NTNU

Norwegian University of
Science and Technology

Presentation Attack and Detection in Keystroke Dynamics

Jonathan Ness

Master in Information Security
Submission date: June 2017
Supervisor: Patrick Bours, IIK

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

Presentation Attack and Detection in Keystroke Dynamics

Jonathan Ness

NTNU

Master's Thesis - IMT4904
NTNU Gjøvik, 2017

Dept. of Information Security and Communication Technology
NTNU Gjøvik
PO box 191
NO-2802 Gjøvik, Norway

Abstract

Biometric authentication is already making an entry in consumer security, notably with face and fingerprint recognition. Keystroke dynamics is an aspect of behavioral biometrics based on typing behavior to verify the identification of the user. It is both easy to implement, and is less intrusive for the user. If the system already relies on a password for authentication, then keystroke based biometrics can offer an additional hidden layer of security. Alternatively it can be used for continuous authentication to continuously verify the user identity based on their interaction with the system.

Many biometric systems have been demonstrated to be vulnerable to attacks. The research on attack detection is a way of making the systems more robust, and can be seen as an indicator of the maturity the systems have for security applications.

This project will focus on presentation attacks and detection techniques on keystroke dynamics using hardware to artificially inject keystrokes for the system. The goal of this project is to study the impact of keystroke dynamics as a possible security feature, we do this either by demonstrating the existing security, or by proposing techniques that can defend against targeted attacks from malicious actors.

Based on our results, we find that keystroke dynamics is probably not secure against a resourceful attacker. However, we also find that it does protect the user against attacks that most systems are normally vulnerable against. Further, we did not find any reliable way to deteect an ongoing attack. For future work we would therefore like to see more thorough study of the security, as well as research into possible ways of detecting ongoing attacks.

Sammendrag

Biometrisk autentisering gjør inrykk i forbrukersikkerhet, spesielt med ansikts- og fingeravtrykksgjenkjenning. Tastetrykkbasert biometri er en underkategori av adferdsbasert biometri som baserer seg på at tastaturadferden kan verifisere identiteten av en bruker. Det er lett å implementere, og tastetrykkbasert biometri forstyrrer brukeren veldig lite. Dersom systemet allerede benytter passordautentisering, kan tastetrykkrbasert biometri fungere som en skjult ekstralag med sikkerhet.

Mange biometriske system har blitt demonstrert sårbare for angrep. Forskning på angrepsdeteksjon er en måte å gjøre systemene mer robuste på, og kan bli sett på som en indikator av hvor modent systemet er til bruk i sikkerhetsapplikasjoner.

Denne oppgaven vil fokusere på presentasjonsangreps- og deteksjonsteknikker innenfor tastetrykkrbasert biometri, ved bruk av spesialisert maskinvare for å kunstig injisere tastetrykk i systemet. Målet med prosjektet er å modne tastetrykkrbasert som mulig sikkerhetsfunksjon. Enten ved å demonstrere eksisterende sikkerhet, eller ved å foreslå teknikker for å oppdage og forsvare mot målrettet angrep fra ondsinnede aktører.

Basert på våre observasjoner kom vi frem til at tastetrykkbasert biometri trolig ikke er sikkert mot en dedikert angriper. Allikevel ser vi at det beskytter brukeren mot angrep som de fleste systemer er sårbare mot. Videre kunne vi ikke finne noen pålitelig måte å detektere angrep på. For fremtidig forskning ønsker vi derfor å se mer grundig gjennomgang av sikkerheten, og videre undersøkelse i å oppdage pågående angrep.

Preface

The thesis before you is the final project to fulfill the graduation criteria from NTNU in the program Master Information Security, Digital Forensics track. The project was on behalf of the Norwegain Biometrics Lab and focus on attacking and detecting attacks in continuous keystroke dynamics based authentication systems. The project ran from January through May 2017.

To get the most out of the paper, general knowledge about information security is expected, basic knowledge of biometric systems is beneficial.

I would like to thank my supervisor, Dr. Patrick Bours, for his assistance and guidance throughout the project. Yngve Tandberg deserve a mention for his help when we faced hardware problems during the project. Finally I would like to thank my friends and colleagues that helped proofreading and going over my thesis the days prior to submissions.

As well as support and motivation from my loving girlfriend.

Jonathan Ness

Gjøvik, June 1st, 2017

Contents

List of Figures

List of Tables

1 Introduction

Every person possesses unique traits and characteristics that are inherent to their being. These traits and their use in uniquely identifying a person make up what is known as biometrics. These biometric traits are split into two categories: the inherent biological traits, and the learned behavioral traits. The biological traits are body features that are stable and generally hard to change. These features include DNA, ear, face, fingerprint, iris, and palmprint. Some behavioral traits are gait, keystroke, voice and signature, which are based on the normal behavior for a person, and can be changed with effort over time.

Advances in technology has resulted in more people having access to biometric security than ever before, with fingerprint and face recognition being common in many of the recent smartphones, tablets and laptops. As such there is a strong need to demonstrate the security of biometrics, and eliminate weaknesses from the systems to maintain security. Most of the systems rely on a single authentication during login rather than continuous authentication during system use.

Continuous Authentication refers to a system that continuously verifies that the user is the claimed identity, based on activity by the user. Keystroke dynamics on a computer is very suited for this purpose due to how unintrusive it is to the user of the system. This continuous authentication would be a constant ongoing challenge for an intruder[4]. Studies on mimicking someone's keystrokes behavior find that it is quite hard, even with full information about the victim's typing behaviour being provided[5].

We attempted to attack the system using specialised hardware that works as an intermediary through which the attacker interacts with the system. Tricking the computer into thinking thinking we are sending genuine input from the user is called a presentation attack. Then we tried to implement countermeasures to detect such attacks.

1.1 Topic covered by the project

The project attempted to attack a system for continuous authentication relying on keystroke dynamics, as a way to test how secure and robust the technology is to targeted attacks. This is a highly unexplored domain, with previous studies being conducted on mimicking the behaviour of the target manually[6, 7].

We attacked the system using custom hardware and a biometric template of the genuine user. A template is the stored information about the biometric characteristics the system relies on. The origin of the template could either be created by the attacker after observation, or a compromised template leaked from a database, akin to the several password leaks over the recent years. Some of the larger which include Yahoo[1], Adobe[2] and eBay[3]. It has also been demonstrated that biometric templates could possibly be reverse engineered[8].

[1] "Yahoo confirms data breach affecting at least 500 million accounts". Washington Post. September 22, 2016. Retrieved May 20, 2017.

[2] "Over 150 million breached records from Adobe hack have surfaced online". The Verge. November 7, 2013. Retrieved May 20, 2017

[3] "eBay asks 145 million users to change passwords after data breach". Washington Post. May 21, 2014. Retrieved May 20, 2017.

1.2 Keywords

Presentation Attack Detection, Keystroke Dynamics, Behavioral Biometrics, Continuous Authentication, Intrusion Detection Systems (IDS)

1.3 Problem description

There is currently limited research on presentation attack and presentation attack detection in keystroke dynamics based biometric systems. Many existing biometric systems are known to be vulnerable to presentation attacks, with various research proposing techniques for enhancing the robustness of the systems against attacks[9, 8, 10]. The research into attacking and defending a system relying on a given biometric characteristic can give an indication of the maturity of said characteristic for security applications.

If the characteristic can be attacked with ease, then there is inherently little security in systems relying on it. As of this time, there is little data to suggest that keystroke dynamics based systems are secure or insecure against presentation attacks.

1.4 Justification, motivation and benefits

The EU project TABULA RASA has previously stated the need to address spoofing attacks in biometric systems[11]. Biometrics is further the third way of identification, alongside tokens and passwords. It can offer advantages that tokens and passwords by themselves do not, and can be used to complement existing solutions to increase security[12]. For example, by adding fingerprint recognition to an access sytem that already relies on a access card and pin code as a two or three step authentication.

As it stands, there is minimal research on presentation attacks in keystroke dynamics, and even less so on attack detection. There have been studies of mimicking attacks where an attacker imitates the target's behaviour, yet that is narrow in scope as it does not account for attacks utilising external hardware[4, 6, 7]. We focus then on attacks that are utilizing specialised hardware to artificially send keystrokes to the system. If artificial keystroke input is shown to bypass the defence and avoid detection then we cannot proclaim a system as secure.

If we can demonstrate that keystroke dynamics based systems are secure even against hardware aided attacks, then we can hopefully see an increased adaptation of the biometric for security applications. Users can increase their security with minimal inconvenience to their daily use of the system, while attackers will face an additional security system to bypass in order to conduct malicious activities.

1.5 Research questions

Are continuous keystroke dynamics based systems secure against presentation attacks?

It is shown that biological biometric systems are vulnerable to presentation attacks, yet there is little research on presentation attack detection in behavioral biometrics. We will attempt to fill this gap by researching keystroke dynamics, a subsection of behavioral biometrics. We know that keystroke dynamics based systems are secure against least effort attacks, that is, an attacker simply behaving normally; as well as against mimicking attacks where an attacker tries imitate the target. As such we tested targeted attacks where the attacker has put in a lot of effort. We then checked if a given system is vulnerable to attacks using specialised hardware and a compromised template of the genuine user's typing behavior.

The attack is against the sensor level, not any of the underlying systems or protocols[8]. Figure 1 illustrate a biometric security system. Each of the boxes and their arrows are potential attack vectors. Attacks such as on the stored templates, matcher or to override the output fall within the domain of general cryptology, and protection against such attacks exist in that field already. Using fuzzy hashing, asymetric encryption and including timestamps.

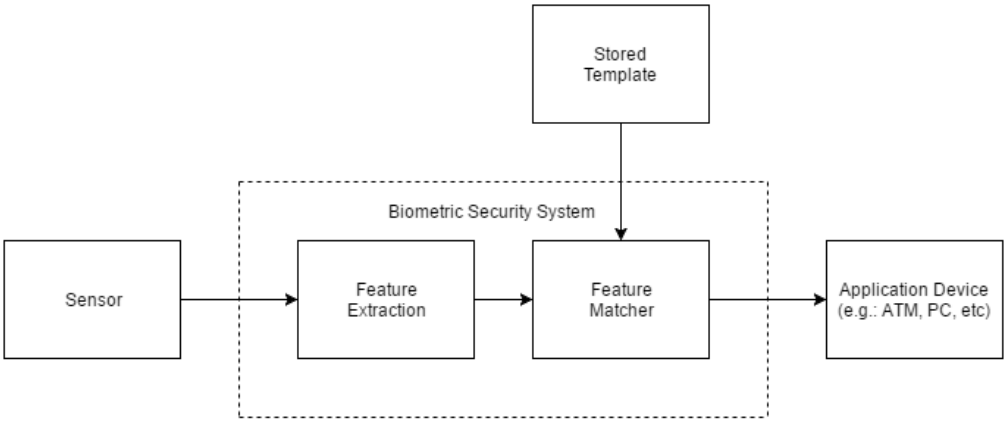

Figure 1: Illustration of a basic biometric system[1]

Following the initial research question, we have a secondary question: can we distinguish a targeted attack from legitimate use?

The primary question is to see if the system is robust. A system that can only sustain least effort attacks without compromising are less secure than system that sustain targeted attacks without failing. If we can demonstrate that a system is vulnerable to an attack, we can then proceed to develop countermeasures against the attacks. Therefore creating a more robust security system.

The secondary question is to see if the system can detect that it is being attacked. If the system detects an ongoing attack, it can override standard procedures and lock out the attacker to maintain the security, despite how all the recorded keystrokes fit within the threshold as specified by the template.

1.6 Contributions

The contributions of this project are complementing existing studies on the security and robustness of keystroke dynamics as a biometric feature. We will either demonstrate the existing security of systems, or that they are vulnerable to attacks utilizing custom hardware. If it is vulnerable, we tried to find reliable techniques of detecting presentation attacks as a way to make systems more secure.

