number in L, and let h_i be the i^{th} element in H. We can now calculate the low and high standard deviation for D as:

$$\sigma_H = \sqrt{\frac{2}{n-1} \sum_{i=1}^{\lfloor n/2 \rfloor} (\tilde{d} - h_i)^2} \tag{6.1}$$

$$\sigma_L = \sqrt{\frac{2}{n-1} \sum_{i=1}^{\lfloor n/2 \rfloor} (\tilde{d} - l_i)^2} \tag{6.2}$$

Formula 6.1 and 6.2 hold both when |D| is odd and when |D| is even. This is easy to see if you assume that H and L is the half of a symmetric vector around $\tilde{d}$. Observe that this method is not perfect either, because it is ignorant to the distribution in H and L. When the values of D is normal distributed $\sigma = \sigma_H = \sigma_L$.

$\min_\sigma$ When calculating any spread in values using short vectors of discrete values, we risk to measure no spread. e.g. $\sigma = 0$. When using $\sigma = 0$ in scaling this will lead to errors. In our data the maximum resolution is 1 millisecond, but it is common with keyboards having 8 millisecond response time, see section 6.2.1. Assume $D = \{d_1, ..., d_n\}$ where d_i is the i^{th} duration and $\forall d \in D \mid d = 72$ and let the resolution $r = 8$. The real value of d is then in the interval $(72-8, 72+8)$, see section 6.2.1. We assume an even distribution[1] of durations in the given interval and calculate a step s to describe this $s = 2r/n$. Since all distances are a multiple of equal steps and we assume an equal distribution we can simplify the formula for mean and express the standard deviation based only on r and n:

$$\mu = \frac{(d - r + s) + (d + r - s)}{2} = d \tag{6.3}$$

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\mu - (d - r + si))^2} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (si - r))^2}$$

[1] In this special case a normal distribution may be more correct. Assume we have many data points within a narrow range, and assuming an even distribution, it is then unlikely that there are no values outside the given range. Nevertheless we choose an even distribution because of simplicity and uniformity, ref section 6.2.1.

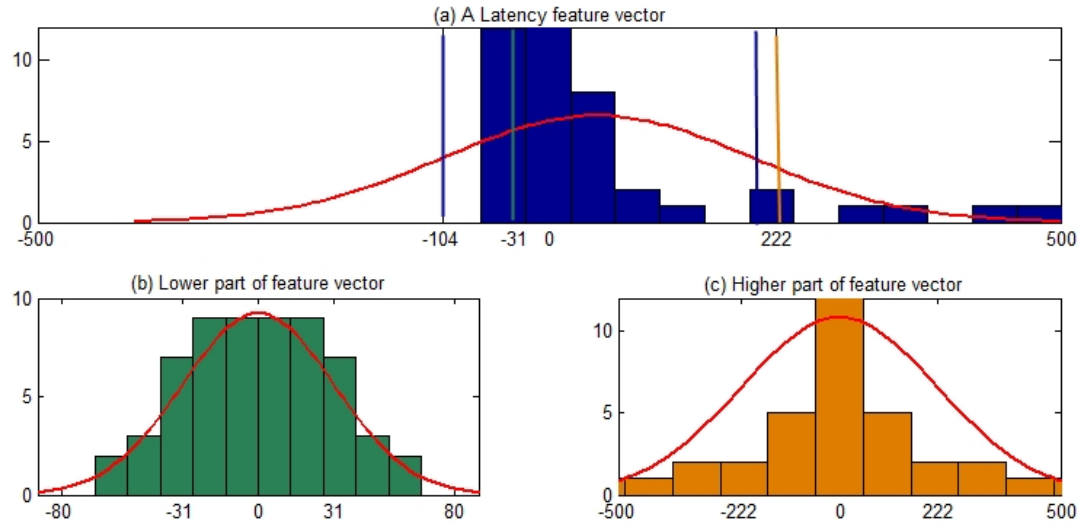

Figure 9: Dual Standard Deviation. In (a) we can see one latency feature vector from a real keystroke dynamics data set, minus median. The blue vertical lines at -104 and 199 indicate the one standard deviation from mean. (a) is split by the median which is 0, and the lower part is shown in (b) and the higher part is shown in (c). In addition the mirror data is included in (b) and (c) to make the vectors symmetric around the median. The standard deviation from (b) is shown as a vertical line in (a) at -31, and the standard deviation from (c) is shown as a vertical line in (a) at 222. This briefly describes how to calculate the dual standard deviation.

$$= \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\frac{2ri}{n} - r))^2} = r\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\frac{2i}{n} - 1)^2} = r\Delta \tag{6.4}$$

By computer simulation we found that Δ converges as seen in formula (6.5).

$$\lim_{n\to\infty} \Delta \approx 0.5774 \Rightarrow min_\sigma = r \cdot 0.5774 \tag{6.5}$$

6.1.2 Named Data Sets

Three main classes of data sets are used during the analysis. PC data(PD) including imitator data(IP/IS), smart phone data(SD) and a external reference data set(EXT)[35]. The SD and PD data were both collected from the Data Collection Experiment in section5.1, and the IP and IS data were collected during the Imitator Experiment in section 5.1.

PD and IP, PD and IP have the same structure, thus only PD is described. PD $= \{U_1, ..., U_{19}\}$ where U represent each of the 19 users. U=$\{S_1, ..., S_{60}\}$ where S_r is the r^{th} signature. A signature is a set of durations and latencies $S_r = \{Durations_{1:12}, Latencies_{1:11}\}$. Durations and latencies are further described in section 4.1.1.

SD and IS, SD and IS have the same structure. SD is in several aspects similar to PD and only the differences are described. SD contain data for only ten users, and the signature are dif-

ferent. The signature is a set of durations, latencies, x positions, y positions, pressures, size down size up, and swipes. $S_r = \{D_{1:12}, L_{1:11}, X_{1:12}, Y_{1:12}, P_{1:12}, SD_{1:12}, SU_{1:12}, SW_{1:12}\}$.

EXT The EXT data set is described in detail[35], and we only list some of the key differences. They used .tie5Roanl as password. As they use one less key in the signature they have twelve durations and eleven latencies. In addition they use the latency(DD), which actually is $DD_i = duration_i + latency_i$. The use of a alphanumeric password also means that the user may use both hands and operate over greater distances on the keyboard. So even if the password is shorter it should differentiate the users better, give a greater range in learning and, when learned, typed faster than our password which normally is typed using only one hand. EXT is collected using a bigger population, 51 users, and the signature is typed 400 times by each user. Also their signatures were collected in sessions, but they collected 50 signatures in each session and waited at least one day before next session. The timing accuracy in EXT is controlled and is within ± 0.2 milliseconds.

$F_{u,r,c}$ However, no testing is done directly against the described data sets. Instead a new data set F is generated by feature selection. By doing this we ensure that the same treatment and methods are used regardless of what source we use for the features. The structure of F is described in *Data set* in definitions section above.

6.1.3 Outlier

Outliers are commonly known as data deviating so much from other data that one might suspect that the source of data is corrupted. In keystroke dynamics such data could be any interruption while writing e.g. noise, sneeze, uncomfortable typing position or any cognitive breaks. We have two ways of detecting outliers, one is commonly used and here referred to as *standard*, and one method, not seen elsewhere, named skewed because it is based on a standard deviation adapted to a skewed distribution.

Standard3 A common way of detecting mono variate outliers is based on using the standard deviation. Values outside the interval $(\mu - 3\sigma, \mu + 3\sigma)$, where μ is the mean of the data set and σ is the standard deviation of the data set, are considered as outliers.

Skewed3 When using the standard method on skewed data we risk removing to much data on the long-tail end and missing outliers on the other side. We propose the following method using σ_H and σ_L described in 6.1. Values outside the interval $(\tilde{x} - 3\sigma_L, \tilde{x} + 3\sigma_H)$, where $\tilde{x}$ is the median of data set, are considered as outliers.

6.2 Data

6.2.1 Timing resolution and accuracy

Before using our data we need to examine it for noise. One source of noise is the sampling of events. [40] found that the windows-event clock has a resolution of 15.625 milliseconds. However, there are three obvious sources for this kind of noise, the priority the foreground application has on keyboard handling, the USB bus load and how fast the keyboard itself are scanning the keys for actions. Eg. in old OS like windows 95/98 one could set a value in the

system.ini file, KeyBoostTime to a higher value to make the foreground application increase the priority of keyboard handling. Gaming keyboards are often promoted with the claim "*response time up to 1 millisecond, 8 times faster than a normal USB keyboard, which is 8 milliseconds*".

Since every participant decide their own experimental environment we need a overview of this kind of noise. Most likely the processor and bus are not overloaded when doing the experiment, thus the following test will tell us each participants keyboard response times. The response times may differ between duration and latency. Therefore we need to check response time from duration data and latency data separately. The procedure below is shown by using durations, and is used to determine one test subjects timing resolution:

1. Make **duration clusters** by clustering all durations. We use all durations, that is 12 durations per signature and 60 signatures. A cluster is a list of values. When we start we have no clusters. For each duration, we search for clusters containing similar values, which is values not more than 2 millisecond different. If we find a cluster we add the new duration, if we do not find a cluster we create a new cluster and add the duration. When all durations are clustered we need to clean up. First we turn together groups that contains similar values, not more than 2 milliseconds different. Second we remove small clusters, which we consider as noise.
2. Make **distance clusters**. We follow the same procedure as in step 1, but this time the values we cluster are the distances between the duration clusters we made in step one. We use the same cluster criteria, the values should be similar, not more that 2 millisecond different. We do not remove any clusters during clean up this time.
3. Decide the response time. If more than 50% of the durations are in clusters that have a greater range than 20 milliseconds, then the resolution is considered to be 1 millisecond. This mean that the values in the cluster are spread over at least 20 milliseconds, and that there are not any gaps between two consecutive values greater than 2 milliseconds if we sort the values. If no such wide groups exists we base our decision on the average value of data in the the distance clusters from step 2. The resolution is then the smallest distance between any two such average values.

Table 1 show the resolution of our data. It is not checked if we get the same results using only the template, but if it is possible it could be used in calculation of $\mathtt{min}_\sigma$, see section 6.1. Time is a continuous measure, which we already have dicretized into discrete values of one millisecond. When we use slow keyboards or slow systems, it result in lower resolution and less accuracy. Another consequence is that the probability of equal values increases. Consider the process of making a template. Then we may want to calculate the standard deviation for the first duration of e.g. ten signatures. If all values are equal the standard deviation is zero. This is bad for two reasons, first because it is wrong which we easy can see in figure 10, secondly zero is not suitable for scaling a value.

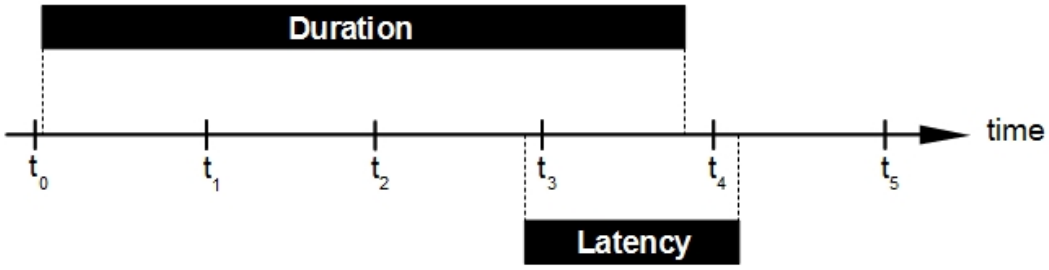

Figure 10: The figure show how the accuracy of the duration and the latency depends on timing resolution. t_0 to t_5 is moments in time where actions are detected e.g. keyboard is scanned for key-status. The resolution r is then $r = t_n - t_{n-1}$. In this example the duration is measured to $3r$ and the latency are measured to $2r$, however we see that the duration are closer to $4r$ and the latency closer to r. The accuracy of a timing value x is therefore $\pm r$.

6.2.2 Timing distribution

We have discussed timing resolution as one noise factor, another factor is outliers. Table 2 show how several descriptive statistical measures are influenced by introducing outliers in a normal distributed data set. Thus, we need to consider precautions against outliers when using the collected data. However, keystroke dynamics timing data are seldom normal distributed. In figure 11 we see an example from our PD data set showing the various feature distributions as box plots. We have a closer look at feature 3 from the box diagram as a histogram 12. Notice that $\mu - \sigma$ is less than minimum in the feature vector. Thus, the standard deviation may not be a proper way to detect outliers in keystroke dynamics timing data. Outliers on the heavy side may be overlooked and too many data points on the long tail side may be classified as outliers.