Our goal is to mature keystroke dynamics such that it can be put to use in real applications and enhance the security for users, without infringing on the accessibility of the system.

2 Authentication

In our daily life there are numerous acts of identification and authentication happening. Anything from phone calls to debit card usage follows an identification or authentication. Identification is the act of establishing an identity, while authentication is verifying the claimed identity. This can be done through multiple means, commonly we split the authentication into three different ways, as Figure 2 illustrates[13].

- Knowledge Based - something you know
- Possession Based - something you have
- Biometric Based - something you are

Commonly only one means of authentication is used, but two- and multifactor authentication is becomming more common as we utilize new advances in technology, such as using smartphones as a possession based authentication or fingerprint scanners for biometric authentication[14, 15]. The next section gives a brief explanation of the different authentication methods with their strengths and weaknesses.

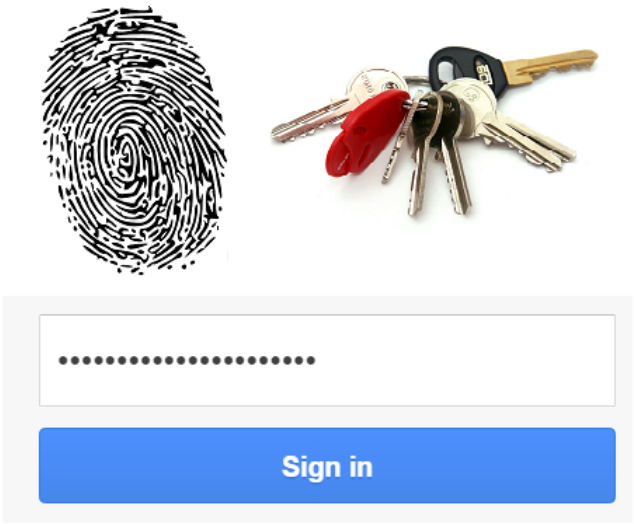

Figure 2: Examples of the three different means of authentication

2.1 Knowledge Based Authentication

One of the most common ways of authentication to systems are by means of a shared secret between you and the system, something only you know. A password, passphrase or pincode that will verify that you are the claimed identity. Today we use knowledge based

authentication for a wide range of services. Ranging from device access such as phone, tablet or PC, to financial transactions such as online banking, ATM and credit card usage.

Users generally prefer to use easy to remember knowledge for authentication, and often reuse the passwords on multiple services[16]. Commonly birthdates for pincodes, known names, favorite places, movies or books for passwords and passphrases[17]. Many of which can be easy to guess if one knows the person, and others which are easy to attack using brute force or dictionary attacks. Further, people commonly reuse their passwords across services, and keep them for longer periods of time rather than change them frequently. In keypads, this can lead to the keys used in the pin being worn down before the rest, as well as smudge attacks where leftover smudge indicate the keys used in the pincode[18, 19]. Physical notes of the passwords are common if the user is forced to make strong passwords to meet requirements such as including uppercased letters, special characters or numbers[20].

One of the strongest advantages to a knowledge based authentication is that it is cheap and easy to implement, with standard ways of solving the problem. The users are also used to remembering passwords, and will accept a password based authentication without much complaint. Managing passwords, such as changing passwords, forgotten passwords or new passwords for new users, is also relatively easy with existing standards.

The weakness is that the users make bad passwords, and reuse them across systems[20]. Enforcing requirements tend to lead to incremental passwords. An example is the increased password length, where users go from "123456" to "12345678" to comply with the new length requirement for the password. As well as capitalisation of words and appending special character requirements at the end, such as "Welcome42!" rather than randomly distributed such as "w2!elCo4me"[17]. This leave the systems vulnerable to various attacks, including across systems.

2.2 Possession Based Authentication

Possession based authentication relies on a physical item that only the legitimate user of claimed identity should possess. Commonly this takes the form of keys, visitor cards, bank tokens or smartphones[21, 22]. The security behind this form of authentication is that a physical item might be harder to forge on the spot, compared to looking over one's shoulder to notice the pincode or password[18].

The benefit of token based authentication is that it can be hard to falsify and replicate, and the users don't have to remember a password or passphrase. The physical device can also be accompanied by a knowledge based security mechanism to prevent strangers from utilizing the device. Using a mobile devices as a token is convenient option for the users, as they tend to be readily available most of the time[22].

The weakness of a token is that it can be expensive and tedious to replace, and the users might forget their token at home, leaving them unable to access the systems. A token by itself can also be abused if a stranger find the token lying around.

2.3 Biometrics Based Authentication

The inherent characteristics of a person such as behavior and appearance are the background for biometrics as an authentication factor. If a stranger approaches you, claiming to be a friend you know very well, you will almost instantly be able to dismiss them based on physical differences in appearance. Even they look alike, you can tell them apart by

their voice or behavior. For people, this is a relatively easy task as long as you are familiar with the person in question. Identical twins can often be mistaken for each other by strangers, while close friends and family are able to distinguish them more easily.

For an automated system, things become more complicated. A system need strict instructions on which characteristics and traits which should be used to compare against. There are also limitations to the technologies. Unlike token and knowledge based authentication, biometrics based authentication is not binary 100% or 0% correct. A token is either correct, or incorrect. In biometrics, it's mostly correct, or mostly incorrect, and this is described as the trust level the system has that it is the genuine user. Since the majority of cases are not black and white, there is a need for a threshold to determine what we accept as a valid user, and what we accept as an invalid user. For example, anything with a trust at or above 90% is accepted, and anything below 90% is rejected.

In biometrics we operate with a False Match Rate (FMR) and False Non-Match Rate (FNMR)[23]. These are false positives and false negatives of the matching algorithm respectively. A high false match rate means that an attacker is very likely to be granted access when presenting his own fingerprint, while a false non-match rate means that the genuine user is likely to be denied entry, despite presenting the valid fingerprint.

The system further operates with a Failure to Enroll (FTE), which means that the user is unable to enroll into the system for various reasons, this is mentioned below in the requirements for a biometric feature. A Failure to Capture (FTC), is an error where the sensor is unable to capture the trait, for example because of a dirty sensor.

Combining FMR and FTC, we get the False Acceptance Rate (FAR). Similarly, FNMR and FTC give False Rejection Rate (FRR). FMR and FNMR are algorithmic errors, while FAR and FRR are systemic errors, which include issues such as the sensor not working properly or the fingerprint being distorted when presented[23, 24].

Following are brief descriptions of the requirements for a trait to be considered for biometric usage[24]. Each of the traits relate to the security and accessability of the trait for security systems.

1. **Universality:** It should be a trait that every user possess, or a significant enough subset for the system to work properly. Users without the trait will not bee able to register for the system. For example: fingerprint based biometrics on a person without fingers.
2. **Uniqueness:** The trait should have a big enough variation between users. This is what allows the system to distinguish between the users.
3. **Performance:** It should be reliably able to distinguish users at an acceptable pace in relation to the application.
4. **Permanence:** The trait should be stable and not have significant variation across short timespans. If the trait is unstable, then it can be changed in a short period of time, and therefore preventing access to the user.
5. **Collectability:** Collection of the data from the trait should be feasible and convenient.
6. **Acceptability:** The users should accept that their biometric information is captured.
7. **Circumvention:** It should be hard to circumvent the system by mimicking or forgery. If it's easy to circumvent, it is not secure.

However, the requirements vary for different applications. Fingerprint scanning has recently become widely accepted for authentication to portable smart devices such as phones, tablets and laptops. The process of authenticating is done in a short enough time to be considered practical for the purpose, but it has a relatively weak defence against circumvention[25, 26]. It is worth noting that the users are also willing to give their fingerprint to the system. DNA based biometrics are currently much slower, but have a significantly higher protection against circumvention. The slow processing time, in combination with the reluctance to hand over your DNA, make it unsuitable for use in the same kind of system, despite being more secure[24, 20].

Assuming a given trait fulfill all the aforementioned criteria, it is suitable for biometric based authentication. The biometrics trait then get split into one of two different categories that are briefly explained below.

2.3.1 Biological Biometrics

Stable physical traits that change slowly over time, without any real option for the user to change it immediately. Commonly used are fingerprint, iris, ear, face, DNA, etc.. Many of these traits have been demonstrated vulnerable to various attacks[25, 9].

2.3.2 Behavioral Biometrics

A learned behavioral trait from the user in their normal activity. It can be changed, but it takes time and requires effort from the user. This includes traits such as gait, signature, voice and keystroke. These traits are visible to everyone, but can be hard to mimic and forge, even if the specific movement is known[18].

Chapter 3 will go through the state of the art for Behavioral Biometrics and Keystroke dynamics, the focus of this project.

3 State of the Art

This chapter covers the state of the art in relation to this project. It will provide a brief overview of behavioral biometrics, before narrowing down towards presentation attack detection in continuous keystroke dynamics based authentication systems. It is important to have the baseline of knowledge about general biometrics in order to understand the technicalities as we narrow down towards the specific issue we work with.

3.1 Behavioral Biometrics

In the Second World War, the military would send messages of morse code. A technique called "Fist of the Sender" arose, in which the keying of dots and dashes were used to identify the sender as either ally or enemy. The operators would recognize the patterns of other operatros they had communicated with previously, and by noticing drastic changes in the way the messages were sent, they could tell if the sender was the same person. This is the predecessor to keystroke dynamics, and among the earliest practical applications of it.

Aside from keystroke dynamics, we have several other characteristics that fall within the category of behavioral biometrics. Such as gait, signature and voice. Common for these are that they are learned, and can be changed through effort, while biological biometrics cannot be permanently altered by the user. It would also seem unlikely that the user will forget how to behave naturally, which mitigates some of the issues with knowledge and token based authentication. Because they represent the natural behavior of the user, behavioral biometrics have shown to be suitable for continuous authentication, which is covered in Section 3.1.1.

While there is significantly less research on behavioral biometrics compared to biological biometrics, there is a lot of research on using behavioral biometrics, for example gait[27, 28, 29, 30], voice[31, 32, 33] and keystroke dynamics[34, 35, 36]. Additionally there is a fair amount of research on the security against mimicking and presentation attacks for the different traits. In particular gait[37, 38, 39, 40], voice[41, 42, 43], handwriting[44] and keystroke dynamics. We cover attacks on keystroke dynamics further in Sections 3.2.1 and 3.2.2.

We have made the distinction that mimicking attacks are the malicious user themselves interacting with the system, trying to imitate the genuine user, and presentation attacks relies on some intermediary software and/or hardware layer between the malicious actor and the system.

Some of the studies on attacking other behavioral biometric features do illustrate the difficulty of of attacking the continuous keystroke dynamics based systems. Because biological biometrics are static, they can be easier to capture and extract the valid signature. Behavioral biometrics add another dimensionality of time for the feature, and you often need significantly more data to extrapolate the normal behavior for the user. As a result, we found that there are few studies on presentation attacks of keystroke dynamics specifically, where the user's genuine timing information is artificially presented to the system, either by software or hardware.

3.1.1 Continuous Authentication

Continuous authentication is based on a ongoing verification that the user remains the same. This is different from periodic authentication, which is a check of the user at fixed or random intervals, or for specific actions. Static authentication is the common way of authenticating at the very start of a session, but is vulnerable should the user leave without logging out of the system, as another malicious actor could act on behalf of the original authentication.

Periodic authentication will have a trade-off between how often to authenticate, and how long a malicious actor would get to use the system before being required to authenticate themselves. The usability of the system would decrease based on how short a timespan before you have to authenticate, while the potential damages would increase with a longer timespan. This is commonly mitigated by requiring a period of inactivity before being locked out.

Using continuous authentication, every action on the system is evaluated against the user template, which results in a trust level the system have that this action come from the genuine user. It happens seamlessly and immediately in the background of the system, compared to with periodic authentication where the intruder have free access until the next authentication. Therefore the system would remain secure, even if the user forget to log out of the system after their session.

Due to the continuous nature of such a system, FMR and FNMR are of limited interest for performance measure. We rather focus on how long it takes the system to recognise the malicious actor. If an attacker can use the system for hours before being locked out, the damage might be done, but since they were locked out eventually, they did not count towards FMR. For use in continuous authentication systems the need for an algorithm that accounts for ongoing use is needed. Soumik Mondal and Patrick Bours created a dynamic trust level algorithm that would overcome some of the limitations of a static trust level for each input[45].

It is based upon dynamically changing the trust level as the user interacts with the system, and will lock down when the trust falls below a threshold. The performance measure for continuous authentication is thus the average actions a genuine and imposter can take before the system locks them out. We want the average number of genuine actions (ANGA) to be high, and the average number of imposter actions (ANIA) to be low.

3.2 Keystroke Dynamics

Keystroke Dynamics is a subset of behavioral biometrics, it is based on the fact that a learned way of typing is unique to each person[46, 47]. The slight variation in timing information for each key press and release provide us with enough information to reliably distinguish between users[36]. While it normally refers to keyboard usage, keystroke dynamics have been demonstrated to work on touch screens, like smartphones, and keypads[34].

For keystroke dynamics, timing data is the feature we extract and analyse[48]. This includes duration, time between key-press and key-release, as well as the latency, time between key-release and the following key-press. This is illustrated in Figure 3, which also include an example of negative latency. Negative latency occurs when you press the next key before releasing the previous key. Commonly this occur when using modifiers such as shift, but can also occur when typing at a fast pace.

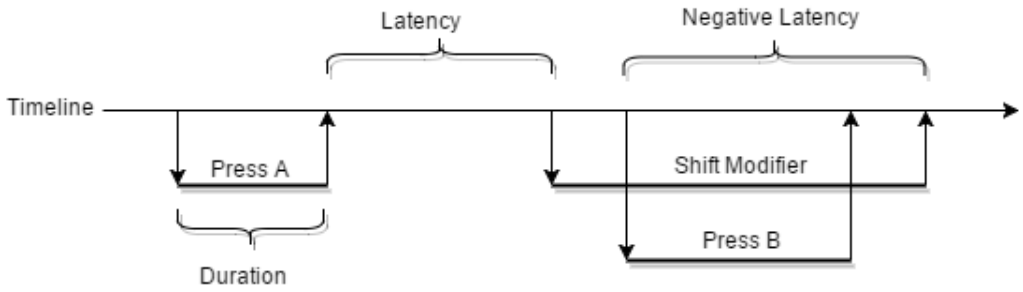

Figure 3: Illustration of timing details from keystrokes.

Loss of timing information can occur when the user deletes previous keystrokes. How to handle missing timing information is tricky, as the user can still be legitimate, despite making corrections. On the other hand, a malicious actor can intentionally delete keystrokes in order to leave little information for authentication. It was shown that simply rejecting a password entry because of corrections and missing timing data decreased performance significantly[49].

Multiple algorithms are viable options for determining if a given set of keystrokes belong to a given user[50]. Among the more common methods is using a distance metric between the input and the stored template values. Some of the common ones includes Euclidean, Manhattan, and Mahalanobis with normalized variants[51, 52, 53]. Other methods include Neural Networks, Clustering algorithms, Fuzzy logic and Support-Vector Machines[53, 54, 55, 56].

While the normal approach relies on timing data, there are possibilities of using other traits as well. Behavioral traits such as spelling mistakes or corrections are not visible from the timing information, but could be used to identify the user. However, these traits might not occur often enough to give sufficient performance on their own. As such it would be more of a secondary trait to filter on. A language filter could be possible in detecting an attack. In the event that a user suddenly start writing fluently in a foreign language, it might indicate that the system is compromised. Currently there does not appear to be any linguistic check in continuous authentication systems relying on keystroke dynamics, input with random characters and valid timing information has been shown as being accepted[57].

Following we have two subsections on mimicking attacks, where a malicious user attempts to imitate a valid user, and presentation attack where the keystrokes are synthetic as they are sent to the system.

3.2.1 Mimicking Attacks

In keystroke dynamics, a mimicking attack would be imitating a target's timing in their keystrokes. Either for a static password sign on, or for continuous authentication in all keystrokes while using the system. Characteristics of the target's typing behaviour could be recorded by sound or video, or by a keylogger on a different compromised system, in order to allow the attacker to train and improve mimicking without having compromised the target system[58].

One study found that when knowing the target's typing pattern, and attacker could learn and successfully imitate the target with almost perfect acceptance rate[6]. A differ-

ent study found that even with feedback, some of the attackers were unable to improve the performance of their mimicking attack. This suggested that mimicking of keystroke dynamics were not as easy as previous studies concluded[7]. This could possibly be attributed to the typing behavior of the victim.

The typing behavior or the genuine user naturally influences the performance of mimicking attacks. An irregular or inexperienced typist would have larger standard deviations for their typing behavior than an experienced typist, and the margin of error would therefore be greater. Comparably an experienced writer would have a much more stable behavior, and the standard deviation and margin of error would be smaller, making it harder to mimick.

To bridge the two opposing conclusions, a study by Vathsala Komanpally found that the success rate of mimicking depended on the password and typing characteristics of both the target and attacker[5]. The password "welcome42" was significantly easier to attack than a more complex password such as "Aidu50Xv", which again supports the use of stronger passwords in general. It was suggested that on more complex passwords, it was harder for the attacker to change their own typing behavior compared to on easier passwords. Attackers with similar typing characteristics af the genuine user also achieved a better performance on their mimicking attacks, however there were few cases where it was sufficient to reliably gain entry into the system.

This indicates that keystroke dynamics for static authentication hold up against efforted mimicking attacks, especially when one considers the feedback provided during the experiment and the number of attempts the attacker had to practice. Given the poor performance of mimicking attacks on passwords, it would seem unlikely that an attacker could reliably mimick the target for a longer period of time, as would be required for a continuous keystroke dynamics based authentication system[57].

This follows in line with the results presented by Kevin Killourhy and Roy Maxion, where they analyse the different factors that affect the performance of password based keystroke verification[59]. They conclude that practice from mimicking attacks have a minor impact on the Equal Error Rate (EER). Compared to the algorithm used in the system, the training on data, and updating the user template over time, all of which had a greater impact on the EER. The EER is the threshold balance where the FAR and FRR are equal.

3.2.2 Attacks on Keystroke Dynamics

There is little research on presentation attacks within keystroke dynamics. A presentation attack is the act of tricking the sensor with a false input, rather than attacking some other aspect of the system. Refer to Figure 1 again, which illustrates a basic biometric system, where each box and line are possible vectors of attack. To attack the sensor of a system require little technical knowledge, and can be as simple as holding up a printed picture to a facial recognition camera[9].

For continuous authentication, things get more complicated. The attacker will have to continuously present the correct behavior of the victim for a prolonged period of time. This will require a more robust way of attacking the system, to avoid being locked out after a short timespan. Particularily for keystroke dynamics, which is one of the normal ways of interacting with the system, it will require some software and hardware between the user and the system.

Following we have the most relevant research of presentation attacks on keystroke dynamics. The TUBA project and the Snoop-Forge-Replay attack.

TUBA:

A project by Deian Stefan and Danfeng Yao presented a synthetic attack on keystroke dynamics using bots that generate keystrokes based on statistical analysis of the user's typing behavior[2]. The project got the codename TUBA, which stands for Telling Human and Bot Apart. The architecture is set up in a client-server model, which would also allow it to detect malware and automated attacks against the system[60]. Figure 4 illustrate this setup.

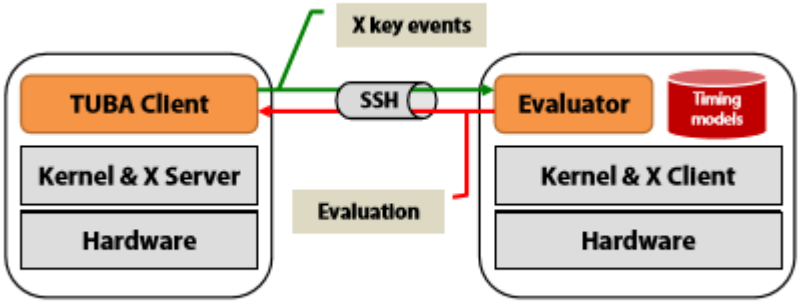

Figure 4: The client-server architecture for TUBA[2].

Due to the inherent network capabilities of X Window System (X11) that was used in the project, TUBA is suitable for thin clients, VPN and remote desktop solutions. The XTrap extension for X11 is what allow the core input to be captured for timing information. Because of how X11 works, TUBA can also be run locally on a machine. The applications are then set up to run as X Clients, and the operating system on the machine itself act as the X Server. This assumes that the computer is safe from malicious software which could otherwise bypass TUBA.

As with other biometric systems, the user has to go through an enrollment phase in which the security system is trained to learn the patterns of the user. This is described as happening by a secure connection to the server using SSH and X Window System with XTrap enabled, which allows for capturing of keystrokes by the user and sending directly to the server. To complete the enrollment, the user has to type a given number of strings a given number of times, after which the TUBA server trains on the data using a support vector machine[2].

After the system is trained, TUBA can run in a few different modes. It can both present a challenge as it observes suspicious behavior or at regular intervals, and as a non-intrusive mode where the keystrokes are analyzed continuously. This would be periodic authentication and continuous authentication respectively.

Because the projects assume that the X Server and user's typing information is secure, they do not send attacks to the system based on the genuine user's typing behavior. This potentially makes it vulnerable against an attacker which has access to the genuine user's timing information. The two generative bots they've created does thus not represent a

strong targeted attack.

The Gaussian Bot and the Noise Bot generate keystroke timing information based on Gausiaan distribution, and noisy information based on the mean $\pm$ noise. With fixed strings, presumable for the challenge-response setup, they report 95% true positive and 98% true positive for Gaussian Bot and Noise Bot respectively. Indicating that the detection rate for the bots are high. For comparison, they report a 92% true positive for distinguishing between the users.

The main contribution from the project would therefore be against malware and automated attacks, as well as providing a personalised alternative to CAPTCHA for periodic authentication. It does not consider targeted attacks from resourceful attackers.

Snoop-Forge-Replay:

Snoop-Forge-Replay is an attack developed by Rahman et al. for keystroke dynamics based continuous authentication systems, expanding upon their previous research[61]. They present it as an attack that exploits the overreliance on least effort attacks as a de facto performance evaluation[57]. They claim that as little as 20 to 100 characters is enough for them to achieve high error rate in state of the art systems. The attack follows 3 steps as outlined below.

1. **Snoop:** steal the victim's keystroke timing information using a keylogger.
2. **Forge:** create a typing sample using the stolen timing information.
3. **Replay:** to replay the forged text such that the system accepts it as genuine.

The attack uses a keylogger and keystroke emulator to capture and generate the keys that the security system registers at the sensor level. This suggests that the system have to be compromised from the beginning. They also relied on synthetic data to represent the victim's typing behavior, which was done using the keystroke emulator. This was defended by arguing that the cumulative typing time from genuine users would have been too long. Their estimates were approximately 10 years worth of typing, generated through virtualisation and software. The act of virtualisation could also be used by an attacker to run multiple attacks in parallel against the systems. By having multiple paralell channels of attack that get authenticated separately.