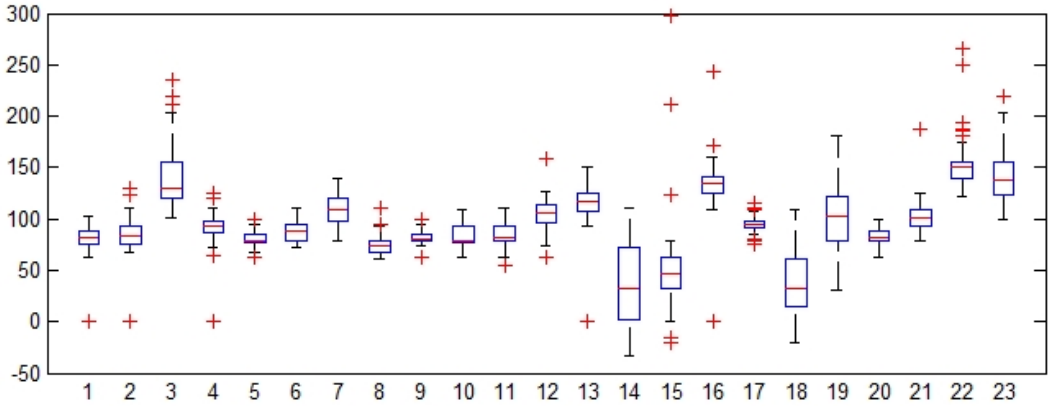

Figure 11: Timing distribution of the PD, test subject 7. The features 1-12 are durations and feature 13-23 are latencies. Feature 14 and 18 are examples of vectors containing negative latencies.

Resolution	Duration	Latency	Both
1-3	PC2, PC7, PC9, 13-PC, PC15, PC16, PC17, PC19, SP6, SP21, SP22, SP23	PC2, PC10, PC7, PC9, PC13, **SP5**	PC2, PC7, PC9, PC10, PC13, PC15, PC16, PC19,**SP5**, SP6, SP22, SP23
4-6	PC10	PC4, PC5, SP6, SP7, SP11, SP15, PC17, SP20, SP21, SP23	PC4, PC5, PC17, SP11, SP21
7-9	PC1, PC12 , SP2	PC1, PC11, PC12, PC15, PC16, PC18, PC19, SP22	PC1, PC11, PC12, PC18, SP7, SP15, SP20
10-12	PC4, PC5, PC11	SP2	SP2
13-17	PC3, PC8, PC14, PC18	PC3, PC6, PC8, PC14	PC3, PC6, PC8, PC14
18-22	**SP5**, SP7, SP11, SP15, SP20		

Table 1: Timing resolution in our data. The resolution is in milliseconds and the test subjects are participants from 1 to 23 where SP refers to smart phone and PC to a computer with physical keyboard. The duration column show the result using only duration features, the latency column are results using latency features only, while the last column show results using all features. Some devices have different result depending on features used, e.g **SP5**. Such results are manually inspected.

Description	Median	μ	σ	Skew	Kurtosis	Count
Standard Normal Distribution	0	0	1	0	3	68
One outlier 10 * max	0.01	0.25	2.7	7.8	72	98.5
Two outliers 10 * max	0.02	0.5	3.6	6.1	41	97.9
Three outliers 10 * max	0.04	0.75	4.4	5	28.1	96.9

Table 2: Robustness of statistical measures. Standard normal distribution is generated using randn in MATLAB with 100 values. Count is number of values within one standard deviation from mean.

We therefore need a better way to detect outliers when the data vector has a skewed distribution. Initially we had two ideas for how to solve this. (1)To cluster data points using the distance between them, and then use cluster size and distance to other clusters as input to decide on outliers. (2)To adapt the three times standard deviation to skewed data. In the initial test we did not succeed with method one and hence put it aside, but method two is promising. The idea is to divide the vector in two parts, and then to calculate a standard deviation relative to the original median or mean. The problem may be solved using mean and median in several combinations. We wanted to find the optimal combination and conducted a search of all possible variations using our PD data set.

Using method two, the detection of outliers may be divided into three steps, where each step has the following options for choosing it's individual reference; (1)always use mean, (2)always use median, (3)use median if positive skewness, or (4)use mean if positive skewness. Be aware that the described procedure only consider outliers with higher values, as we may apply any result on the lower value outliers later.

1. Divide the feature vector into two parts, by the chosen reference.

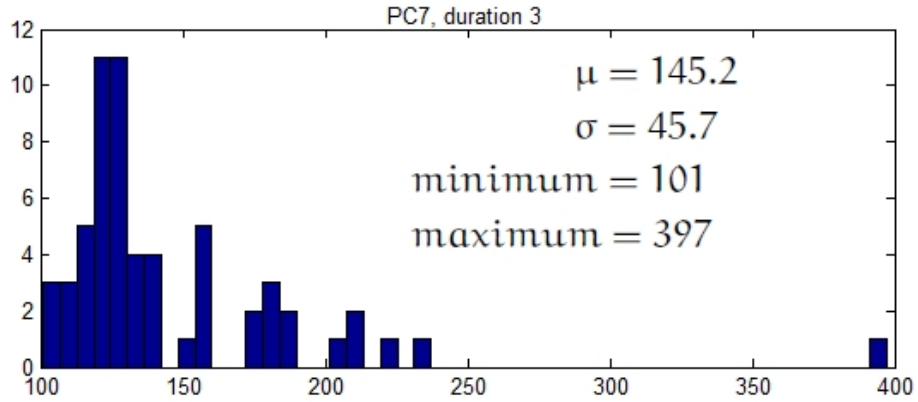

Figure 12: Histogram for the duration feature nr 3 from PD test subject 7.

2. Calculate standard deviation σ_H relative to the reference.
3. Find the limit where greater values are outliers using the reference $+3\sigma_H$.

In our testing we used an imperfect but simple method to detect skewness. We say it is positive skew when median has lower value than the mean. This is not true for all distributions, but when comparing this simple method against a proper skewness measure on our PD data we got the same results. By using the described procedure, where we have three steps, and where we in each step have four options, we have 64 variations to test. In step one where we divide the feature vector, we need one more option to cover a special case. If several data points have the same value as the median, we could end up with one or two very short halves. We introduce a new option to test, where we just divide the sorted vector into two parts of equal length. In total we have 80 combinations to test on.

We have a method, but do not know how good it is. At this point we can only do a *subjective test*, where the main goal is to learn and get ideas when we see how the two methods compare, and develop a test method we can reuse later when evaluating detectors, and not most important find the subjective performance to detect outliers.

We manually and subjectively inspected 152 of 437 data vectors from the PD data set. For each vector we subjectively decided how many outliers there should be based on a histogram, and stored the count in a score table. Then we ran our test using the described procedure and tested the result against the score table. The result from this comparison is found in table 3. The combination of options giving best performance is to always use the median to divide the vector into two halves, then using median to compute standard deviation when skewness is positive, and always use median as reference to find the limit where we consider higher values as outliers. The ides achieved here is used in the description of σ_H, σ_L and outliers in section 6.1.

Another aspect of outliers are how often do they occur. From table 4 we can see that we have similar results from the EXT data set from our PD data set. Typically less than 25% of

Performance	Standard3	Skewed3	Min	Max	σ	μ
Good	72	123	77	123	18.8	103.9
Ok	26	10	8	17	2.8	11.9
Bad	54	19	19	62	10.1	36.2

Table 3: Outlier detection performance. The table compare two methods of mono variate outlier detection, described in section 6.1, against a manually created score table. The score table contains the number of outliers in each of the selected 152 feature vectors. Some outliers are obvious and rated as *Good*, and some are more uncertain and rated as *Ok*. Outliers detected that are not in the score table are considered as *Bad*. The scores given are the number of outliers detected by each method. The Skewed3 method has 80 variations, the best is shown, and the statistics of all is shown in the four right columns.

the signatures contains outliers, and most of the latency features are affected while most of the duration features are not affected by outliers. When we use Skewed3 to detect outliers the average number of outliers per test subject is reduced from 20 to 12.3.

	N	OK	OK-DU	OK-UD	μ	σ
Signature	60	46.1(3.3)	-	-	1.4(0.3)	0.8(0.4)
Signature(t)	60	43.6(7.1)	-	-	1.8(0.5)	1.2(0.6)
Feature	23	8.7(3.3)	7.4(2.6)	1.3(1.6)	1.4(0.2)	0.6(0.2)
Feature(t)	23	10.2(3.1)	7.2(2.7)	3.1(1.8)	2.6(1.0)	1.9(1.4)
Signature(e)	60	46.3(2.6)	-	-	1.4(0.2)	0.7(0.2)
Signature(t,e)	60	39.8(6.8)	-	-	1.5(0.3)	0.8(0.4)
Feature(e)	21	7.4(2.2)	6.8(1.8)	0.6(0.7)	1.4(0.1)	0.6(0.1)
Feature(t,e)	21	8.1(2.7)	4.5(1.5)	3.6(1.5)	2.3(0.8)	2.1(1.7)

Table 4: Outlier distribution in pc timing data. There are average 20 outliers per test subject. Values are average on the 19 subjects, with standard deviation in brackets. The columns N show number of tests, column OK show how many tests without outliers, mean μ and standard deviation σ describes how many outliers there are in tests NOT OK. The columns OK-DU and OK-UD show the OK column divided in durations and latency features. Method used to detect outliers are 3Standard. (t)indicate that only the template is used to calculate limits for outlier tests, and (e) is the tests performed on the EXT data set[35]. To make it possible to compare results from EXT and PD, 19 of 51 subjects are randomly selected and the 7-8 first tries in each session is used.

6.2.3 Touch Screen Data

In addition to the well known timing data, one can collect more data describing the typing characteristics from a touch screen. In our experiment we collect position data, pressure data and size data. We also generate swipe data from either of these three data classes. Any of these features are, possible and in varying degree, biased by the device used. With our limited data set it is not possible to determine the degree of such bias. The bias may increase the separability, but may also cause problems when a user change her smart phone. The latter can be avoided by collecting the unique device id and make the user enroll again when start using a new device.

The rest of this section describe the touch screen data in our experiment.

Position

Position is a measure on where the finger hits the key. In our data set the position is given by two features x and y, for each key. Both is relative to the size of the key, and have values between 0 and 100. One signature contains 24 position features, and the distribution of our data is shown in figure 13. The first 12 features are x-values and the latter 12 are y-values. A trend for all our test subjects, and also in the figure, is that the variation per feature is larger in y-direction that in x-direction.

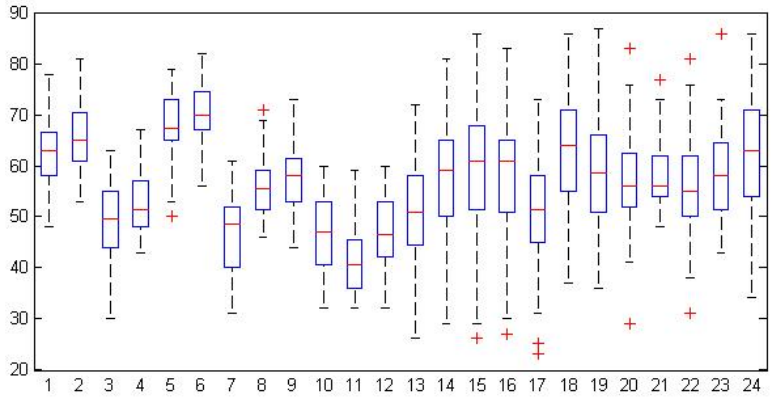

Figure 13: The diagram show the distribution of position features for one randomly selected user. The 12 first features are x-values, while the 12 last are y-values.

Pressure

Pressure is a measure of the force applied by the finger on the touch screen(button). The scale of the pressure measure seems to vary between devices. Two of the devices seem to not measure the pressure. We assume this because the values does not change and it is unlikely that a user apply exactly the same pressure for the complete experiment. Both user having constant but different pressure values(39 and 156) were using SEMC smart phones, with same OS release installed. The five users that used Samsung GT-I9100 applied pressure between 33 and 266, the Nexus one user has feature values between 66 and 133, the HTC user has values between 66 and 200, and the Huawei user has values between 172 and 474. The pressure distribution for one Samsung user is show in figure 14.

Size

The size is a measure of the surface of finger in contact with the key when key down and key up are detected. The size when key is pressed are consequent less than size when key released, in our data set. The size values seems less device dependant than the pressure values, with one exception. The Huawei device has much larger values for size on key release that the others. The Huawei range from 3000 to 8000, while the other range from 50 to 600. No device present significantly different values for size on key pressed, which range from, -11 to about 88. How a

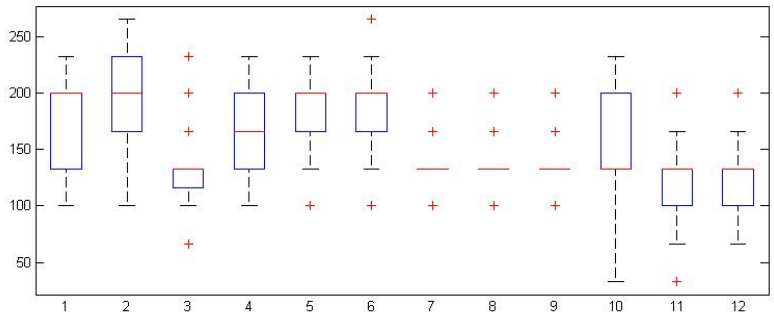

Figure 14: The diagram show the distribution of pressure measures for one randomly selected Samsung user.

value can be negative is not revealed. An example size distribution is shown in figure 15.