The data was collected from the 150 users at 3 different sessions with 6 months in between each, in total spanning a year. The templates they used were a 26x26 matrix containing all bigraphs for the English letters of latencies. For duration, each bigraph would only hold the duration of the first letter. So the **ab** bigraph would only hold the duration for **a**, which could potentially be different from the duration for **a** stored in the **aa** bigraph. Statistical anomalies such as outliers were filtered out prior to the keystroke emulator.

The participants were asked to type two types of text. Free text where the users were allowed to compose and type their own text, allowing for spelling mistakes, typographic errors and corrections. As well as one text where the participants were copying a provided text provided by the project. They free text were about 300 characters, and the copy text was 1800 or 1200 characters.

The attacks presented from the Snoop-Forge-Replay attack had two parts. A Dummy text and a series of timing information for the characters in the dummy text. In the event of multiple timing information for a particular bigraph, the average is used. In the event

that there was no timing information, the characters could be dropped. Alternatively, if the attacker require the complete text, they suggest forging the timing information as large values, intending for the outlier detection to remove them before authenticating the user. This was defended by how the goal is to demonstrate how forgeries based on snooped text could evade detection. They also argue that the continuous authentication systems do not run a linguistic check on the text presented, allowing for nonsensical text or different languages to be generated[57]. For example the text "Hello World!", without timing information for **l** and **d**, turn into "Heo Wor!". Which could be detected using a linguistic check of the text.

During the Snoop-Forge-Replay attack they use a gaussian perturbation of the timing information. This because they assume that a system could pick up on fixed timing information. for instance, should they find the duration of **a** 150ms on average for a victim, it would be an anomaly if all the durations were 150ms in the replayed text to the system. This would allow the system to detect an ongoing attack by looking for these anomalies.

They note that the best algorithm for zero effort attacks appeared to perform worst during the attack. Supporting their claim that the performance measure of simply distinguishing users have inherent flaws. The best approach for distinguishing users might be the worst approach during an attack. They also mention that they get a high EER when using fewer characters, 20-100, but that these face the limitation of largely repeating characters, and thus produce no meaningful text. They also observe that longer text became less effective against certain detection methods. This is because the system relied on bigraphs, and how a few bigraphs make up a significant portion of the English language.

Their performance score show that zero effort attacks had an EER between 0.03 and 0.285, while the EER during attacks were between 0.487 and 0.912. Meaning that the EER increased between 69.33% and a staggering 2730.55% for the various matching approaches they used during an attack.

Since the attack currently does not appear viable for malicious purposes, the main contributions from this project was to illustrate that systems are vulnerable to forged attacks, and that the performance measure for behavioral biometrics is flawed. The performance measure does not account for how easy it is to attack the system outside of zero effort attacks. They further propose that including text and language for authentication would improve the resilience against attacks.

4 Hardware

This chapter will cover the hardware we intended to use for the project, including the Typing Mimicking Robot that we later replaced due to technical difficulties. There are multiple viable approaches for the attack other than a robot, such as software or small microcontrollers that interact directly with the system. One such controller is the USB Rubber Ducky, which is the tool we decided to use as a replacement for the Typing Mimicking Robot.

4.1 Typing Mimicking Robot

In this section we will briefly cover the hardware that was chosen during the creation of the robot, as specified by Yngve Tandberg and Rune Bjørneseth in their project of building the robot[3]. The robot is depicted in Figure 5, and was a bachelor project at Gjøvik University College (GUC) in 2015.

The design of the robot is mainly for demonstration purposes. It would let you visually demonstrate artificially pressing keys up and down, as opposed to purely with a microcontroller as in the USB Rubber Ducky described in Section 4.2. Both methods have strengths and weaknesses.

Figure 5: The robot to be used in the project.

4.1.1 Microcontroller

Various solutions were suggested for controlling the robot, among them Field Programmable Gata Array (FPGA) and Programmable Logic Controller (PLC). The project decided on using a microcontroller in order to make it cost effective. There were also suggestions to using multiple microcontrollers for different aspects of the robot, but they decided against it due to the various conflicts it could present.

They decided on using an ACR microcontroller from Atmel. Due to uncertainty around the required filesize of the keystroke template, they assumed a minimum of 45kB. This calculation was based on each of the 105 keys on the keyboard having a specified duration and standard deviation of one byte, as well as each of the keys having a latency to the next key. Which would give a matrix of 105*105 in size for all possible latency combinations of keys, and each with latency and standard deviation.

$$105{\times}2B + 105{\times}105{\times}2B = 22.260B$$

Since they intended to allow for two templates to be stored on the robot, they assumed a minimum of 45kB. The selection of microprocessor ended at AVR AT90USB646, since it had 64kB storage, and 48 bidirectional dataports that could be used for activating the keyboard.

It turns out the duration and latency values would require more than one byte of storage, as the values have a greater variation than 256 values. The standard integer value is four bytes, and would quadruple the size of the template such that it wouldn't fit within the available memory on the microcontroller. Alternatives such as an offset value, or to have each of the values represent a stap of 5-10ms would be possible, and would allow the system to control keystrokes with enough granularity. If there was no need for two templates in the system, it would be possible to use a short integer for the values. A short integer consist of two byte, and would provide us with a range of 65536 different values, over a minute in milliseconds.

4.1.2 Keyboard Coils

For activating the keys on the victim keyboard, the project decided to use solenoids that pull a pin down such that it presses the keys, illustrated in Figure 6. Various configurations of size of the wire and the rods were tested before eventually ending up with a copper wire at 0.28mm radius, spun 1190 times into a coil. This provided sufficient force to press down the keys at the keyboard, while maintaining an acceptable power draw and temperature.

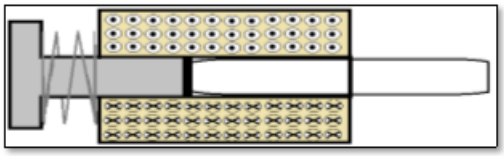

Figure 6: Illustration of the keyboard activation mechanism[3].

A solenoid for each key were then placed on a board plate over the keyboard, and activated using a shift registry to allow the 48 outputs to control the 105 keys on the keyboard. Due to how each solenoid had to fit over a key, the size of the solenoids had a limitation of about 17mm*17mm, the size of the majority of keys.

By controlling multiple solenoids, we would be able to manipulate the duration of keystrokes, as well as the latency between them. We could also spoof negative durations in this manner, by activating the solenoid for the following key, before the former key is released by the solenoid.

4.1.3 Other Aspects

A power supply, as well as the microcontroller and shift registers were enclosed in the red box as seen in Figure 5. The positioning of the Power Supply Unit is such that the fan will try to cool the coils from the keyboard. On the opposite side of the keyboard is another fan mounted, though not yet connected at the time. It will cool down the solenoids during longer typing sessions to avoid overheating.

Drivercards were created to power the solenoids, as the shift registry would be unable to provide enough power by itself. The finalised design is depicted in Figure 7. This, along with the solenoids, sits on the circuit board overlaying the keyboard. Allowing for the microcontroller to activate each key individually.

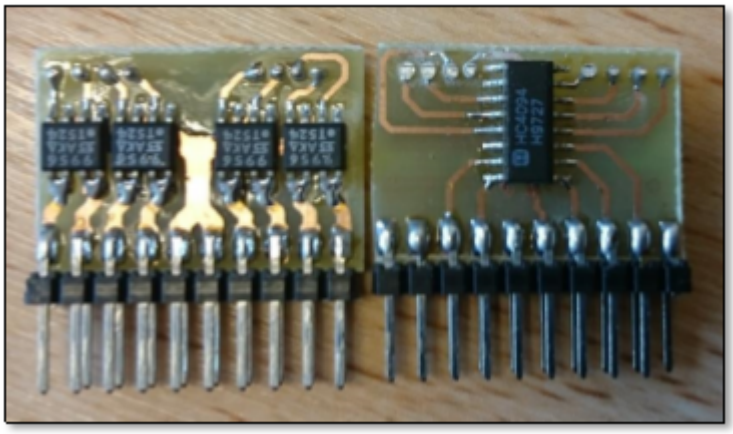

Figure 7: Picture of the drivercards to power the solenoids[3].

4.1.4 Issues

We experienced issues with the Typing Mimicking Robot, and it was later replaced with the USB Rubber Ducky in order to get results within the timeframe of the project. These were hardware related, and required consulting with Yngve Tandberg for locating. As such we did not feel confident in fixing them within a reasonable timeframe and had to abandon the robot.

Initially we were unable to load code onto the microcontroller. This was because the wire powering the bootloader had been cut to avoid tampering prior to the presentation of the project back in 2015. After locating the problem and discussing why it was done, we ended up soldering a physical switch as a safeguard. As shown in Figure 8 and 9.

Following that we found there was an issue with the power rail used to deliver power to the solenoids. We started debugging and consulting with Yngve Tandberg as to where the issue was, and how to fix it, but due to time constraints we ended up switching to the USB Rubber Ducky. This was also partially because we were uncertain in regards to future issues with the robot, which would delay the project further.

It should be noted that the software side was still working as expected by the time we switched to the USB Rubber Ducky, and we would have been able to modify both the duration and latency of the keystrokes with the robot.

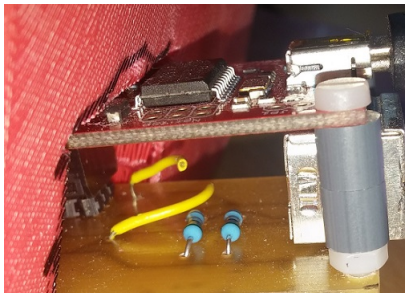

Figure 8: Picture of the wire that would power the bootloader.

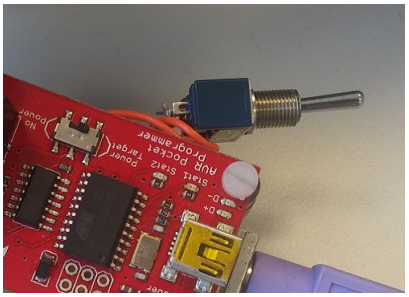

Figure 9: Picture of the physical switch that replaced the wire.

4.2 USB Rubber Ducky

USB Rubber Ducky is a USB-stick designed to look like a generic storage device as shown in Figure 10. Inside the device however, lies a microcontroller with its own SD card, depicted in Figure 11. Upon connecting to a USB port, the device will boot a small Operating System that presents itself as a keyboard to the system. After which it will inject keystrokes as defined by the payload.

Because is a penetration testing tool, timing of keystrokes have not been the focus of the tool. There is a built in delay function meant to be run between inputs, as to give the system time to process them. This can be used to mimic latency between keystrokes by having a delay between each key press. There was no built in functionality that would let us adjust the duration of keystrokes for the USB Rubber Ducky.

The payload is a file loaded on to the USB Rubber Ducky, which is compiled from the Ducky Script, which will be mentioned in Section 4.2.1. The payload is normally intended to gain remote access to the system, or elevate privileges if we already have access to the system. Less malicious purposes can include automation of setup and identical installation across multiple devices.

Figure 10: Picture of the USB Rubber Ducky when completely assembled

Figure 11: Picture of the inside of the USB Rubber Ducky

4.2.1 Ducky Script

The USB Rubber Ducky comes with a scripting language used to generate binary payloads for the device. Out of the box these send a string of text to be output into the system with minimal delay. There is a delay function, but it is mainly intended between commands as to allow the system to process the instructions.

The script needs to be compiled down to a binary executable for the USB Rubber Ducky to run. The compilation is done by a tool called Ducky Encoder, which is both available as a jar executable, java source code, or as an online service[1].

There is no functionality to control the duration of keystrokes out of the box. It could be possible to adjust the duration of keystrokes by modifying the source code for the Ducky Encoder, because we do observe both a key-press and a key-release when using the USB Rubber Ducky. It should simply be to input a delay between these two actions. However, we would have to locate and modify the code in a suitable manner to achieve this. This was not done during this project due to time constraints, and is explained further in Chapter 8.

Table 1 shows the basic commands of the script with a description of their parameters and uses. There are more commands specific for special keys and modifiers. Example being Ctrl, Alt, Backspace and Enter.

Command	Parameters	Function
REM	Any free text	Works as a comment.
DEFAULTDELAY	Whole number	Sets a default delay in ms between commands
DELAY	Whole number	Overrides default delay before the next command
STRING	Any free text	Writes the given string to the system
REPLAY	Whole number	Repeats the previous command the specified times

Table 1: Basic functionality of the DuckyScript

Table 2 shows a simple Ducky Script for opening Notepad in Windows, and writing "Hello World!" into the application. Normal script on the left, and the latency adjusted script on the right. As is evident in the table, the latency adjusted script is significantly longer and harder to follow, while the normal Ducky Script is more straightforward. It also illustrate our need for a script to automatically generate the Ducky Script, rather than trying to write it manually in a manner where we would get significant text to input to the continuous authentication system.

4.2.2 Assisting Scripts

We created a script in Matlab that will convert a given input text to Ducky Script, with matching latencies between each keystroke. The script takes a latency template and free text as input, and output directly to a text file with the formatting for the Ducky Script. This again have to be compiled down to binary such that the USB Rubber Ducky can execute the payload.

Notable here is that we cannot set a negative latency, because there is no adjustment to duration of a key press out of the box. We are therefore unable to make it such that the second key is pressed prior to the release of the first key, only make a delay between the key-release and the following key-press. The USB Rubber Ducky does however auto-

[1] "USB Rubber Ducky Wiki", http://usbrubberducky.com, retrieved May 31, 2017

Script	Explanation
DELAY 1000	Initial delay
GUI r	Open "run"
DELAY 500	Wait for system
STRING notepad	Write notepad
DELAY 500	Wait for system
ENTER	Start notepad
DELAY 500	Wait for system
STRING Hello World!	Enter text
ENTER	Newline.

Script	Explanation
DELAY 1000	Initial delay
GUI r	Open "run"
DELAY 500	Wait for system
STRING n	Writing of notepad
DELAY 124	Latency
STRING o	Writing of notepad
DELAY 245	Latency
STRING t	Writing of notepad
DELAY 178	Latency
STRING e	Writing of notepad
DELAY 219	Latency
STRING p	Writing of notepad
DELAY 98	Latency
STRING a	Writing of notepad
DELAY 141	Latency
STRING d	Writing of notepad
...	...

Table 2: Side by side comparison of a simple ducky script.

matically uses modifiers such as **shift**, **control** and **alt** to modify the keystrokes during execution. When the Ducky Script contain special characters or uppercased letters, the payload will inject the appropriate modifier prior to the keystroke in order to get the desired input. This creates some negative latency as we normally see with modifiers, but we were unable to adjust the negative latency of these instances manually without modifying the source code.

5 Data Collection and Template Generation

This chapter will cover the tools and processes used during data collection and template creation, as well as the technique we used to estimate and fill in missing data in the templates.

5.1 Behavior Logging Tool

Behavior Logging Tool, or BeLT for short, is a tool that captures user interaction with the system[62]. The data is stored in a CSV file with a format for easy reading. We will only focus on the keystrokes, and can therefore ignore the other information provided, such as mouse movement, mouse click, program selection, etc..

Note that we experienced some anomalies when running BeLT. For instance, the character "∼" appears immediately if BeLT is capturing the keystrokes, rather than remaining in memory for accenting letters such as **ñ**. We do not believe the few anomalies we experienced would have any significant impact on the captured data.

Table 3 shows example of the relevant data stored by BeLT. Note that this only applies to keystroke events. Other types of events have other fields that describe different information relevant to that type of event.

1. The first column represents the event number. Mouse movement in particular generate a lot of events.
2. The second column represents the event type. M for mouse, S for software, H for hardware and K for keystroke.
3. The third column represents the type of action. For keystrokes, it's D for down, and U for up, the key press and release.
4. The fourth column contains which key that is being pressed or released.
5. The fifth column represents the timestamps. Duration and latency will be calculated using these.
6. The sixth column is the preceding event. Every key-release follow a key-press. A key-press can refer to the application currently active, for example to distinguish between a search in the browser and writing in a text editor. This is explained more in Chapter 7.
7. The seventh column represents the modifiers, such as Shift and control.

5.2 Data Collection

Data was captured from several users over several days, without any strict requirement as to their activity during capture. For best effect the user should use the system as normal, and preferably do some writing to generate keystroke data. We assume QUERTY keysboard layout, but the typing behavior should represent the user's normal typing behavior. If the user normally type using the Dvorak layout, it would be their normal behaviour, and this again affect their timing information. The same applies to the keyboard language

1	2	3	4	5	6	7
1043	K	D	a	505457812	0	0
1044	K	U	a	505458328	1043	0
1045	K	D	b	505459656	0	0
1046	K	U	b	505459765	1045	0
1047	K	D	c	505460796	0	0
1048	K	U	c	505461453	1047	0
1049	K	D	d	505462703	0	0
1050	K	U	d	505462812	1049	0

Table 3: Example of the data stored by BeLT

settings.

Short bursts of typing does not necessarily represent the normal typing behavior of the user, and could often be login credentials. Usernames and passwords could typically be patterns that are ingrained in muscle memory from repeated use, rather than being something that the user focus on typing. This would naturally distort the data away from the natural behavior of the user. Continuous authentication performs better with longer paragraphs of writing.

BeLT was run in the background on the user systems, capturing the activity during a normal day of work. The data captured include mouse movement and system information, which was later filtered in the purpose of this project. We observed varying amounts of writing across different users and sessions of data capture.

Because the typing data is biometric by nature, it is inherently sensitive. It could further contain sensitive data, for example login credentials or credit card information, and therefore we did not analyze the data directly. Instead we would only run overall algorithms to process the data.

5.3 Data Processing and Filtering

We run a script to extract the useful information from the data collected. Since we only care about keystrokes, we can do a rough initial filter on the event type. We then iterate over all keystrokes, and calculate duration and latencies for use in a template. The majority of keystrokes are centered around text, rather than function keys, special characters or numbers. We also observe that the use of function keys impact the timing information of keystrokes, such that "shift+a" for instance, will give a different duration for "a" than if typed normally. Similarily the latency between "A" and "b" appear influenced by the capitalisation.

It could also be possible to distinguish between left and right shift and control, which could reveal other typing traits.

5.3.1 Timing Extraction

To extract the duration of a keystroke, we need to find the key-press and key-release pair. Since the event for key-press is referenced in the key-release event. Simply looking for each key-release and subtracting the preceding key-press would give us the duration of the keystroke.

Latency is going from key-release to key-press of the following key, and require us to store bigraphs of which keys are pressed in sequence. This is done because pressing "A-B"

has a different pattern than pressing "A-C". As mentioned, latency can be negative when the release of the first key happen after the press of the second key. This complicates the extraction. When we find a key-release, we have to find the corresponding key-press, and then find the following key-press. If we only look for the key-press that happen after key-release, we miss out on instances with negative latencies.

5.3.2 Processing

After the initial filtering and data extraction, a script iterates through all keystrokes and stores durations and latencies in their respective arrays. Following that, we go over the stored data and calculate the mean and standard deviation on keys with sufficient data points. We remove outlier data that are more than a given multiples of standard deviation away from the mean. Outliers are values that significantly deviate from the mean value, for example, if the user pause their writing at the end of a sentence. The latency between the keystrokes could potentially be minutes or longer, while the normal latency is in milliseconds.

After outlier removal, we recalculate the updated mean and standard deviation. These steps are repeated a few times until the mean and standard deviation are fixed, or we end up with too few data points for it to be sufficient. On keys with few or no data points, we skip this step.

5.4 Template Generation

After extracting and filtering the data, we will store it in two tables as templates for duration and latency respectively. This represent the normal behavior of the user. The template will be structured as seen in table 4.

The duration template will contain one entry for all the possible keys, with the key indentifier, mean and standard deviation. The latency will contain every possible key, squared. This because we have to consider every possible combination of key pairs. Each entry will contain the key pair entry, as well as mean and standard deviation. Due to the size, it is expected that several of the latency pairs will be empty, because the user never type them consecutively in their normal behavior. In particular function keys (F1-F12) and special characters (#$@...) appear to be infrequently used together.

Duration		
Key	Mean	St.Dev
A	137	19
B	154	24
C	208	57
D	232	65
...	...	...

Latencies			
First key	Second key	Mean	St.Dev
A	A	274	38
A	B	193	55
A	C	241	63
A	D	178	72
...	...	...	...

Table 4: Template structure with example data.

We create several templates using varying amounts of keystrokes as the basis. This because we want to check if a larger data set would result in better and more complete template, and because we want to test the USB Rubber Ducky with multiple templates against the system. This is explained further in Chapter 6.

A	A	100	20
A	B	150	30
A	C	0	0
A	D	0	0
A	"	200	50
A	*	250	100
B	!	300	0
C	?	0	0
-	A	300	120
?	B	0	0
F1	@	0	0
F1	$	0	0

Table 5: Illustration of a template with incomplete data.

A	A	100	20
A	B	150	30
A	C	125	25
A	D	125	25
A	"	200	50
A	*	250	100
B	!	300	75
C	?	225	75
-	A	300	120
?	B	300	120
F1	@	nan	nan
F1	$	nan	nan

Table 6: Illustration of template after values have been estimated.

5.5 Data Estimation

We attempted to estimate missing or insufficient values in the template, using existing valid data for the respective category of keystrokes, such as numbers or letters. The estimate is based on the data from the valid timing information within the category. The template before values have been estimated is illustrated in Table 5 and the template with estimated values are illustrated in Table 6. The pseudocode for the data estimation is illustrated in Figure 12, but does not include exception handling such as divide by zero.

For duration, the estimate function calculates an average mean and average standard deviation for the respective groups of keys, such as letters and numbers. We excluded empty values, because they would skew the data, but also values without a standard deviation. This is because we don't want to influence the estimate by values that were typed once.

For the latency, we average the entire pair of groups. Letters to letters, letters to numbers, numbers to letters, etc.. The latency is therefore made up of all the combinations between one group of keys to another, rather than an average between each single key to a whole group of keys.

Cases where there is no comparable data, such as for uncommon combination of groups, function key and special character for instance, the value will be set to "nan" (Not a Number) in the template. We are making the assumption that a fast typist should still type uncommon keys and key combinations, with comparable timing information to their other data, rather than pause and look on the keyboard.

The result is that we fill in unknown values with values that appear close to the normal typing for the user. This can compensate for limited data when creating the template, but is not always sufficient. With very limited data you might not have any entries within a given group of data, and would therefore be unable to estimate any values for it. Similarily, even with large amounts of data, you can still lack entries within certain groups. This because it is not within the normal typing behavior of the user to use certain keys and key combinations.

It should be noted that the estimates within a category are identical, and rounded to a whole number of milliseconds. The actual typing for the different keystrokes could

```
mean equals 0
stDev equals 0
counter equals 0

// Duration only use one of these loops
foreach firstKey in firstGroup
  foreach secondKey in secondGroup

    // Add up the values that occurred more than once.
    if secondGroup.stDev is greater than 0
      add secondGroup.mean to mean
      add secondGroup.stDev to stDev
      increment counter
    }
  }
}

// Average the values
set mean to mean divided by counter
set stDev to stDev divided by counter

where( templateKeyOne in firstGroup
  and templateKeyTwo in secondGroup)

  if templateEntry.mean equals 0
    set templateEntry.mean to mean
  }

  if templateEntry.stDev equals 0
    set templateEntry.stDev to stDev
  }
}
```

Figure 12: Rough pseudocode for the value estimation in the template.

vary quite significantly, and deviate from the estimates. With entries only typed once, we leave the mean value as is, but estimate the standard deviation based on the other values.