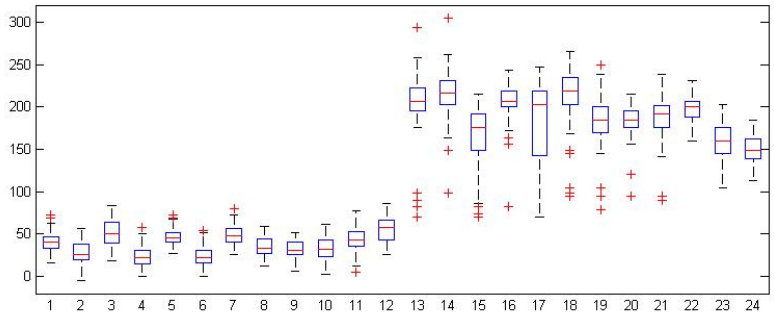

Figure 15: The diagram show the distribution of size measures for one randomly selected user.

Swipe

When a user moves the finger across screen surface when a key is pressed data sample containing position, pressure and size are collected. We have not considered what the actual pressure, position and size values are during a swipe, but we can use either of these to collect the swipe feature. We illustrate this using two pressure vectors, P1 and P2, acquired pressing key1 and key2. $\mathrm{P1} = \{\mathrm{P}_{\mathrm{down}}, \mathrm{P}_{\mathrm{up}}\}$ and $\mathrm{P2} = \{\mathrm{P}_{\mathrm{down}}, \mathrm{P}_2, \mathrm{P}_3, \mathrm{P}_4, \mathrm{P}_{\mathrm{up}}\}$ then the swipe is $swipe(\mathrm{P1}) = 0$ and $swipe(\mathrm{P2}) = 3$. As described earlier we have always two pressure values, one when the key is pressed and one when the key is released. If we have more values we count those exceeding the two, and refer to it as swipe.

Some users swipe only occasionally where most signatures are without any swipe, others swipe a little on every signature, while two users swipe regularly on every signature and every feature. To avoid a lot of zero values we calculate the mean swipe per signature. The distribution of swipe data for all users are shown in figure 16.

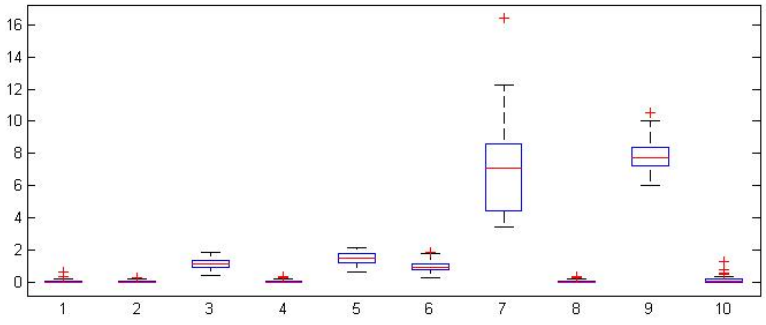

Figure 16: The diagram show the distribution of mean swipe per signature. It is one box plot for each user, since it is only one swipe feature per signature.

6.2.4 Imitator data

Imitator data is data collected by using the ImitatorUno5.2. The data from it was initially collected using a laptop. But, the accuracy of the timing data was insufficient. To be useful as a tool to imitate someone, the variance in data must be controlled and not be a result of external factors. We solved this by using the ImitatorUno in conjunction with a gaming keyboard that have good response time.

As described in section 5.2, typing patterns are generated by using the mean values from the users template. For the ImitatorUno the variation is calculated using all duration and latency features from the ten first signatures. Using only the ten first signatures from template is not optimal, but because it seems to be sufficient is has not been prioritized to reprogram it to use all twenty signatures. The composed latencies, DD and UU, are not used. The values received from the ImitatorUno is stable, and it is not necessary to look at the individual data. We calculated the standard deviations for each feature vector for test subject one. From these 23 values we calculated minimum(3.5milliseconds), mean(6.6 milliseconds) and maximum(8.5 milliseconds. For the same twenty three feature vectors we also calculated maximum absolute distance to its mean, which is 25.2 milliseconds.

In figure 17 we can see how this small variation affect the performance. The distances between the imitator signature and the template are minimal and much less the distance between the genuine user and the template. If we introduce a second threshold it will be possible to reject imitator attacks based on template. The natural countermeasure from the imitators side will then be to introduce variations similar to the genuine user. We discuss this further within the security discussion.

6.2.5 Learning

The performance and stability of most human actions generally improves with practise. This effect do also apply to keystroke dynamics and we investigate the effect learning has in our data set. Figure 18 illustrate how latency, latency variation and number of outliers change over time.

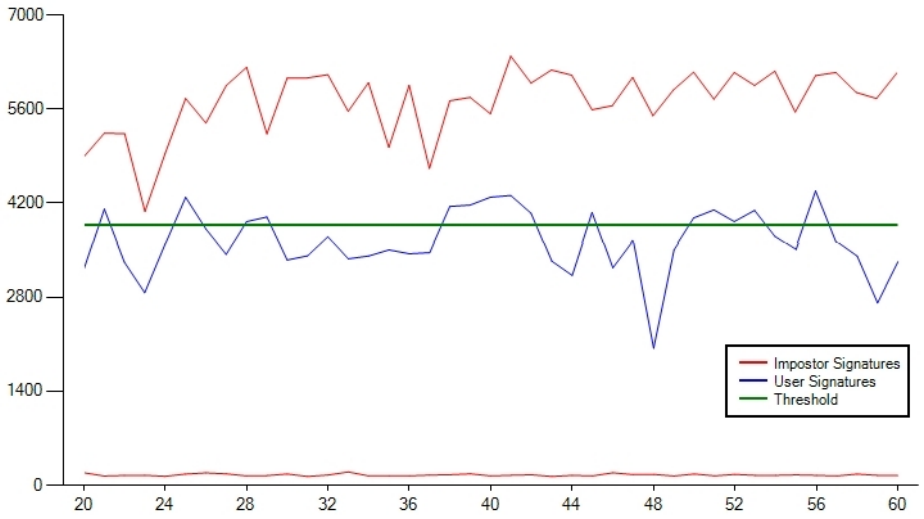

Figure 17: The diagram illustrate how the ImitatorUno perform compared to a genuine user and another impostor. Each graph is assembled using all 60 signatures available from the PC part of the experiment. The user signatures are from PC-1, the impostor above threshold is PC-2 and the impostor at the bottom of the diagram is the ImitatorUno using template from PC-1. The threshold is the threshold for EER using all other PC signatures as impostor signatures, in addition to the two shown in the diagram. Plain manhattan is used to calculate the distances.

Learning is also affected by how many times the signature is typed during one session. We use many small sessions while the EXT data set use larger sessions, typing 50 signatures each time. Figure 19 show the variation in latency when using larger sessions.

6.3 Methods

The main goal of the *Methods* section is to describe how various tasks are solved and why. Results are then shown and discussed in section Summary. However, some results are used in this section too, mainly to support choice of solutions and methods.

6.3.1 Detectors

When analyzing data we found that A detector is used to measure the distance between a template and a signature. Based on the distance we can decide if we believe the signature belongs to the genuine user or an impostor. How well a detector distinguish between a genuine user and an impostor is referred to as the performance. In the literature most detectors have different performance in different research, even the ranking of detector performance change. Further, it is difficult to explain these variations, if not impossible. To avoid ending up with unexplained results we compare performance using a external public available data set[35] against performance using our own data set. Another benefit of using the external data set is easy validation of our test environment, by comparing performance of the same detectors they have used. From their research[36] we use the Euclidean because it is commonly used and we use the Manhattan detector because it is simple and among the best performing.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1	2	0	12	0	0	0	1	0	40	0	10	0	0	0	0	0
2	3	2	7	0	1	3	0	1	0	40	0	0	0	0	0	0
3	36	0	4	0	0	7	5	5	40	0	40	0	0	8	2	0
4	3	0	15	10	0	0	14	4	0	0	0	40	0	0	32	0
5	3	21	0	0	7	23	0	6	0	7	1	0	40	13	0	0
6	18	13	6	0	14	5	0	8	1	0	25	0	0	40	0	0
7	11	3	33	0	0	3	3	0	0	0	0	0	0	0	40	0
8	40	3	39	0	14	25	19	14	40	0	40	0	0	40	28	40
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5: The table show number of false matches and number of false non matches for a subset of the test subjects. The rows is genuine user and the columns are attackers. The number interval [1,8] is PC test subjects and the interval [9,16] is from ImitatorUno programmed to imitate each of the test subjects. The diagonal where row number is equal to column number show number of false non matches. All the other numbers are false matches. The total number of attempts, both genuine user and impostors, are 40. The test included all PD and IP test subjects in experiment setup using detector Manhattan(fn). The EER for this test is 0.101.

Both Manhattan distance and the Euclidean distance are a L_k norm of the general Minkowski[54] distance. Where Manhattan is the L_1 norm and Euclidean is the L_2 norm. In the equation (6.6), T are the template vector and S are the signature vector,t_i and s_i refer to the i^{th} element in each vector, the d refer to the length of the vectors. The rest of this section are devoted to these two detectors and the various forms they may be used in.

$$L_k(T,S) = \left(\sum_{i=1}^{d} |t_i - s_i|^k \right)^{\frac{1}{k}} \tag{6.6}$$

Euclidean

Euclidean distance is L_2 norm of the general Minkowski distance, which is the basic form of the Euclidian detector referred to as only Euclidean. During enrollment a template vector T is created by using n training signatures of length d. The i^{th} element of T is calculated as $t_i = mean(F_i) = 1/n \sum_{s=1}^{n} f_s$, where f_s is the s^{th} element of the i^{th} feature vector. In the testing phase the distance $L_2(T,S)$ between the template vector and the provided signature vector is calculated. If $L_2(T,S) < ts$, where ts is a limit known as threshold, the signature is accepted as genuine, else it is rejected as an impostor signature.

In addition to the basic form, Killourhy and Maxion [36] use the Euclidian in the form, normed. We describe this below.

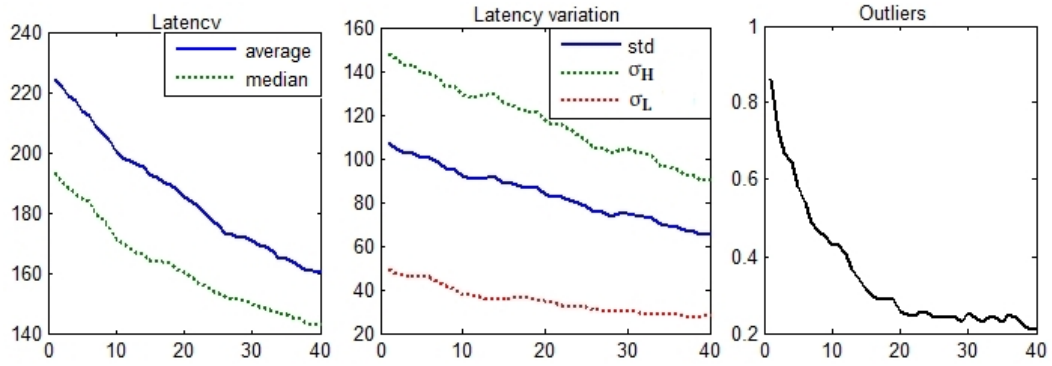

Figure 18: Three measures of learning. The y-values for latency and latency variation are both in milliseconds, while outliers are average number of outliers per signature. A gliding window is used to smooth the graph. In the first window we calculate the average latency of the 20 first signatures as value 1 in the graph, the next window is signature 1 to 21, used to calculate value 2 in the graph , and up to the last window using signature 41 to 60 to calculate value 40 in the graph. Outliers are removed.

Manhattan

Manhattan distance is the L_1 norm of the Minkowski distance in it's basic form. We refer to the basic form as just, Manhattan. During enrollment a template vector is created as described for the Euclidean detector. In the testing phase the distance $L_1(T, S)$ between the template vector and the provided signature vector is calculated. If $L_1(T, S) < ts$, where ts is a limit known as threshold, the signature is accepted as genuine, else it is rejected as an impostor signature.

There are two more forms of the Manhattan detector in [36], the filtered and the scaled form. We describe both below.

Filtered(f)

Filtering is done during template creation, and is about removing outliers from the template, and thus making the template more representative for the user.[7] used the technique to calculate a more robust mean as template, They used the *standard3*, from section 6.1 to detect outliers, but only the outliers above mean were removed. The performance is only slightly improved when also removing outliers below mean, thus we implement filtering both above and below mean in all results presented using filtering. Any detector using filtering are marked with an (f), e.g. Manhattan(f).

Scaled(s)

The scaled form of Manhattan is described by Araújo et.al.[27]. Each dimension of the template vector and the signature vector are scaled by a factor. [27] refer to the scaling factor as standard deviation while [36] refer to the same factor as the average absolute deviation. We generalize the scaling factor Sf to be used with equation(6.6), and call it a scaling factor Sf_k (6.7). When used with the Manhattan distance the Sf_1 is used and when used with the Euclidean distance the Sf_2 is used. $Sf = \{sf_1, ..., sf_d\}$ where sf_i is the i^{th} element of the scaling factor vector, and $sf_i = Sf_k(F_i)$.

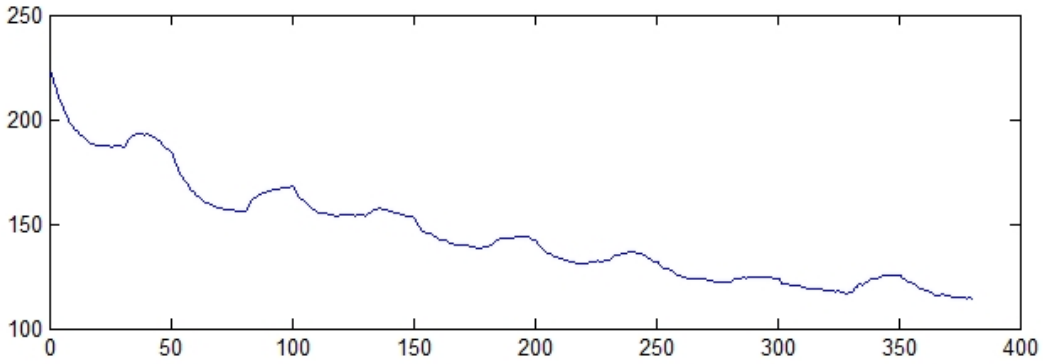

Figure 19: The figure above show how learning when signatures are given in sessions with a delay between. The graph is calculated from the EXT data set[35], where 51 users typed 400 signatures in eight sessions. We clearly see how we forget between the eight sessions. Outliers are removed.

$$Sf_k(F) = \sum_{i=1}^{n} \frac{|f_i - \mu|^k}{n-1}, \quad \mu = \frac{\sum_{i=1}^{n} f_i}{n} \tag{6.7}$$

The scaling factor vector can now be used in the testing phase as shown in equation (6.8) to calculate the scaled Minkowski distance Ls_k.

$$Ls_k(T,S) = \left(\sum_{i=1}^{d} \frac{|t_i - s_i|}{sf_i} \right)^{\frac{1}{k}} \tag{6.8}$$

Now we take a closer look at what happens when we use this form of scaling. As we have seen in the data analysis the feature variation is individual per user, and within a users data each feature has different variance. Without scaling any distance between a signature feature and a template feature has the same contribution to the total distance, regardless of the feature variance. This is bad, because a distance on 60 milliseconds from a signature feature to mean, may be within the normal for some features and considered very irregular for other features. By the described scaling we convert the scale from time to deviation units, and now the contribution from the individual feature is dependant of the feature variance.

Another benefit of scaling is that it is easier to set a good threshold, because the mean distance contribution per feature should be 1. This depends on that the template is a good predictor for the yet unknown data.

Normed(n)

[36] defines the Euclidean normed as $L_2/(\|T\|\|S\|)$. By using formulas from [54] page 606, we can express this using equal notations like this, $\|\Delta\|/(\|T\|\|S\|)$, where $\|\Delta\| = \sqrt{(T_i - S_i)^t (T_i - S_i)} = \sqrt{\sum_{i=1}^{d} (T_i - S_i)^2}$. We regard this as a peculiar distance metric, and even with worse performance that the other we still include it and name it as *EuclideanX*.

We reuse the term *normed* for another type of scaling. We use the term from "standard *normal* distribution" where the mean is zero and the standard deviation is one. The idea is to change the

mean to zero and change scale to standard deviation units, individual per feature. The method may be used in combination with any detector since it is done on the date before it is used in the actual detector. For simplicity we have made an variant of the general Minkowski distance for this form too, Ln_k. Where Euclidian distance use the Ln_2 and the Manhattan distance use the Ln_1 form. In equation(6.9) T is a template vector containing mean values for each feature, and σ is the standard deviation, as we know it, per feature vector.

$$Ln_k(T,S) = \left(\sum_{i=1}^{d} \frac{|t_i - s_i|}{\sigma_i} \right)^{\frac{1}{k}} \tag{6.9}$$

Performance

We have described some detectors used by [36], and we have generalized on the forms of these detectors. Now it is time to see how the various combinations performs on their setup, see table 6. The known variants from [36] are in **bold**. We can see that the result are the same as they did, They got the best result, EER=0.096 , using manhattan(s), we got the best result **EER=0.087**, when using normed and filter.

EER	**Euclidean**	**Manhattan**
X	**0.216**(0.119)	
Basic	**0.171**(0.095)	**0.153**(0.094)
Scaled(f)	0.162 (0.092)	**0.135**(0.084)
Filter(s)	0.108 (0.084)	**0.096**(0.069)
Filter and scaled(fs)	0.105 (0.083)	0.091 (0.064)
Normed(n)	0.105 (0.068)	0.093 (0.070)
Normed and filter(nf)	0.103 (0.065)	0.087 (0.067)

Table 6: Performance - detector forms. The emphasized results(EER) are using detectors from [36], the other are from variations over the same techniques they have used. The X is the same as they refer to as normed. The normed variants shown in the table are not from their research. Each result is an average of the test ran for all users, thus the number in brackets are the standard deviation of the set of results.

There are several problems with this setup in [36]. They use signature 1:5 as impostor signatures, while signature 1:200 in another test is used as template. That they use 200 signatures as template is also not very realistic. When we use the signatures 201:205 as impostor signatures, the Manhattan(nf) is still best but now it has EER=0.014(0.095). When we use signature 1:20 from practice, 21:80 for enrollment, 81:400 for testing and 81:91 as impostor signatures, the Euclidean(sf) performes best with EER=0.154(0.10). In all these tests the X variant is far worse than the others and is not included in further research on our part. The interesting point in the change in setup is the reorder of detectors when considering performance. In most situations Manhattan outperform the Euclidean detector, but not always. Is it possible to take advantage of the properties of the Euclidean detector in certain situations?

6.3.2 Dual Scaling(+)

During data analysis we discovered that adapting the standard deviation to skewed data may have advantages. Based of the findings, we have described how we can calculate two standard deviations from one feature vector, σ_H and σ_L, see section6.1. When discussing detectors and the various forms, we used the standard deviation both on the normed form and the filter form.

Based on the good effect scaling has on performance and the fact the timing data in keystroke dynamics not are normal distributed, we describe a new detector-form based on using σ_H and σ_L instead of σ. We refer to the form as dual scaling. We can use dual scaling both on the filter form and the normed form, and we indicate the use by marking the detector with (f+) or (n+).

During enrollment the median, σ_H and σ_L are calculated for each feature. We use Minkowski to illustrate the formula. The template T now contain both the median t, the σ_H and the σ_L for each feature vector.

$$Ln+_k(T,S) = \left(\sum_{i=1}^{d} \frac{|t_i - s_i|^k}{\sigma_i} \right)^{\frac{1}{k}}, \quad \sigma_i = \begin{cases} \sigma_H, & \text{if } s_i > t_i \\ \\ \sigma_L, & \text{if } s_i < t_i \end{cases} \tag{6.10}$$

The Manhattan(n+) is implemented using the $Ln+_1$ and the Euclidean(n+) is implemented using the $Ln+_2$.

Dual scaling may also be used when using filter, then we use skewed3 instead of standard3 to detect outliers. Skewed3 is described in section 6.1.

6.3.3 Evaluation

Until now we have done ad-hoc testing and used a setup giving same results as in [36]. Here we describe the methodology used to evaluate the detector performance on the named data sets EXT, PD and SD described in 6.1. We have built a framework where feature selection and evaluation are independent of what detector are tested, the test setup is also shared. In this way we reduce complexity and probability for making different logic in duplicated code. Tests are always run from the main script. In the main script we choose one setup and the number of detectors we want to test against the setup. The setup has two parts, feature selection and test setup. Evaluating performance is done in a separate script.

main script pseudo code

```
clear all;
load selected_setup;   %feature selection and setup.

for detector = all_selected_detectors

  execute detector;

  EER = evaluate( DS , DI );
end
```

Setup

The setup is done in one script only, and only if a known setup type is defined in a global variable. If case of an unknown setup type all setup is cleared and it is impossible to start any detector

tests. By doing this we make sure that the setup and data match, and that any other play-around with variables don't make problems for the tests.

We have named two default setups, experiment and EXT.

Experiment Signature 1:20 are used for enrollment, signature 21:60 is used for testing. Signature 21:60 from the impostors are used for attack. In this way we avoid using signatures for both enrollment and attack. This setup is used against all data sets containing 60 signatures per test subject.

EXT EXT is the setup described in [36]. Signature 1:200 are used for template building, signature 201:400 are used for testing, and signature 1:5 from all impostors are used for attack. This setup is only used against the data set EXT.

Feature selection

Feature selection is the process of filling data into our feature table $\mathbf{F_{u,r,c}}$, see definition in section 6.1. We do this in form of one script file for each setup. We have a unified procedure where we (1) load raw data from files, (2)Moving data into the feature table. (3)Choosing setup type. (4) Clear all data loaded in step 1. Some of the main setup scripts are:

PD This is the easiest setup because we have the same structure in our PD data as we want in our feature table. Both latency features and duration features are selected. The setup is set to *experiment*.

SD Extracts all durations and latencies from the smart phone part of the data collection experiment. Setup type is set to *experiment*.

SPD_All Imports all available features into the feature table, except for swipe data. The swipe data contains too many zero values to be useful. Instead of importing all 12 swipe features, we import the mean of all swipe features in each signature. The setup type is set to *experiment*.

EXT_default Loads all features available and set the setup type to "EXT".

EXT_PD This setup is not straightforward because it will be used to compare the EXT data with our PD data. In addition to use only durations and latencies we select only sixty signatures from each of the nineteen users we also select. EXT data has 400 signatures collected over 8 sessions, while we have 60 signatures collected in 20 sessions. We believe we get the closest match by selecting the first 7-8 signatures from each EXT session. The 19 users are selected at random each time the script is run. We utilize this function by making a script that run the main scripts several times and aggregate results each turn, and provide us with a much better material to compare our PD data against. Setup type is set to *experiment*.

PD_SD This is almost the same setup as PD. The difference is that only 10 users are imported. The users are selected randomly. We reuse the script described under EXT_PD to make it possible to compare PC and smart phone results.

In addition to the main setup scripts there are several others e.g. to test only a few features.

Execute detector

The execution of the test is controlled by the detector scripts. The procedure is unified, and illustrated in the pseudo code below.

detector script - pseudo code

```
clear DI; %DI contains impostor distances for all users
clear DS; %DS contains all genuine user distances

for subject = allusers

  T= CalcTemplate(  F( subject, template_signatures ) );

  DS = CalcTestDist( T , F( subject, test_signatures ) );

  for impostor = [allusers - subject]
    DI= CalcImpostorDist( T, F( impostor, impostor_signatures ) )
  end
end
```

As we see the detector scripts create two results DS and DI. DS has one column for each genuine user,which contains all calculated distances between a genuine signatures and the template. DI is a table containing one sub-table for each user. Each sub-table has one column for each impostor, which contains all distances between impostor signatures and the template. In the PD setup, the DS table has 19 columns and 40 rows, and the DI table has 19 sub-tables, where each sub-table contains 18 columns and 40 rows.

Performance

We use equal error rate (EER) as a measure on detector performance. To find EER we need to calculate two other values, false match rate(FMR) and false non match rate(FNMR). Then we have to search for a threshold where EER = FNMR = FMR. In figure2 we can see that one only need to search a certain interval, namely from the minimum distance from impostor signatures and to the maximum distance from test signatures. In some cases the curves does not overlap, then EER=0 and the threshold is any value between the two points. Usually there are a overlap and we need to search for the solution. We could search real values in the given interval, but there are a simpler solution. The optimal threshold is one of the distances we have calculated in DS/DI or any value between two values in DI/DS. This observation reduce the search space to the finite set of values in DS and DI, and greatly simplify the procedure.