Estimating the values in a template could also have a negative effect, if it was applied on the template in a continuous authentication system. This is because uncommon keystrokes are given valid timing information, rather than stand out as activity that is unnatural for the user. An attacker can then forge keystrokes with valid timing, even if the user never type those values into the system. Both of these scenarios will be analyzed in Chapter 7.

6 Testing Methodology

In this chapter we will cover the process of how the tests are run, and what data samples are being used. It will also explain the artificial input that is used to test presentation attacks.

6.1 User Baseline

We will compare the template against genuine user input, as well as multiple imposter users under a zero-effort attack. This to balance the ANGA and ANIA values such that the genuine user remain online on the system as long as possible, while the imposter user is locked out as quickly as possible. The system is tweaked both by adjusting the lockout threshold for the user, as well as how much the trust level changes based on input.

This is done to have a baseline performance to compare against when we run the artificial input as a presentation attack. Without the baseline it would be impossible to tell if the presentation attack performed better or worse than a zero-effort attack where we compare the natural timing information between users.

The baseline imposter data will be from 58 users captured previously over multiple sessions, with varying quality and quantity. For the genuine user, we use three sets of 15.000 keystrokes, captured at different sessions. We also avoid generating templates based on the data captured from the same sessions as the 45.000 keystrokes used for the baseline.

6.2 Artificial Baseline

For the artificial input we have created three templates using approximately 3.000, 9.000 and 15.000 keystrokes. We will use multiple templates to see if the size of the data used would significantly impact the results of the attack. It seems probable that more data used during the creation of the probing text, would lead to better performance during the attack. At the same time, it would seem unreasonable for the attacker to have access to more data to create the attack template, than the continuous authentication system have for authentication.

The templates were used to generate three different text outputs with latency adjustments, each roughly 10.000 keystrokes each. They were generated by a script that took a text and template as input, and generated Ducky Script as output. The output script was again compiled into a payload for the USB Rubber Ducky as described in Section 4.2. The texts are the following:

- **Chapter 1:** The first chapter of this master thesis, which is genuine writing. Noting that there was no template created during the writing of the chapter.
- **German Wiki:** Several paragraphs of the Wikipedia article on biometrics, in German. This is to test using a different language. Appendix A.
- **Lorem Ipsum:** A generated default filltext of Lorem Ipsum, which contain nonsensical words that look Latin. Appendix B.

The goal of using three different texts for the attack is to test if we can falsify text that the valid user would not write, as well as completely nonsensical text. The normal writing should contain keys and key combinations that should appear in the template, while different languages should not. Therefor it should be easier to forge English text for an English user, than to try and forge different language texts, for example German, for an English user.

Each of the three texts will be combined with the three attack templates to provide a total of nine attack texts to use against the continuous authentication system. We will both compare when the attack template used for the attack text is the same as the template used in the continuous authentication system, but also with different templates of different sizes. The argument here would be that the attacker can have compromised the stored template from a remote server, or compromised the user data elsewhere, and generate their own template for the victim.

As mentioned we find it unreasonable that the attacker would hold more data than the continuous authentication system, so worst case scenario the attacker have the same template as the system. As Figure 13 illustrate, we will compare probe texts generated for multiple templates against the continuous authentication system, but will not compare stronger attack template against a weaker continuous authentication system template.

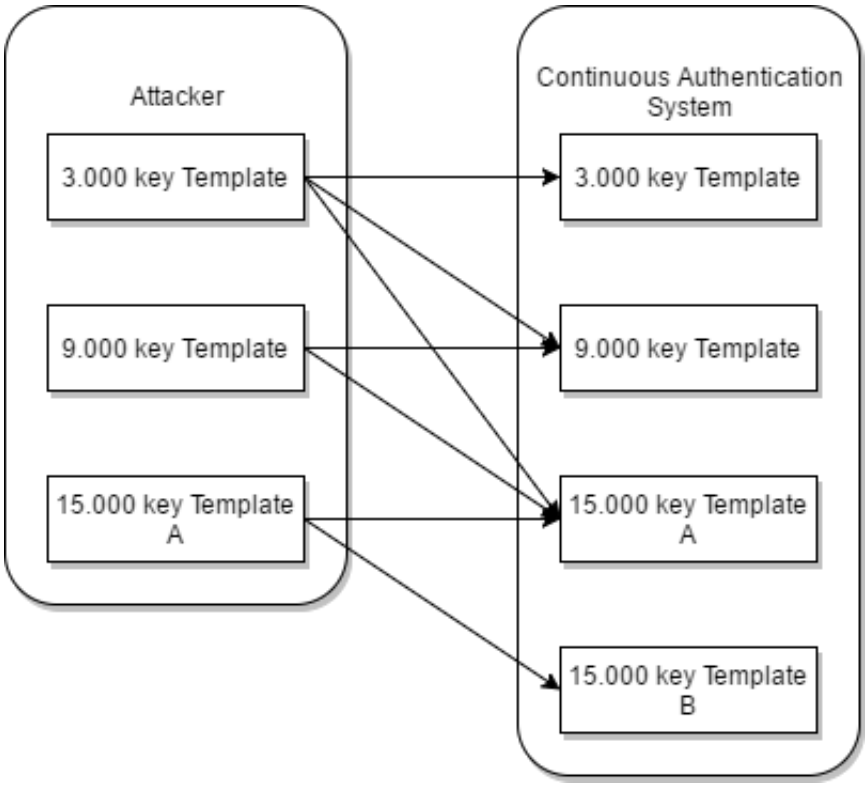

Figure 13: Template interaction between attacker and system

The 15.000 key template from the attacker will both compare against an identical 15.000 key template, as well as one based on different data. The 3.000 key template and the 9.000 key template will compare against identical templates of the same size, or larger templates based on different data.

This should give us an indication if the size of the template on the system significantly reduce the ANIA for the probing text. It should also let us compare if the size of the attack template used for the probing text relate to the ANIA rate.

6.3 Methodology

This section will cover the details of the process in which the tests are run.

The system will contain a template for the user and a given threshold where the user is expected to be within most of the time. The threshold is calculated and tweaked based on user input, both genuine and zero-effort imposter. We attempt to maximize the ANGA, and minimize ANIA.

During runtime, a function will convert each keystroke data input into a duration and latency with the following bigraph. The continuous authentication system will then adjust the trust level based on deviation from the expected value. This will scale based on standard deviation. While the data in this study is not a Gaussian curve, it has been found that the three-sigma-rule is applicable to datasets that deviate from the curve[63]. The three-sigma-rule refer to to how 99.7% of all data fall within three standard deviations away from the mean. Following the three-sigma-rule, roughly 68.3% of the inputs of a valid user should fall within one standard deviation from the mean[64].

We change the trust based on latency and duration separately, meaning that the duration can result in negative change trust, while the latency give a positive change in trust, which add up to no change in trust level. We have four different scenarios that change the trust.

1. If the timing information is within one standard deviation, then we increase the trust level by one.
2. If the timing information is between within one and two standard deviations away from mean, it remain neutral.
3. If the timing information is more than two standard deviations away from the mean, we decrease trust level by one.
4. If the template does not have any values for the keystrokes, the trust remain neutral.

The trust level and threshold is illustrated in Figure 14. When the user goes below the lockout threshold, they will be locked out. After which they will log back in, and the login using password will reset the trust to 100%.

The trust is reset because we assume only the genuine user would be able to log back in to the system using the password.

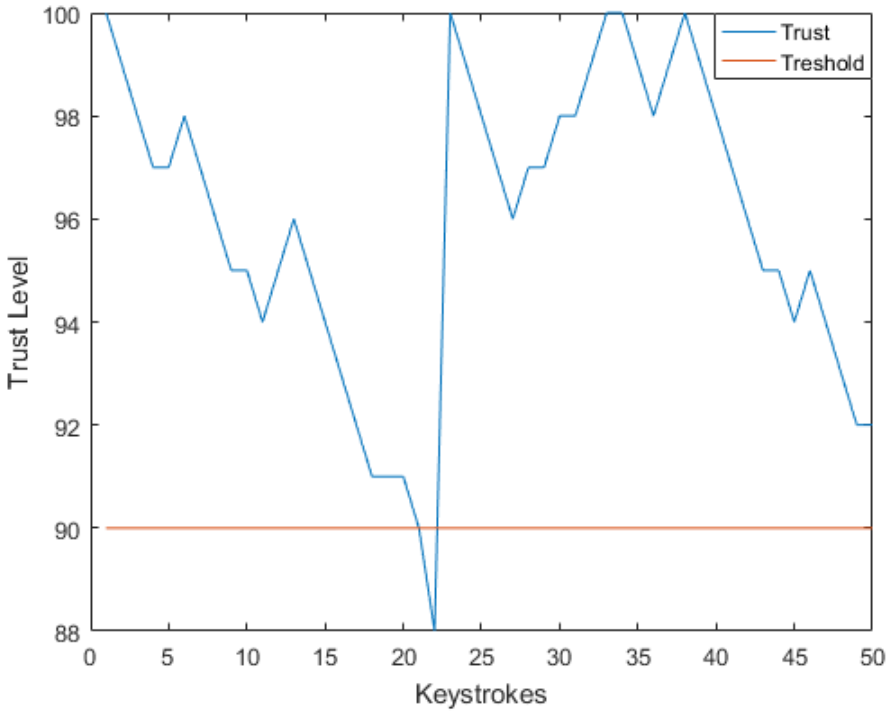

Figure 14: Example of trust level over time.

7 Results and Analysis

In this chapter we will cover the results from the data as well as analyzing the different aspects of it.

7.1 Baseline Performance

We used 15.000 keystrokes each from three different sessions to validate the ANGA. For imposter actions we used varying amounts of data from multiple sessions, from a total of 58 users. This will be the baseline performance of the system prior to targeted attack. In calculating the ANGA we have observed some anomalies. A total of seven instances we observe that the genuine user fell below the threshold, and within 10-20 keystrokes, fell below the treshold again. We have included the anomalies, but argue that the genuine user could adjust to avoid the consecutive lockouts, while an imposter could not. The total number of lockouts were 35, out of which 13 were considered anomalies and 22 were valid lockouts. The anomalies were mostly occurring on the template with fewer keystrokes, and at most we saw five consecutive lockouts within approximately a hundred keystrokes.

This could possibly be linked to burst of typing, such as login credentials, or using an application that trigger different keyboard activity from the user. For example, a game.

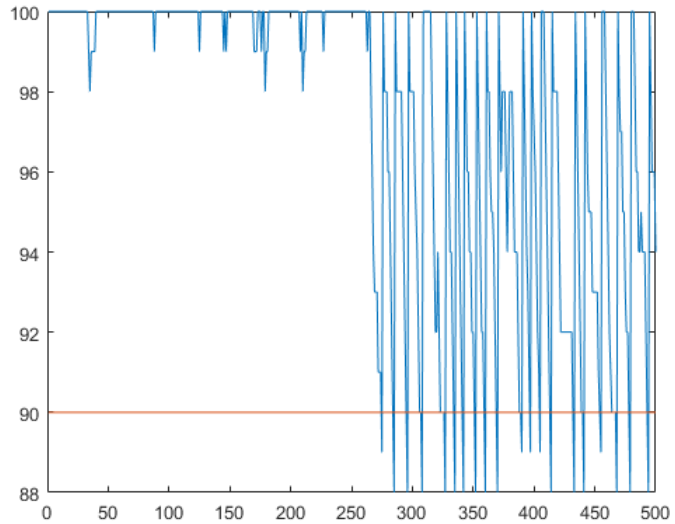

Figure 15: Example of the trust drastically changing based on application.