The search is done in three main steps. The full listing is in appendix Evaluate Performance

- Sort all the user test distances(DS).
- Perform a binary search on DS. We start with the interval [minimum(DI), maximum(DS)]. We identify the middle value and calculate FNMR/FMR using the value as threshold. Based result we chose one of the halves as the new interval. The search stop when the interval is two consecutive values from DS.
- If there are values from DI in the interval from 2. We perform a linear search to find a optimal

solution, where we use each value as threshold to calculate FMR/FNMR.

The performance is based on calculating a optimal threshold per user. That is not possible in a system perspective, then a threshold is decided in advance and the system performance is measured. Further if testing a system the almost equivalents FAR and FRR is used instead of FMR and FNMR. We believe that testing performance on algorithm level is sufficient to answer or research questions, and also make it possible to compare with the external reference data set.

6.4 Summary

First we sum up the results using all detectors on all main data sets, see table 7. We can see that the values differ both between detectors and between data sets, but we do not know if the differences are significant or not. Up until now we have described both data and results using descriptive statistics, now we will interpret the data and make inferences.

6.4.1 Inferential statistics

Inferential statistics help us based upon our data set. There are two main groups of statistical procedures[55] parametric and nonparametric. Parametric statistics are based on assumptions like, that the scale of data must be an interval or ratio scale, and that the data has a normal distribution. The non parametric statistics are not based on such assumptions, but are generally not as powerful as the parametric statistics. [56] used a paired t-test and assumed that both error rates followed a normal distribution, while [36] found that the error rates did not follow a normal distribution and used the Wilcoxon matched-pair signed rank test. However, before choosing a procedure we need to understand the nature of the EER and *why it is not suitable to be used in any statistical analysis directly*.

	EXT_PD	**PD**	**PD_SD**	**SD**	**SPD_all**
Euclidean	0.296	0.307	0.306	0.277	0.173
EuclideanX	0.346	0.316	0.326	0.294	0.164
Euclidean(s)	0.162	0.206	0.202	0.263	0.121
Euclidean(n)	0.161	0.212	0.211	0.245	0.068
Euclidean(f)	0.286	0.288	0.284	0.269	0.175
Euclidean(fs)	0.165	0.206	0.203	0.262	0.122
Euclidean(fn)	0.162	0.214	0.209	0.245	0.069
Euclidean(n+)	0.154	0.200	0.200	0.229	0.067
Manhattan	0.256	0.254	0.253	0.251	0.124
Manhattan(s)	0.153	0.194	0.192	0.236	0.061
Manhattan(n)	0.153	0.195	0.192	0.239	0.058
Manhattan(f)	0.242	0.229	0.227	0.252	0.124
Manhattan(fs)	0.152	0.194	0.192	0.230	0.060
Manhattan(fn)	0.156	0.196	0.193	0.235	0.058
Manhattan(n+)	0.147	0.182	0.175	0.227	0.059

Table 7: Main performance table. The values shown are equal error rates for each detector against the data sets in the column headers.

Let us use EXT as an example. EXT has 51 test subjects. If we use each test subject as genuine user one time, we end up with a list of 51 EER values per detector. From the list of EER values we can compute an average EER from the first detector. We repeat the same procedure for the second detector, and have two average EER to compare and decide which is best. We can not compare the two values directly, because the draw of 51 test subjects may be a lucky draw for one detector and a bad draw for the other. So, why not use the lists of 51 EER's to make some confidence intervals(CI)? First, the EER in the list is not independent. By replacing only one test subject all EER in the list will change. Secondly, by the definition of standard error $SE(\overline{X}) = \frac{\sigma}{\sqrt{n}}$[57, p.218] it should decrease with the number of test subjects. It does not. Few test subjects will generally give better EER. The more test subjects the more difficult it is to separate them. We have tested both claims on our data, but they are logical assumptions too. Regardless of the distribution of the list of 51 EER values there are a solution.

By the central limit theorem [57, p.184] where a stochastic variable $\overline{X}$ represent the mean of a random selection, the set of n such mean values will have a close to normal distribution when n is high. Each $\overline{X}$ must be from the same probability distribution that may be far from normal distributed.

We consider that our *average EER* is suitable as $\overline{X}$. This make logical sense too. Let's go back to the EXT example. Let the object we observe be the complete test, where average EER is the stochastic variable. For each time we do the experiment with 51 new random test subjects we improve our estimate of the mean, and the standard deviation. Thus we are able to make a CI. The discussion so far shows that we can and should choose a parametric statistical procedure. But obviously, we cannot do our complete experiment many times, and we have to approximate.

We use the EXT_PD to estimate the mean and standard deviation that we can use as basis to make a CI. We select 19 random test subjects 100 times. From figure 20 a) we can see that the distribution are close to normal. While the distribution for the individual EER is far from normal b). By using CI and one-sample T-test it is possible to decide if the one PD value and the mean of EXT_PD is significantly different.

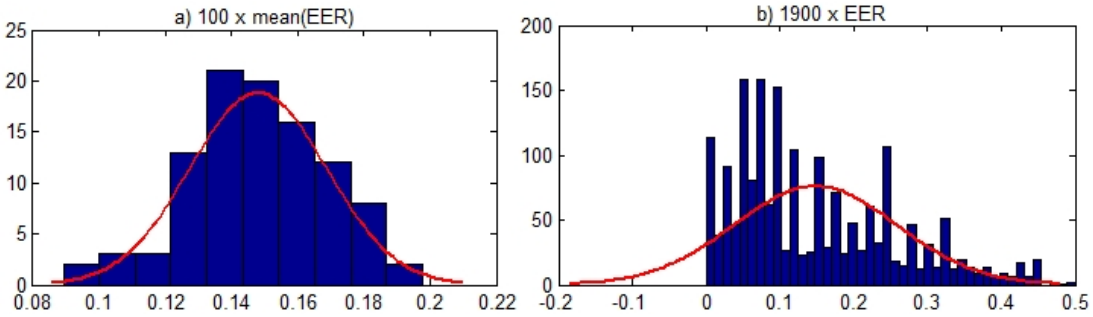

Figure 20: The effect of the central limit theorem [57, p.184] is shown in the figures. We test the Manhattan(n+) detector 100 times, using 19 random selected test subject from the 51 in EXT[35]. a) show the histplot for mean EER for each test, and b)show the histplot for the 1900 EER values calculated during the tests. The mean is of course the same in both a) and b), but in a) the standard deviation 0.0207, and in b) the standard deviation is 0.111.

The statistical procedure[57] is illustrated by an example, using the Manhattan(n+) to compare EXT_PD and PD from table 7. X is the 100 mean(EER) from EXT_PD where μ is the unknown mean, and μ_0 is the expected mean(EER) from PD. We have already seen that X is close to normal distributed, we assume that the values in X are independent of each other and have the expected value $\overline{X}$. We do not know the true standard deviation and estimate it using X, and refer to it as s. We want to test the following hypothesis:

$$H_0 : \mu = \mu_0$$
$$H_1 : \mu \neq \mu_0$$

In our hypothesis test we can do two types of error. If we reject H_0 when it is true, we do a type I error), or if we keep H_0 when H_1 is true, we do a type II error. In most situations a type I error is the most serious. Thus we must decide an acceptable probability to do errors of type I. We refer to this probability as the significance level, α. In our test we use the commonly used 5%, $\alpha = 0.05$.

We have estimated the standard deviation and need to address the uncertainty in our estimate. We do this by using the student T distribution, and we reject the H_0 hypothesis if $|T| > t_{\alpha/2}$. Equation (6.11) show how to calculate our T value.

$$T = \frac{\overline{X} - \mu_0}{s\sqrt{n}} = \frac{0.147 - 0.182}{0.024\sqrt{100}} = -14.5 \tag{6.11}$$

We find the critical t-value $t_{\alpha/2} = 1.984$ in a t-distribution table using the degree of freedom $df = n - 1 = 99$. Because $|T| > t_{\alpha/2} => 14.4 > 1.984$ we reject the H_0 hypothesis. By calculating the CI(6.12) we also see that $\mu_0 = 0.182$ this outside this interval.

$$\left[\overline{X} - t_{\alpha/2} \cdot \frac{s}{\sqrt{n}}, \overline{X} + t_{\alpha/2} \cdot \frac{s}{\sqrt{n}}\right] = [0.1361, 0.1508] \tag{6.12}$$

From this example we can conclude that the detector manhattan(n+) perform, statistically, significantly worse on the PD data set compared to on the EXT_PD data set. We will discuss possible reasons for this during our discussion in the next chapter.

6.4.2 Data sets

The described procedure is applied to compare EXT_PD and PD in table 8, PD_SD and SD in table 9, and PD_SD and SD_ALL in table **??**. Another useful value when interpreting our results are the p-value. The p-value are related to our H_0 and is the probability of making an error of type I, if we are going to reject H_0 with the data present. Another common definition of the p-value given by [57], is that the p-value are the lowest value of α that will make us to reject H_0 given our observed data. in the mentioned tables the p-values are included, but for simplicity we have calculated the closest value to the p-value where H_0 is not rejected. For example, if our H_0 is on the form $H_0 : \mu \geq \mu_0$, then we reject the hypothesis when $T > t_\alpha$. Our α is a real number in the interval $[0, 1]$ thus the difference is about precision/rounding and not important because the p-value will only be used for human interpretation. E.g. add or subtract 0.000001.

EXT_PD - PD	CI	PD	T	$\|T\| > t_{\alpha/2}$	$T > t_{\alpha}$	$T < t_{\alpha}$
Euclidean	[0.292 ,0.3]	0.307	-4.82	T(0.0000)	F(1.0000)	T(0.0000)
EuclideanX	[0.342 ,0.351]	0.316	13.2	T(0.0000)	T(0.0000)	F(1.0000)
Euclidean(s)	[0.158 ,0.167]	0.206	-19.6	T(0.0000)	F(1.0000)	T(0.0000)
Euclidean(n)	[0.157 ,0.165]	0.212	-24.4	T(0.0000)	F(1.0000)	T(0.0000)
Euclidean(f)	[0.281 ,0.29]	0.288	-1.11	F(0.2708)	F(0.8646)	F(0.1354)
Euclidean(fs)	[0.16 ,0.17]	0.206	-17.2	T(0.0000)	F(1.0000)	T(0.0000)
Euclidean(fn)	[0.157 ,0.167]	0.214	-21	T(0.0000)	F(1.0000)	T(0.0000)
Euclidean(n+)	[0.15 ,0.158]	0.2	-21.6	T(0.0000)	F(1.0000)	T(0.0000)
Manhattan	[0.251 ,0.261]	0.254	0.671	F(0.5038)	F(0.2519)	F(0.7481)
Manhattan(s)	[0.148 ,0.157]	0.194	-17.8	T(0.0000)	F(1.0000)	T(0.0000)
Manhattan(n)	[0.148 ,0.158]	0.195	-16.8	T(0.0000)	F(1.0000)	T(0.0000)
Manhattan(f)	[0.237 ,0.247]	0.229	5.6	T(0.0000)	T(0.0000)	F(1.0000)
Manhattan(fs)	[0.148 ,0.157]	0.194	-18.3	T(0.0000)	F(1.0000)	T(0.0000)
Manhattan(fn)	[0.152 ,0.16]	0.196	-18.5	T(0.0000)	F(1.0000)	T(0.0000)
Manhattan(n+)	[0.143 ,0.152]	0.182	-14.5	T(0.0000)	F(1.0000)	T(0.0000)

Table 8: CI is the confidence interval, using 100 samples of EXT_PD and a significance level $\alpha = 0.05$. From the t-distribution quartile table $t_{\alpha} = 1.66$ and $t_{\alpha/2} = 1.984$. The three rightmost columns show when to reject the respective hypothesis $\mu = \mu_0$, $\mu \leq \mu_0$ and $\mu \geq \mu_0$. T=true=reject H_0. The number in brackets is the p-value.

6.4.3 Detectors

In table 12 and 13 we show the best performing detectors on EXT_PD and PD_SD. To select detectors we defined a H_0 hypothesis that all detectors has equal performance. Then we selected those detector that had a p-value against Manhattan(n+) i.e., there exist a possibility for doing an error of type I. This procedure resulted in the same detectors for both tables.

PD_PD -SD	CI	SD	T	$\|T\| > t_{\alpha/2}$	$T > t_{\alpha}$	$T < t_{\alpha}$
Euclidean	[0.298 ,0.314]	0.277	7.07	T(0.0000)	T(0.0000)	F(1.0000)
EuclideanX	[0.316 ,0.337]	0.294	6.23	T(0.0000)	T(0.0000)	F(1.0000)
Euclidean(s)	[0.193 ,0.21]	0.263	-14	T(0.0000)	F(1.0000)	T(0.0000)
Euclidean(n)	[0.203 ,0.219]	0.245	-8.53	T(0.0000)	F(1.0000)	T(0.0000)
Euclidean(f)	[0.273 ,0.294]	0.269	2.93	T(0.0052)	T(0.0026)	F(0.9974)
Euclidean(fs)	[0.194 ,0.211]	0.262	-14.1	T(0.0000)	F(1.0000)	T(0.0000)
Euclidean(fn)	[0.201 ,0.218]	0.245	-8.73	T(0.0000)	F(1.0000)	T(0.0000)
Euclidean(n+)	[0.191 ,0.21]	0.229	-6.32	T(0.0000)	F(1.0000)	T(0.0000)
Manhattan	[0.244 ,0.261]	0.251	0.502	F(0.6181)	F(0.3090)	F(0.6910)
Manhattan(s)	[0.185 ,0.2]	0.236	-11.8	T(0.0000)	F(1.0000)	T(0.0000)
Manhattan(n)	[0.185 ,0.2]	0.239	-12	T(0.0000)	F(1.0000)	T(0.0000)
Manhattan(f)	[0.219 ,0.236]	0.252	-5.86	T(0.0000)	F(1.0000)	T(0.0000)
Manhattan(fs)	[0.183 ,0.201]	0.23	-8.77	T(0.0000)	F(1.0000)	T(0.0000)
Manhattan(fn)	[0.185 ,0.202]	0.235	-9.64	T(0.0000)	F(1.0000)	T(0.0000)
Manhattan(n+)	[0.166 ,0.183]	0.227	-12.7	T(0.0000)	F(1.0000)	T(0.0000)

Table 9: CI is the confidence interval, using 50 samples of PD_SD and a significance level $\alpha = 0.05$. From the t-distribution quartile table $t_{\alpha} = 1.677$ and $t_{\alpha/2} = 2.01$. The three rightmost columns show when to reject the respective hypothesis $\mu = \mu_0$, $\mu \leq \mu_0$ and $\mu \geq \mu_0$. T=true=reject H_0. The number in brackets is the p-value.

PD_PD -SD_All	CI	SD_All	T	$\|T\| > t_{\alpha/2}$	$T > t_{\alpha}$	$T < t_{\alpha}$
Euclidean	[0.298 ,0.314]	0.173	32.7	T(0.0000)	T(0.0000)	F(1.0000)
EuclideanX	[0.316 ,0.337]	0.164	31	T(0.0000)	T(0.0000)	F(1.0000)
Euclidean(s)	[0.193 ,0.21]	0.121	18.3	T(0.0000)	T(0.0000)	F(1.0000)
Euclidean(n)	[0.203 ,0.219]	0.0681	35.6	T(0.0000)	T(0.0000)	F(1.0000)
Euclidean(f)	[0.273 ,0.294]	0.175	21.3	T(0.0000)	T(0.0000)	F(1.0000)
Euclidean(fs)	[0.194 ,0.211]	0.122	19.1	T(0.0000)	T(0.0000)	F(1.0000)
Euclidean(fn)	[0.201 ,0.218]	0.0689	34.2	T(0.0000)	T(0.0000)	F(1.0000)
Euclidean(n+)	[0.191 ,0.21]	0.0672	28.9	T(0.0000)	T(0.0000)	F(1.0000)
Manhattan	[0.244 ,0.261]	0.124	32	T(0.0000)	T(0.0000)	F(1.0000)
Manhattan(s)	[0.185 ,0.2]	0.0614	35.4	T(0.0000)	T(0.0000)	F(1.0000)
Manhattan(n)	[0.185 ,0.2]	0.0581	34.8	T(0.0000)	T(0.0000)	F(1.0000)
Manhattan(f)	[0.219 ,0.236]	0.124	24.6	T(0.0000)	T(0.0000)	F(1.0000)
Manhattan(fs)	[0.183 ,0.201]	0.06	30	T(0.0000)	T(0.0000)	F(1.0000)
Manhattan(fn)	[0.185 ,0.202]	0.0581	31.6	T(0.0000)	T(0.0000)	F(1.0000)
Manhattan(n+)	[0.166 ,0.183]	0.0586	28.3	T(0.0000)	T(0.0000)	F(1.0000)

Table 10: CI is the confidence interval, using 50 samples of PD_SD and a significance level $\alpha = 0.05$. From the t-distribution quartile table $t_{\alpha} = 1.677$ and $t_{\alpha/2} = 2.01$. The three rightmost columns show when to reject the respective hypothesis $\mu = \mu_0$, $\mu \leq \mu_0$ and $\mu \geq \mu_0$. T=true=reject H_0. The number in brackets is the p-value.

EXT_RND -PD	CI	PD	T	$\|T\| > t_{\alpha/2}$	$T > t_{\alpha}$	$T < t_{\alpha}$
Euclidean	[0.321 ,0.331]	0.307	7.98	T(0.0000)	T(0.0000)	F(1.0000)
EuclideanX	[0.371 ,0.38]	0.316	27.2	T(0.0000)	T(0.0000)	F(1.0000)
Euclidean(s)	[0.204 ,0.214]	0.206	1.42	F(0.1597)	F(0.0798)	F(0.9202)
Euclidean(n)	[0.201 ,0.21]	0.212	-2.45	T(0.0159)	F(0.9920)	T(0.0080)
Euclidean(f)	[0.311 ,0.32]	0.288	12	T(0.0000)	T(0.0000)	F(1.0000)
Euclidean(fs)	[0.202 ,0.213]	0.206	0.589	F(0.5570)	F(0.2785)	F(0.7215)
Euclidean(fn)	[0.202 ,0.212]	0.214	-2.95	T(0.0039)	F(0.9980)	T(0.0020)
Euclidean(n+)	[0.195 ,0.205]	0.2	0.0332	F(0.9735)	F(0.4868)	F(0.5132)
Manhattan	[0.283 ,0.292]	0.254	13.6	T(0.0000)	T(0.0000)	F(1.0000)
Manhattan(s)	[0.191 ,0.201]	0.194	0.84	F(0.4027)	F(0.2013)	F(0.7987)
Manhattan(n)	[0.189 ,0.2]	0.195	-0.366	F(0.7149)	F(0.6426)	F(0.3574)
Manhattan(f)	[0.273 ,0.283]	0.229	18	T(0.0000)	T(0.0000)	F(1.0000)
Manhattan(fs)	[0.188 ,0.196]	0.194	-0.874	F(0.3844)	F(0.8078)	F(0.1922)
Manhattan(fn)	[0.187 ,0.197]	0.196	-1.7	F(0.0920)	F(0.9540)	T(0.0460)
Manhattan(n+)	[0.189 ,0.199]	0.182	4.62	T(0.0000)	T(0.0000)	F(1.0000)

Table 11: CI is the confidence interval, using 100 samples of EXT_RND and a significance level $\alpha = 0.05$. From the t-distribution quartile table $t_{\alpha} = 1.66$ and $t_{\alpha/2} = 1.984$. The three rightmost columns show when to reject the respective hypothesis $\mu = \mu_0$, $\mu \leq \mu_0$ and $\mu \geq \mu_0$. T=true=reject H_0. The number in brackets is the p-value.

	CI	1	2	3	4	5	6
1)Euclidean(n+)	[0.1501,0.1584]		64.5	63.7	49.6	59.4	3.42
2)Manhattan(s)	[0.1483,0.1573]	64.5		97.1	83.2	33.6	10.7
3)Manhattan(n)	[0.1476,0.1577]	63.7	97.1		87.0	34.4	13.5
4)Manhattan(fs)	[0.1476,0.1567]	49.6	83.2	87.0		23.9	16.0
5)Manhattan(fn)	[0.1515,0.1602]	59.4	33.6	34.4	23.9		1.02
6)Manhattan(n+)	[0.1427,0.1522]	3.42	10.7	13.5	16.0	1.02	

Table 12: Compare the detectors with best performance on the EXT_PD data set, using a two sample t-test assuming equal but unknown variance. The column headers corresponds to numbers in front of each detector name. The H_0 hypothesis is that the two detectors has the same mean, and the numbers in table are p-values in percent. An example: we want to compare detector 6 against detector 1. From the table we find the value 3.42. This mean that there is a 3.4% probability to make a type I error if we reject H_0. Thus, with a significance level $\alpha = 0.05$ we reject H_0.

	CI	**1**	**2**	**3**	**4**	**5**	**6**
1)Euclidean(n+)	[0.1911,0.2096]		17.6	19.0	18.0	27.1	
2)Manhattan(s)	[0.1849,0.1997]	17.6		98.1	92.6	84.8	0.19
3)Manhattan(n)	[0.1847,0.2002]	19.0	98.1		91.0	86.8	0.22
4)Manhattan(fs)	[0.1829,0.2006]	18.0	92.6	91.0		79.2	0.55
5)Manhattan(fn)	[0.1848,0.202]	27.1	84.8	86.8	79.2		0.22
6)Manhattan(n+)	[0.1665,0.1829]		0.19	0.22	0.55	0.22	

Table 13: Compare the detectors with best performance on the PD_SD data set, using a two sample t-test assuming equal but unknown variance. The column headers corresponds to numbers in front of each detector name. The H_0 hypothesis is that the two detectors has the same mean, and the numbers in table are p-values in percent. An example: we want to compare detector 4 against detector 1. From the table we find the value 18.0. This mean that there is a 18% probability to make a type I error if we reject H_0. Thus, with a significance level $\alpha = 0.05$ we keep H_0.

7 Discussion

This chapters covers discussions on methodology used, performance, security and research questions.

7.1 Research work

Leedy and Ormrod[55] describe four general criteria every research project should consider during planning, universality, replication, control and measurement.

Universality is that any competent person should be able to take your place and complete the project with essentially the same results. This implies that research are planned and that it describes the methodology used to answer the research questions. Our approach have been to establish a coarse plan and coarse description of methodology, see chapter 2. Details and technicalities are solved during research. The balance of what is detail and what is not is always dependant of researcher. We believe our approach is good because any competent person would have completed the research with the same answers, but flexible enough to encourage the researcher always to look for options and new ideas that may open new doors.

Replication is that the research should be repeatable by any other competent researcher, and have comparable results. In our research this is solved by documenting every important step from research question to conclusion, and not rely on any specialized skills or rare events. By following our described methodology and setup any competent researcher should end up with the same result. Differences in performance is almost unavoidable due to different participants in the experiment, but the performance itself is not important to answer our questions. And by following or procedure differences should be detected and explained. Findings outside the research questions would hopefully be different that our and prove our thinking i.e. enforce control to achieve valid conclusions and not unnecessary control to allow additional findings.

Measurement is that the data should be possible to measure in a consistent way. In research like our the measurement issues are small, we need to check resolution and accuracy of timing data. In addition it is important to understand the variables used in analysis to be sure we analysis what we intend to. We have used a external reference data set, both to test methods on and to compare results against. It has proved to be a good choice because we have detected and corrected many bugs in our setup by doing so.

7.2 Performance

To understand performance i.e., what factors do influence on performance, is necessary to be able to answer our research questions. This was one of our main reasons to choose two simple statistical based detectors, and also to use an external reference data set[35], which we refer to as EXT. We successfully replicated the results of [36], by using the selected detectors, the Euclidean and Manhattan.

Euclidean and Manhattan is different in only one way i.e., the norm of general Minkowski distance. The Euclidian square the distance of each dimension and the Manhattan do not. This has consequences. Consider the Euclidean. Large distances in few dimensions will influence much more on the total distance, that medium distances in many dimensions. This is a wanted property on stable data where few deviations from the template identifies impostors. Unfortunately most typists has larger variation in at least some features. When using the Manhattan, any distance counts equally.

Different variance across features also influence on the performance. If one feature has larger variance than the other it may obscure smaller variations in other features. Outliers are extreme cases, but it do not need to be outlier to have this effect. Filtering has proved to make a more stable template and scaling even out the difference in variance across features and make them count almost equally to the distance. We also included a scaling of the data, where we scaled each feature by the standard deviation from the template. By combining filtering and scaling in new combinations we achieved improved performance on the EXT data set.

Skewness in data has been a concern also in previous research. We found a new way to handle the skewness. By splitting the feature vector in two halves we are able to compute two standard deviations, one for each part. By using this technique to, scale data, in a Manhattan detector we achieved our best performance. We had two data sets suitable to compare detector performance statistically. The EXT_PD and PD_SD data sets, which both were generated using a bootstrapping[58] technique. We included any detector that has a lower p-value than 100 when compared against our best detector. On both data sets we ended up with the same six detectors. When using significance level 5% and the data set EXT_PD our technique was not significantly better than Manhattan(s) (i.e., the p-value was 3.42). On the PD_SD data set Manhattan performed significant better that the other detectors(i.e., highest p-value was 0.6%).

We compare performance between data sets to answer our research questions. To do this we need to rule out confounding variables regarding the data sets. Because of the limited number of participants we compare our data against EXT or more precise the bootstrap generated EXT_PD. If data is collected under equal conditions results from EXT_PD and PD should be equal, it is not. There are several factors that explain the difference. In EXT_PD an alphanumeric signature is used, while we have used a numerical signature. Using both hands is known to discriminate users better than using only one hand[37]. Learning also influence on performance. We tried to eliminate this in the EXT_PD by using only the first signatures from each session. Outliers were analyzed and seems to have an equally occurrence and spread in the data sets. The resolution is found to be sufficient good to not influence significantly on the data, but half of the test subject in the smart phone experiment has 18-22ms resolution on durations which may explain for some of the poor performance.