Anomalies such as the ones we observed can often be explained by looking at the application that actively being used. As Figure 15 illustrate, the application can change the typing behavior of the user quite significantly, and could potentially distort the data. The sixth column of the BeLT data refer back to the application that is open for investigation.

In this case, at around 250 keystrokes, we observe a game being played, which utilize W, A, S, D and Spacebar for movement. This data has significantly longer durations than normal, and latencies vary wildly between keystrokes.

A complete continuous authentication system would have to account for the different applications when adjusting the trust levels. While it would make sense for a workstation to trigger and lock out the user for playing games, there could be enough differences between programming and report writing to warrant separate templates. Alternatively when using one template, the mean and standard deviation become more vague as it has to encompass multiple different behaviors in one, which potentially weakens the system against intrusion.

The attacker again would have to create his script based on which application they are utilizing during the attack. If the system uses multiple templates for different programs, then the attacker should also have different templates to correspond to the system in order to increase the success rate of an attack. Several applications can be grouped together into one template if the typing behavior between them are similar enough.

How many templates the attacker need to create depend on what the attacker wishes to achieve. If the attacker only need to forge an email, they would need a template corresponding to the writing of emails. If the attacker also want to inject a backdoor into some code for a programmer, they would need an additional template for the typing behavior of the user when they are programming.

Table 7 show the baseline performance of average number of genuine and imposter actions for the various templates when we consider both latency and duration in authenticating the user. In 15.000 Template B we observed no instances where the trust level went below the threshold, and the average number of genuine user actions would therefore be above 45.000 keystrokes. In theory, this would give an infinite number of keystrokes before lockout, as such we will exclude it from further operations, and put "nan" in the table.

Template	ANGA	ANIA
3.000 Template	5000	993
9.000 Template	6429	674
15.000 Template A	22500	1278
15.000 Template B	nan	1682

Table 7: Baseline ANGA and ANIA when looking at latecy and duration.

We did observe that four imposter users did not get locked out when considering duration and latency. Two of the users had few keystrokes in total, 4.149 and 2.579, and this could likely explain the lack of lockouts. While the other two users had 25.508 and 14.087 keystrokes respectively. These would likely be considered anomalies, and they show significantly better performance than the majority of the users. It is very likely that the typing behavior of these two users are very similar to the genuine user, and therefore they do not get locked out like normally. These non-lockout cases have been excluded when we averaged the ANIA across the imposter users, as they have a theoretical infinite ANIA score by themselves.

Since the USB Rubbery Ducky is only capable of mimicking the latency out of the box, we will also analyze the impact when the system only considers the latency. We make the

assumption that forged values for duration would be comparable to the forged latency values in regards to the trust level in the continuous authentication. Table 8 shows the average number of genuine and imposter actions when we only consider latency. With the 9.000 template appearing to perform noticeably better in both ANGA and ANIA.

Template	ANGA	ANIA
3.000 Template	3750	657
9.000 Template	45000	283
15.000 Template A	15000	896
15.000 Template B	45000	716

Table 8: Baseline ANGA and ANIA when only looking at latecy.

We see from Figure 16 that the genuine user has comparable trust level when only considering the latency, compared to latency and duration. The Figure is based on the genuine user's input and is a cropped version that best illustrate the similarity. The spikes in trust mostly appear in similar places, with minor exceptions where the duration would repeatedly decrease or increase the trust over several keystrokes. Note that the larger spike seen here is not uncommon, overall we observe similar spikes throughout the baseline performance for the genuine user. Both the trust calculated using latency only, and using duration and latency, show good ANGA and ANIA performance.

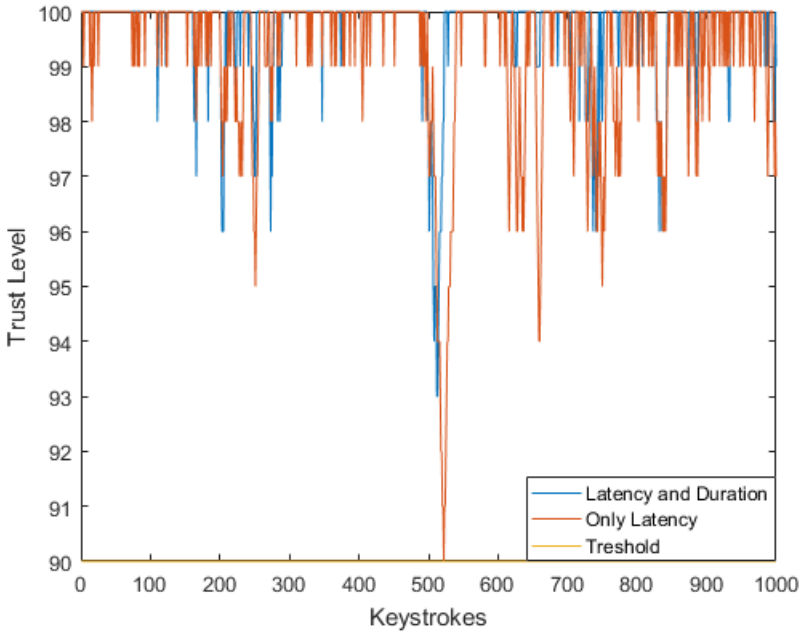

Figure 16: Comparison of trust levels for the 9.000 template ANGA baseline.

7.2 USB Rubber Ducky Attack

This section will cover the data we got when injecting the system with keystrokes using the USB Rubber Ducky. First we look at the data when the continuous authentication system is looking for both duration and latency, and then when we look at latency only.

7.2.1 Latency and Duration

When we are looking at both the duration and latency, we are actively looking at an aspect that the USB Rubber Ducky is unable to forge during this project. Since the data is artificial input, we observe durations of zero milliseconds. This is impossible for real input, so the template will only count negatively for the duration, with the exception of unknown keys where the template defaults to a zero change in trust.

The latency is then almost entirely responsible for positive adjustments to the trust to avoid falling below the threshold. With a positive score from the latency, we would observe that the trust level remain in place most of the time. We observe few instances where the trust level increased, but it mostly decreased until lockout within a few keystrokes.

As illustrated in Figure 13, we run several probing texts using different templates and text strings, against the different templates on the continuous authentication system. Table 9, 10 and 11 shows the ANIA for the different attack texts and template configurations.

Using the 3.000 template to generate the probing text, we see that the performance is reasonably stable for the three texts. The ANIA is around 30-35 keystrokes before lockout. The little change in performance when attacking the 9.000 and 15.000 A template could be explained by insufficient data in the attack template.

3.000 Template	Continuous Authentication System		
	3.000 Template	9.000 Template	15.000 Template A
German Wiki	45.14	33.58	51.15
Lorem Ipsum	32.43	26.61	30.10
Master Thesis	30.83	26.37	35.99

Table 9: ANIA for the 3.000 attack template

The 9.000 template show higher ANIA before lockout when attacking a system using the same template as basis, but drops down significantly when we compare against a template of 15.000 different keystrokes. This is likely due to the continuous authentication having different and more complete data compared to the probing text.

9.000 Template	Continuous Authentication System	
	9.000 Template	15.000 Template A
German Wiki	199.10	71.93
Lorem Ipsum	105.86	46.52
Master Thesis	55.16	44.37

Table 10: ANIA for the 9.000 attack template

With the 15.000 template based attack text we compared first against the identical template in the continuous authentication system, and then against a different template of equal size. We again observe that the attacker manages more actions before being locked out when they use the same template as the system, with the German Wiki probing text managing significantly more keystrokes than the other texts. When the system is using a different template, the average number of actions drop down to around 33 across all probing texts.

15.000 Template A	Continuous Authentication System	
	15.000 Template A	15.000 Template B
German Wiki	1238.89	38.19
Lorem Ipsum	167.46	31.12
Master Thesis	92.14	30.16

Table 11: ANIA for the 15.000 A attack template

We observe that the attacks perform better when injecting text that deviate from the genuine user's typing. This could be linked to the estimation we described in Section 5.5. By estimating the values of keystrokes that the user normally do not type, we are creating valid timing information for the attacker to use. Since they are estimates, they also have a broader standard deviation value, while the genuine text that the user write is more narrow.

We also observe that attacking the system using the same template as on the system typically give better performance than when the system is using a different template for continuous authentication.

While there are small attack scripts for USB Rubber Ducky that can cause damage in less than 30 keystrokes, the majority require more keystrokes and will therefore be blocked on average before the damage is done.

7.2.2 Latency Only

When we look at only the latency the performance of the forged keystrokes are entirely different. Since the system is only comparing the values that the USB Rubber Ducky can manipulate, it naturally performs better.

While we are unable to test using a modified duration at this time, the duration should perform comparable to the latency if the template data is similarily reliable in comparison to the continuous authentication system.

We did not observe any lockouts with any of the probe texts, across any of the template configurations. This would indicate that an attacker could observe the genuine user and generate a template that is much smaller than the continuous authentication system, and still attack it without being locked out. In our case, if the attacker used the 3.000 template, they could successfully bypass the system even if they relied on a 15.000 template for detection.

7.3 Presentation Attack Detection

This section will cover some of the potential presentation attack detection techniques, and the result of their implementation.

7.3.1 Unnaturally High Trust Level

We started by trying to see if the presentation attack had an unnaturally high trust level in comparison to the genuine user. If the trust level of an attacker constantly have a trust level of 100, then it could be an anomaly that we could use to detect ongoing attacks. For example, if the attacker always inject keystrokes with perfect value according to the mean value of the template.

In our case, the injected keystrokes would randomly deviate from the mean value in the attack template, by up to one standard deviation. Mitigating such a detection

Genuine User	Mean value	Standard Deviation
3.000 Template	99.66	0.897
9.000 Template	99.68	0.801
15.000 Template A	99.77	0.651
15.000 Template B	99.80	0.578

Table 12: Mean and standard deviation of the trust level for the genuine user

technique would naturally be to increase how much the injected keystroke deviate from the mean value, until the trust level compare to the genuine user.

Table 12 show the mean and standard deviation for the genuine user, while Table 13 show the mean value for attack between the various templates.

We observe that the genuine user have quite high trust level on average. To verify this, we took several smaller samples of the genuine user's typing and calculated the mean and standard deviation for the samples. They revealed similar values with smaller fluctuations

We also observe that the attacker have comparably high trust level on average, with the difference between genuine user and presentation attack being as little as 0.02 in some cases. Well within the standard deviation we observe for the genuine user.

Presentation Attack	Attack Template		
Continuous Authentication Template	3.000 Template	9.000 Template	15.000 Template A
3.000 Template	99.78		
9.000 Template	99.70	99.93	
15.000 Template A	99.75	99.84	99.94
15.000 Template B			99.78

Table 13: Mean trust level for the presentation attacks

We tried to compare the average trust level of input to the system, against what the template consider average and standard deviation. This was done by taking a selection of the previous keystrokes at various intervals, and averaging this. We compared this result to the average and standard deviation we calculated based on the template data. Because of the small difference between the genuine user and the attacks, as well as the variation of the genuine user, we were unable to reliably distinguish between genuine input and an ongoing presentation attack.

7.3.2 Reduce Trust on Unknown Values

We could adjust the trust negatively if the input comes for keys that the continuous authentication template does not contain. The argument is that the data unknown to the template does not represent the genuine user's typing behavior, and should therefore improve resistance against forged keystrokes. The zero-effort imposter attacks would also get a reduced ANIA score due to how values that previously were neutral now count negatively. The tradeoff is that the genuine user also have their performance reduced.

When changing the trust level with -1 for unknown keystroke values, we see significantly more lockouts from the genuine user. Using the same 45.000 keystrokes for validation of templates as for the baseline. This time, the templates on the continuous authentication system does not attempt to estimate missing duration or latency values.

From Table 15 we see that the system become borderline unusable for the genuine user when we reduce trust when they type uncommon keys.