7.3 Security Discussion

We have built a device to test how easy it is to imitate someone's typing rhythm. The results show that the imitator has 100% success in imitating the victims. On the other hand none of the victims has success in imitating their own template i.e., the ImitatorUno. During the imitator testing we did some interesting discoveries. Some ImitatorUno profiles unintentionally could imitate other

users with 100% accuracy. By having a closer look it reveals that extreme stable typists likely perform better that unstable typists against the unstable typists template. By using two threshold this effect may be reduced, as identified in our analysis. A method like the one used in [32] based on the individual ordering i.e.,relative timing values would also solved such cases.

The ImitatorUno was planned, designed, built and programmed in less than a week. Easily available components was used. The components probably have higher specifications than needed, resulting in a total cost of the prototype a little less than NOK3000,- . Thus, it is doable for any private person really interested.

It is probably possible to make an imitator for a smart phone as well. The cost is probably much higher because of miniature components. The complexity due to features like pressure, movement and position on key is also much higher. We have not been able to design any ideas solving this complexity, but persons with different background may come up with a viable solution.

In our experiment we have used the template to build the individual profiles. The template should be protected like passwords so the ImitatorUno is useless if only based on the template. Another and probably better way of building a profile is to collect signatures directly. If it is possible it could be used to imitate the real variance in the typing too, and make protection against imitation far more difficult. Other sources are available, [38] used sound to indirectly detect both timing and pressure. Using video as shoulder surfing technique would probably also be sufficient to uncover a users keystroke dynamics biometrics. The smart phone is more resilient to these attacks of several reasons. The location while used is changing, thus more unpredictable where to mount audio and video recording equipment. The movements when typing on a smart phone is more less that the bigger keyboard.

Software attacks are outside the scope, but smart phones do have a vulnerability that is worth mentioning. The built-in sensors can be misused to spy on the owner and reveal also biometrics. One example of this is [44] that implemented a trojan horse that exploited the sensors to steal credit card information. Motion sensors can easily be used to reveal keystroke characteristics [43]. Keyboards are also vulnerable to attacks. [50] show that it is possible to obtain keystrokes from a distance of 20 meters using the keyboard's electromagnetic emanation. Since a smart phone is a electronic device it is probably vulnerable for such attacks too.

Attacks using a physical device is difficult(impossible?) to detect by software, hence protection should be implemented through physical protection. In the future that may not be enough. Today muscles are stimulated using electronic probes, brain activity in monitored, hearing devices are implanted and the brain can learn how to use is to recognize sound. What if it is possible to mount something on the wrist capable of stimulate the muscles to imitate someone's typing?

7.4 Research questions

We have discussed performance and during data analysis we are confident that our data is suitable to answer the research questions.

When comparing PD_SD against SD, the plain Manhattan performed equally good, three Euclidean detectors performed better on SD, while the rest performed best on PD_SD. The tests were done at significance level 5%. Since all the best performing detectors performed best on

PD_SD it is statistically significant better performance using keystroke dynamics on a standard keyboard compared to keystroke dynamics on a smart phone using the touch screen as input. However scientific it is not equally clear. The Euclidian had better result on the smart phone, maybe another kind of detectors are more suitable on the smart phone. Therefor we must include the condition using an Manhattan detector to our answer.

When it comes to using all available features on the smart phone, all detectors performed much better on a smart phone than on a standard keyboard, statistically. The p-values indicate that the statistical possibility to make an error type I is zero, for all detectors. We must add a condition there too. We have identified possible bias from smart phone brand that may have contributed to the good performance. Giving the small data set we have not been able to determine exactly how big the bias is.

Performance can never be evaluated generally. We have discussed certain vulnerabilities and have not found that using keystroke dynamics on a smart phone results in worse security. We stick with such conservative formulation even if we have demonstrated and discussed that several vulnerabilities are greater using a standard keyboard.

8 Conclusion

Findings from using keystroke dynamics on a smart phone using the touch screen as input is promising. Given a small test group and using the extra features from the touch screen it is statistical significant better performance using keystroke dynamics on a smart phone than on a standard keyboard. However, because of the small test group we cannot exclude the possibility of bias from phone model and brand. When not using the extra features from the touch screen, the performance on the smart phone is worse than on a standard keyboard.

Previous research have claimed that imitating someone is difficult. We have built and tested a device that makes it easy to mimic someone's typing characteristics on a standard keyboard. The same vulnerability is less on a smart phone. Both because the biometrics is less exposed when using a smart phone compared to when using a standard keyboard and because it is less complex to build a helpful attack devices for use on a standard keyboard than it is for a smart phone. When we balance this with findings in the literature we can safely say that keystroke dynamics on a smart phone is not less secure that using keystroke dynamics on a standard keyboard. But it is definitive easier to imitate someone on a standard keyboard than it is on a smart phone.

Our findings then support the following answer to our research question: keystroke dynamics on a smart phone is at least as suitable as keystroke dynamics in on a standard keyboard.

9 Future work

Variability in performance due to test group composition is significant and need more attention. We should value variability in data to enable us to describe and understand techniques before optimizing performance without understanding why a method work. This implies that properties of keystroke dynamics should be categorized and explained. Specially analysis on lower levels as distance contribution on feature level.

We must most likely coop with unstable typists in the future too. Use of dual threshold and combination of methods should be investigated. This could be done in a research looking at how the individual feature contributes to the total distance. Such work implies categorize distribution types and optimize ways to handle the different categories.

Last but certainly not less important. Make data sets public available to enable more people to look at challenges and to compare results. After all the data collection activity limit the amount of time used in analysis and creative work.

Bibliography

[1] Miller, B. feb. 1994. Vital signs of identity [biometrics]. *Spectrum, IEEE*, 31(2), 22 –30.

[2] Conklin, A., Dietrich, G., & Walz, D. jan. 2004. Password-based authentication: a system perspective. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, 10 pp.

[3] Crawford, H. aug. 2010. Keystroke dynamics: Characteristics and opportunities. In *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*, 205 –212.

[4] Jain, A. K., Ross, A., & Prabhakar, S. jan. 2004. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1), 4 – 20.

[5] VIJAY DHIR, AMARPREET SINGH, R. K. & SINGH, G. 2010. Biometric recognition: A modern era for security. *International Journal of Engineering Science and Technology*, 2, 3364–3380.

[6] Bishop, M. 2002. *Computer Security: Art and Science*. Addison-Wesley Professional; 1 edition (December 12, 2002).

[7] Joyce, R. & Gupta, G. February 1990. Identity authentication based on keystroke latencies. *Commun. ACM*, 33, 168–176.

[8] O'Gorman, L. dec 2003. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12), 2021 – 2040.

[9] Gafurov, D., Snekkenes, E., & Bours, P. sept. 2007. Spoof attacks on gait authentication system. *Information Forensics and Security, IEEE Transactions on*, 2(3), 491 –502.

[10] Rundhaug, F. E. N. Keystroke dynamics: Can attackers learn someones typing characteristics. Master's thesis, Gjøvik University College, 2007.

[11] Karnan, M., Akila, M., & Krishnaraj, N. 2011. Biometric personal authentication using keystroke dynamics: A review. *Applied Soft Computing*, 11(2), 1565 – 1573. <ce:title>The Impact of Soft Computing for the Progress of Artificial Intelligence</ce:title>.

[12] Gaines, R. S., Lisowski, W., Press, S. J., & Shapiro, N. Authentication by keystroke timing: Some preliminary results. Technical report, RAND CORP SANTA MONICA CA, 1980.

[13] Bleha, S., Slivinsky, C., & Hussien, B. dec 1990. Computer-access security systems using keystroke dynamics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(12), 1217 –1222.

[14] Monrose, F. & Rubin, A. 1997. Authentication via keystroke dynamics. In *Proceedings of the 4th ACM conference on Computer and communications security*, CCS '97, 48–56, New York, NY, USA. ACM.

[15] Young, J. R. & Hammon, R. W. 1989. Method and apparatus for verifying an induvidual's identity. u.s. patent 4,805,222.

[16] Garcia, J. D. 1986. Personal identification apparatus. u.s. patent 4,621,334.

[17] Mahar, D., Napier, R., Wagner, M., Laverty, W., Henderson, R. D., & Hiron, M. October 1995. Optimizing digraph-latency based biometric typist verification systems: inter and intra typist differences in digraph latency distributions. *Int. J. Hum.-Comput. Stud.*, 43(4), 579–592.

[18] Loy, C. C., Lai, W. K., & Lim, C. P. nov. 2007. Keystroke patterns classification using the artmap-fd neural network. In *Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007. Third International Conference on*, volume 1, 61 –64.

[19] Eltahir, W. E., Salami, M.-J. E., Ismail, A. F., & Lai, W. K. 2008. Design and evaluation of a pressure-based typing biometric authentication system. *EURASIP J. Information Security*, –1–1.

[20] Zahid, S., Shahzad, M., Khayam, S. A., & Farooq, M. 2009. Keystroke-based user identification on smart phones. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, 224–243, Berlin, Heidelberg. Springer-Verlag.

[21] Saevanee, H. & Bhattarakosol, P. jan. 2009. Authenticating user using keystroke dynamics and finger pressure. In *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, 1 –2.

[22] Epp, C., Lippold, M., & Mandryk, R. L. 2011. Identifying emotional states using keystroke dynamics. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, 715–724, New York, NY, USA. ACM.

[23] Peacock, A., Ke, X., & Wilkerson, M. September 2004. Typing patterns: A key to user identification. *IEEE Security and Privacy*, 2, 40–47.

[24] Shanmugapriya, D. & Padmavathi, G. 2009. A survey of biometric keystroke dynamics: Approaches, security and challenges. *International Journal of Computer Science and Information Security,IJCSIS,September 2009, USA*, Vol. 5, No. 1, 115–119.

[25] Monrose, F., Reiter, M. K., & Wetzel, S. 1999. Password hardening based on keystroke dynamics. In *Proceedings of the 6th ACM conference on Computer and communications security*, CCS '99, 73–82, New York, NY, USA. ACM.

[26] joo Lee, H. & Cho, S. 2007. Retraining a keystroke dynamics-based authenticator with impostor patterns. *Computers & Security*, 26(4), 300 – 310.

[27] Araujo, L., Sucupira, L.H.R., J., Lizarraga, M., Ling, L., & Yabu-Uti, J. feb. 2005. User authentication through typing biometrics features. *Signal Processing, IEEE Transactions on*, 53(2), 851 – 855.

[28] Balagani, K. S., Phoha, V. V., Ray, A., & Phoha, S. 2011. On the discriminability of keystroke feature vectors used in fixed text keystroke authentication. *Pattern Recognition Letters*, 32(7), 1070 – 1080.

[29] Robinson, J., Liang, V., Chambers, J., & MacKenzie, C. mar 1998. Computer user verification using login string keystroke dynamics. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 28(2), 236 –241.

[30] Hocquet, S., Ramel, J., & Cardot, H. 0-0 2006. Estimation of user specific parameters in one-class problems. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, 449 –452.

[31] Hocquet, S., Ramel, J.-Y., & Cardot, H. oct. 2005. Fusion of methods for keystroke dynamic authentication. In *Automatic Identification Advanced Technologies, 2005. Fourth IEEE Workshop on*, 224 – 229.

[32] Bergadano, F., Gunetti, D., & Picardi, C. November 2002. User authentication through keystroke dynamics. *ACM Trans. Inf. Syst. Secur.*, 5, 367–397.

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

[34] Montalvão Filho, J. R. & Freire, E. O. October 2006. On the equalization of keystroke timing histograms. *Pattern Recogn. Lett.*, 27, 1440–1446.

[35] Killourhy, K. & Maxion, R. Keystroke dynamics - benchmark data set. http://www.cs.cmu.edu/ keystroke/ (last accessed march 2012). Comparing Anomaly-Detection Algorithms for Keystroke Dynamics" (DSN-2009).

[36] Killourhy, K. & Maxion, R. 29 2009-july 2 2009. Comparing anomaly-detection algorithms for keystroke dynamics. In *Dependable Systems Networks, 2009. DSN '09. IEEE/IFIP International Conference on*, 125 –134.

[37] Ord T, F. S. 2000. User authentication for keypad-based devices using keystroke analysis. In *Proceedings of the Second International Network Conference (INC 2000), Plymouth, UK, pp263-272, 3-6 July 2000*.

[38] Nguyen, T. T., Le, T. H., & Le, B. H. 2010. Keystroke dynamics extraction by independent component analysis and bio-matrix for user authentication. In *Proceedings of the 11th Pacific Rim international conference on Trends in artificial intelligence*, PRICAI'10, 477–486, Berlin, Heidelberg. Springer-Verlag.

[39] Hosseinzadeh, D. & Krishnan, S. nov. 2008. Gaussian mixture modeling of keystroke patterns for biometric applications. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(6), 816 –826.

[40] Killourhy, K. & Maxion, R. 2008. The effect of clock resolution on keystroke dynamics. In *Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, RAID '08, 331–350, Berlin, Heidelberg. Springer-Verlag.

[41] seob Hwang, S., joo Lee, H., & Cho, S. 2009. Improving authentication accuracy using artificial rhythms and cues for keystroke dynamics-based authentication. *Expert Systems with Applications*, 36(7), 10649 – 10656.

[42] seob Hwang, S., Cho, S., & Park, S. 2009. Keystroke dynamics-based authentication for mobile devices. *Computers & Security*, 28(1-2), 85 – 93.

[43] Cai, L. & Chen, H. 2011. Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX conference on Hot topics in security*, HotSec'11, 9–9, Berkeley, CA, USA. USENIX Association.

[44] Schlegel, R., Zhang, K., yong Zhou, X., Intwala, M., Kapadia, A., & Wang, X. 2011. Soundcomber: A stealthy and context-aware sound trojan for smartphones. In *NDSS,Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February- 9th February 2011*.

[45] Cai, L., Machiraju, S., & Chen, H. 2009. Defending against sensor-sniffing attacks on mobile phones. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, MobiHeld '09, 31–36, New York, NY, USA. ACM.

[46] Serwadda, A., Phoha, V. V., & Kiremire, A. 2011. Using global knowledge of users' typing traits to attack keystroke biometrics templates. In *Proceedings of the thirteenth ACM multimedia workshop on Multimedia and security*, MM&Sec '11, 51–60, New York, NY, USA. ACM.

[47] Stefan, D. & Yao, D. oct. 2010. Keystroke-dynamics authentication against synthetic forgeries. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*, 1 –8.

[48] Davida, G., Frankel, Y., & Matt, B. may 1998. On enabling secure applications through off-line biometric identification. In *Security and Privacy, 1998. Proceedings. 1998 IEEE Symposium on*, 148 –157.

[49] Schneier, B. August 1999. Inside risks: the uses and abuses of biometrics. *Commun. ACM*, 42, 136–.

[50] Vuagnoux, M. & Pasini, S. 2009. Compromising electromagnetic emanations of wired and wireless keyboards. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, 1–16, Berkeley, CA, USA. USENIX Association.

[51] Clarke, N. & Furnell, S. 2005. Authentication of users on mobile telephones: A survey of attitudes and practices. *Computers & Security*, 24(7), 519 – 527.

[52] Kowalski, S. & Goldstein, M. 2006. Consumers'Awareness of, Attitudes Towards and Adoption of Mobile Phone Security. *HFT'06*.

[53] Arduino. accessed 6th of march 2012. Arduino uno. http://arduino.cc/en/Main/ArduinoBoardUno.

[54] Duda, R. O., Hart, P. E., & Stork, D. G. 2001. *Pattern Classification, 2 edition*. Wiley-Interscience, ISBN-13: 978-0471056690.

[55] Leedy, P. D. & Ormrod, J. E. 2005. *Practical research : planning and design, 8th edition*. Pearson Education.

[56] Cho, S., Han, C., Han, D. H., & il Kim, H. 2000. Web based keystroke dynamics identity verification using neural network. *Journal of Organizational Computing and Electronic Commerce*, 10, 295–307.

[57] Løvås, G. G. 2004. *Statistikk for universiteter og høgskoler, 2..utgave*. Universitetsforlaget.

[58] Theodoridis, S. & Koutroumbas, K. 2006. *Pattern recognition, third edition*. Academic press, Elsevier.

A Experiment data

write something about collection - rearranging of data. Data from android/PC are stored in files. One file for each user, and one file containing meta-data for all users participating in the experiment.

Anonymization and assigning new ID's

We end up with following files, all with a heading row and where columns are separated by semicolon.

ID-pc.data One file for each user. The user is assigned an unique ID. The file contains timing information from experiment. The file contains 60 signatures.

ID-sp.data One file for each user. The user is assigned an unique ID. If same user participate in both experiments, the same ID is used for both files. The file contains the same information as in the pc file. In addition it contains more features, like key pressure, size, movement on touch screen and device movement data.

meta.data One file for all users. For each ID it contains information like, gender, age, left/right handed, skill and device/pc information.

user.data The file contains User information associated with each ID like, Name, phone number, email address. The file is not kept apart from other data files, it is only used for administrative purposes and are deleted when the experiment data is assembled and ready for use.

A.1 pc.data

Each row in file contains data from one sample.

A.2 sp.data

Each row in file contains data from one sample.

A.3 meta.data

The table below contains experiment data about the participants. One row for each participant.

Fieldname	Description
se	Session number%
nr	Sample number within session%
ST	StartTime. When datacollection for this sample was started. The value is number of milliseconds since 1/1-00 00:00. (DateTime in C#) %
D1 to D11	When key x down event occur, where x is the n'th key pressed. The value is number of milliseconds since ST%
U1 to U11	When key x up event occur, where x is the n'th key pressed. The value is number of milliseconds since ST%
ED	When the enter key is pressed. The value is number of milliseconds since ST%
ED	When the enter key is released. The value is number of milliseconds since ST%

Table 14: Samples collected from a PC.

A.4 user.data

The table below contains personal data about the participants. One row for each participant. This file is kept away from experiment data and is used solely for administrative purposes and to generate ID for the other files.

Fieldname	Description
ST	StartTime. When datacollection for this sample was started. The value is number of milliseconds since 1/1-00 00:00. (DateTime in C#) %
D1 to D11	When key x down event occur, where x is the n'th key pressed. The value is number of milliseconds since ST%
U1 to U11	When key x up event occur, where x is the n'th key pressed. The value is number of milliseconds since ST%
ED	When the enter key is pressed. The value is number of milliseconds since ST%
EU	When the enter key is released. The value is number of milliseconds since ST%
X1 to X11	Finger horizontal position on touch screen button. List of values separated by commas. The values are relative positions on key. 0 is far left and 100 is far right. The first value is when key is pressed and the last value is when key is released. Values between indicate finger movement while pressing a key. %
Y1 to Y11	Finger vertical position on touch screen button. List of values separated by commas. The values are relative positions on key. 0 is is far up while 100 is far down. The first value is when key is pressed and the last value is when key is released. Values between indicate finger movement while pressing a key. %
S1 to S11	Size of contact between finger and touch screen. List of values separated by commas. The values are a scaled value between 0 and 1000, but not observed close to max. One value for each X/Y value. %
P1 to P11	Pressure of finger against the touch screen. List of values separated by commas. The values are a scaled value between 0 and 1000, but not observed close to max. One value for each X/Y value. Assumed closely related to size values.%

Table 15: Samples collected from smart phones.

Fieldname	Description
ID	Assigned id%
DoB	Date of birth. Format YYYY-MM-DD%
Gender	Gender. 0 = male, 1 = female%
Hand	Right handed = 0, and left handed = 1%
Skill	Users self evaluation of typing skills. 0=beginnes, 1=Normal, 2=Expert%

Table 16: Experiment data about participants.

Fieldname	**Description**
ID	Assigned id%
Name	Name.%
Phone	Phone number%
Email	Email address%

Table 17: Personal data about participants.

B Evaluate Performance

```
%Using DS and DI to eavaluate detector last run
%We want to fint the threshold when FMR=0, and when FMR=FNMR=EER
%To find EER we need to search between the min(impostordistance) to
%max(subjectdistance), if they overlap.
%
%first we use binary search on DS to find the highest value where
%fnmr-fmr is positive, and the lowest value where fnmr-fmr is negative.
%second we check if there are thresholds in DI between the two found values
%that may change the result.

%Result(:,1) = threshold when fmr=0
%Result(:,2) = EER  (FMR)
%Result(:,3) = EER  (FNMR)
%Result(:,4) = threshold EER


Result = zeros( numel(DI), 5 );
DS = sort(DS);
for subject = 1:numel(DI)

    Result(subject,1) = min( DI{subject}(:) );  %FMR=0
    range = [1 numel(DS(:,subject))];

    if  DS(range(2)) <= Result(subject,1)  % not overlapping, use min as threshold
        Result(subject,2) = 0;  %EER (FMR)
        Result(subject,3) = 0;  %EER (FNMR)
        Result(subject,4) = (Result(subject,1)+DS(range(2)))/2;  %threshold -> EER
        Result(subject,5) = Result(subject,1)-DS(range(2));
    else
        %binary search
        while range(2) - range(1) > 1
            x = int32((range(2)+range(1))/2);
            diff = errorrate( DI{subject}, DS(:,subject), DS(x,subject));
            if diff == 0
                range(2) = x;
                range(1) = x;
            elseif diff > 0
                range(1) = x;
            else
                range(2) = x;
            end
        end
```

```
        t1 = DS(range(1),subject);
        t2 = DS(range(2),subject);

        %linear search on DI if more than 1 value between t1 and t2
        dirange = sort( DI{subject}( DI{subject}> t1 & DI{subject}< t2 ));
        if numel(dirange)> 1
            for i=1:numel(dirange)
                diff = errorrate( DI{subject}, DS(:,subject), dirange(i) );
                if diff == 0
                    t1 = dirange(i);
                    t2 = t1;
                elseif diff > 0
                    t1 = dirange(i);
                else
                    t2 = dirange(i);
                    break;
                end
            end
        end

        %choose the least diff of t1, t2 and (t1+t2)/2
        [diff Result(subject,2) Result(subject,3)] = errorrate( DI{subject}, DS(:,
        Result(subject,4) = t1;

        [x fmr fnmr] = errorrate( DI{subject}, DS(:,subject), (t1+t2)/2 );
        if abs(x) < abs(diff); Result(subject,2) = fmr; Result(subject,3) = fnmr;

        [x fmr fnmr] = errorrate( DI{subject}, DS(:,subject), t2 );
        if abs(x) < abs(diff); Result(subject,2) = fmr; Result(subject,3) = fnmr;E
    end
end

clear i;
clear x;
clear dirange;
clear t1;
clear t2;
clear fnmr;
clear fmr;
clear threshold;
clear subject;
clear diff;
clear range;
```

C ImitatorUno Software

```
int valCount = 12*2;
int delays[] =  {0 , 82 , 158 , 74 , 456 , 90 , 109 , 93 , 574 , 86 , 194 , 68 , 476 , 7
int values[] =  {1 , 1 , 7 , 7 , 0 , 0 , 5 , 5 , 1 , 1 , 4 , 4 , 0 , 0 , 2 , 2 , 2 , 2 ,
int actions[] =  {1,0 , 1,0 , 1,0 , 1,0 , 1,0 , 1,0 , 1,0 , 1,0 , 1,0 , 1,0 , 1,0 , 1,0}

int sessionCount = 20;
int sessionDelay = 5;
int tryDelay = 2;
int tryCount = 3;

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  for (int i = 0; i <= 13; i++){ pinMode(i, OUTPUT); }
  pinMode(11, INPUT);  //switch
  digitalWrite(13, HIGH);
}

int mode = 0;
void loop() {
  delay(50);
  if (mode == 0 && digitalRead(11) == LOW)
  {
    int count = 0;
    while ( digitalRead(11) == LOW)
    {
      delay(100);
      count++;
      if (count > 20) digitalWrite(12, HIGH);
    }
    if ( count < 20 ) mode = 1;
    else mode = 2;
  }
  else if ( digitalRead(11) == HIGH ) mode = 0;

  if (mode > 0)
  {
    for ( int s = 0; s < sessionCount; s++)
    {
      if (s > 0)
      {
        digitalWrite(10,HIGH);
```

```
        delay(70);
        digitalWrite(10, LOW);
        delay(800);
      }

      digitalWrite(13, LOW);   // set the green LED on
      for( int t=0; t<tryCount; t++)
      {
        for( int i = 0 ; i < valCount; i++ )
        {
          delay(delays[i]);
          if (actions[i] == 1 ) digitalWrite(values[i], HIGH);
          else digitalWrite(values[i], LOW);
        }
        if ( mode == 1) break;
        int count = tryDelay * 10;
        while ( count > 0 && t < tryCount -1 )
        {
          delay(100);
          count--;
          if (digitalRead(11) == LOW)
          {
            t = tryCount;
            s = sessionCount;
          }
        }
      }
      digitalWrite(13, HIGH);    // set the green LED off
      if ( mode == 1) break;
      int count = sessionDelay * 10;
      while ( count > 0 && s < sessionCount-1)
      {
        delay(100);
        count--;
        if (digitalRead(11) == LOW) s = sessionCount;
      }
    }
    mode = -1;
    digitalWrite(12, LOW);    // set the red LED off
  }
}
```