	Duration and Latency		Latency Only	
Template	Lockouts	ANGA	Lockouts	ANGA
3.000 template	146	308	389	116
9.000 Template	80	562	92	489
15.000 Template A	86	523	90	500
15.000 Template B	28	1607	34	1323

Table 14: Lockouts and ANGA when reducing trust for unknown values.

The Table does show that a bigger template yield slightly better performance, probably due to more uncommon keystrokes being featured in the data source for the template compared to templates with a smaller data base. The reduced number of lockouts going between 15.000 Template A and B would suggest that the data source affect the performance of the system.

This technique could partically work as a language check, because of how different languages would have different keystroke bigrams occuring at different frequencies. Depending on the sources, the ten most common bigrams in the english language occur between 8.7%[1] and 21%[2] of the time.

We could then analyze the frequence of the more common bigrams as a way to indicate which language is being written, and to detect when the user is not writing their normal language. Table 15 show the ten most common bigrams and their percentage frequency in different languages, according to Practical Cryptography[65]. For example, the two most common bigrams in the English language are not among the top ten bigrams for German or Swedish.

English	
TH :	2.71
HE :	2.33
IN :	2.03
ER :	1.78
AN :	1.61
RE :	1.41
ES :	1.32
ON :	1.32
ST :	1.25
NT :	1.17

German	
ER :	3.90
EN :	3.61
CH :	2.36
DE :	2.31
EI :	1.98
TE :	1.98
IN :	1.71
ND :	1.68
IE :	1.48
GE :	1.45

Swedish	
EN :	2.44
DE :	2.11
ER :	2.10
AN :	1.75
AR :	1.61
ET :	1.27
ST :	1.27
IN :	1.22
RA :	1.21
TE :	1.18

Table 15: Comparison of 10 most common bigrams for English, German and Swedish

[1] "Bigram frequency in the English language", Wikipedia, Retrieved May 29, 2017
[2] "English Letter Frequency Counts: Mayzner Revisited", Norvig.com, Retrieved May 29, 2017

8 Conclusion And Future Work

This chapter will cover the concluding remarks as well as the future work to be done within the field.

A brief summary of the project is as follows. We attacked a continuous authentication system relying on four different templates for security, using three different attack templates. To see if the language of the attack text had any impact on the performance, we used three different texts in English, German and latin-like filltext. This was done using a tool called USB Rubber Ducky, which is a commerically available penetration testing tool.

8.1 Conclusions of Research

We have split the conclusions into two parts. One covering the status against most conventional attacks, and one against the theoretical capacity of a targeted attack. The first is based on the results when the continuous authentication system analyze both the duration and latency of the keystrokes, with the attacker only able to forge the latency as we tested in Section 7.2.1; and the second part is based on the assumption that the attacker can forge both duration and latency values, and the forged duration values perform comparable to the latency as observed in Section 7.2.2.

8.1.1 Out of the Box

Without any modification to the source code for the Ducky Encoder, the USB Rubber Ducky appears to only manage 30-50 keystrokes before being locked out. Without adjusting the latency of the injected keystrokes, we expect the performance to be significantly worse.

Considering that most systems are vulnerable to attacks performed by the USB Rubber Ducky, we find this a strong advantage and argument for the implementation of continuous keystroke dynamics based authentication. We consider the tradeoffs of being locked out after several thousand keystrokes to be in favour of the defence it provides.

In this case we found keystroke dynamics to be secure against presentation attacks, without any need for presentation attack detection.

8.1.2 Resourceful Attacker

As mentioned, time limitation prevented us from testing an attack where both the duration and latency were modified according to the user template. The code in the Ducky Script compiler would suggest that this is feasible given time.

Based on the tests we did in Section 7.2.2, we find that the attacker can inject keystrokes that perform better than the genuine user. If we assume that modifying the duration yield comparable performance on a system considering both duration and latency, then the system is not secure against presentation attacks.

The few methods we proposed in Section 7.3 can easily be mitigated, and does not provide good enough performance to be viable at both securing the system and making it accessible to the genuine user. We are therefore unable to detect a strong presentation

attack in keystroke dynamics.

Unnaturally High Trust

We found that the trust level between the genuine user and the injected keystrokes were very close. While it could be possible to detect patterns that reveal an attack, we find it unlikely. Detection of such an attack would likely require knowledge about the pattern of an attacker as well as the genuine users.

This means of detection could potentially also be mitigated by the attacker, simply by adapting the variance of injected keystrokes such that the trust levels and patterns more closely resemble the genuine user.

Uncommon Keystrokes and Linguistic check

The approach of reducing the trust when the user types keystrokes that the continuous authentication system do not have within the template is one approach. This would partically work as a language check, because the most common bigrams should change for a different languages. Less frequent bigrams might not be present in a template based on a smaller sample size. In particular when few bigrams make up a significant amount of the total keystrokes[66].

Despite a linguistic check probably being applicable against foreign languages, it would still let an attacker inject English text for an English user.

8.2 Future Work

We would like to see research done into attacking the system with the proper manipulation of duration and latency of a keystroke. We have made the assumption that the duration will yield performance comparable to the latency and bypass the current security of continuous keystroke based authentication. The reality could be different.

There should also be some investigation into the ease and accuracy of aquiring the keystrokes from the user, such as to generate an attack template. Currently there are few, if any, systems that an attacker can compromise to aquire a template of the victim's typing behavior. Ongoing research on securing biometric templates should prevent compromising biometric templates for attack purposes. Infecting the user's system with a keylogger is possible, but would require compromising the user's system before the attack.

Bibliography

[1] Ratha, N. K., Connell, J. H., & Bolle, R. M. 2001. An analysis of minutiae matching strength. In *Audio- and Video-Based Biometric Person Authentication*, volume 2091, 223 – 228.

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

[3] Tandberg, Y. & rune Bjørneseth. May 2015. Typing mimicking robot. Bachelor Thesis at GUC.

[4] Roth, J., Liu, X., & Metaxas, D. 2014. On continuous user authentication via typing behavior. *IEEE Transactions on Image Processing*, 23, 4611 – 4624.

[5] Komanpally, V. Mimicking of keystroke dynamics. Master's thesis, Gjovik University College, 2014.

[6] Meng, T. C., Gupta, P., & Gao, D. 2008. I can be you: Questioning the use of keystroke dynamics as biometrics.

[7] Rundhaug, F. E. N. Keystroke dynamics - can attattack learn someone's typing characteristics. Master's thesis, Gjøvik University College, 2007.

[8] Galbally, J. & Gomez-Barrero, M. 2016. A review of iris anti-spoofing. In *2016 4th International Conference on Biometrics and Forensics (IWBF)*.

[9] Boulkenafet, Z., Komulainen, J., & Hadid, A. 2016. Face spoofing detection using colour texture analysis. *IEEE Transactions on Information Forensics and Security*, 11, 1818 – 1830.

[10] Venugopalan, S., Juefei-Xu, F., & Cowley, B. 2015. Electromyograph and keystroke dynamics for spoof-resistant biometric authentication. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

[11] TABULA-RASA. Trusted biometrics under spoofing attacks. http://www.tabularasa-euproject.org/.

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

[13] Khan, S. H., Akbar, M. A., Shahzad, F., Farooq, M., & Khan, Z. 2015. Secure biometric template generation for multi-factor authentication. *Pattern Recognition*, 48, 458 – 472.

[14] Sun, J., Zhang, R., Zhang, J., & Zhang, Y. 2014. Touchin: Sightless two-factor authentication on multi-touch mobile devices. In *IEEE Conference on Communications and Network Security*.

[15] Paseenchuk, V. A. & Volkov, D. A. 2016. Signtologin cloud service of biometric two-factor authentication using mobile devices. In *17th International Conference on Micro/Nanotechnologies and Electron Devices*.

[16] Luo, H. & Henry, P. september 2013. A common password method for protection of multiple accounts. In *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications*.

[17] Awad, M., Al-Qudah, Z., Idwan, S., & Jallad, A. H. December 2016. Password security: Password behavior analysis at a small university. In *5th International Conference on Electronic Devices, Systems and Applications*.

[18] Shahzad, M., Liu, A. X., & Samuel, A. 2013. Secure unlocking of mobile touch screen devices by simple gestures - you can see it but you can not do it. In *13th Proceedings of the 19th annual international conference on Mobile computing & networking*, 39 – 50.

[19] Aviv, A. J., Gibson, K., Mossop, E., Blaze, M., & Smith, J. M. 2010. Smudge attacks on smartphone touch screens. In *Proceedings of the 4th USENIX Conference on Offensive Technologies*, WOOT'10, 1–7, Berkeley, CA, USA. USENIX Association.

[20] Jain, A., Ross, A., & Pankanti, S. June 2006. Biometrics: a tool for information security. *IEEE Transactions on Information Forensics and Security*, 1, 125 – 143.

[21] Jiang, L., Li, X., Cheng, L. L., & Guo, D. October 2013. Identity authentication scheme of cloud storage for user anonymity via usb token. In *IEEE International Conference on Anti-Counterfeiting, Security and Identification*.

[22] Tanvi, P., Sonal, G., & Kumar, S. M. 2011. Token based authentication using mobile phone. In *International Conference on Communication Systems and Network Technologies*.

[23] Schuckers, M. E. 2010. False match rate. In *Computational Methods in Biometric Authentication*, 97–153. Springer.

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

[25] Maltoni, D. & Cappelli, R. 2009. Advances in fingerprint modeling. *Image and Vision Computing*, 27, 258 – 268.

[26] Tan, B. & Schuckers, S. 2010. Spoofing protection for fingerprint scanner by fusing ridge signal and valley noise. *Pattern Recognition*, 43, 2845 – 2857.

[27] Kastaniotis, D., Theodorakopoulos, I., Theoharatos, C., Economou, G., & Fotopoulos, S. 2015. A framework for gait-based recognition using kinect. *Pattern Recognition Letters*, 68, 327 – 335.

[28] Fathima, S. M. H. S. S. & Valanarasi, A. May 2016. Human gait recognition using relevance vector machine classifier. In *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*. IEEE.

[29] Fernandez-Lopez, P., Liu-Jimenez, J., Sanchez-Redondo, C., & Sanchez-Reillo, R. Oct 2016. Gait recognition using smartphone. In *2016 IEEE International Carnahan Conference on Security Technology*. IEEE.

[30] Leyden, K., Koller, M., Niemier, M., Schmiedeler, J., & Cserey, G. Aug 2016. Kinect image processing by cnn algorithm for gait recognition. In *15th International Workshop on Cellular Nanoscale Networks and their Applications*. VDE.

[31] Singh, S. & Yamini, M. July 2013. Voice based login authentication for linux. In *International Conference on Recent Trends in Information Technology*. IEEE.

[32] Johnson, R. C., Boult, T. E., & Scheirer, W. J. Oct 2013. Voice authentication using short phrases: Examining accuracy, security and privacy issues. In *Sixth International Conference on Biometrics: Theory, Applications and Systems*. IEEE.

[33] Yan, Z. & Zhao, S. Aug 2016. A usable authentication system based on personal voice challenge. In *2016 International Conference on Advanced Cloud and Big Data*. IEEE.

[34] Abdulaziz Alzubaidi, J. K. 2016. Authentication of smartphone users using behavioral biometrics. *IEEE Communications Surveys & Tutorials*, 18, 1998 – 2026.

[35] Corpus, K. R., Gonzales, R. J. D., Morada, A. S., & Vea, L. A. May 2016. Mobile user identification through authentication using keystroke dynamics and accelerometer biometrics. In *IEEE/ACM International Conference on Mobile Software Engineering and Systems*. IEEE.

[36] Ceker, H. & Upadhyaya, S. Sept 2016. User authentication with keystroke dynamics in long-text data. In *IEEE 8th International Conference on Biometrics Theory, Applications and Systems*. IEEE.

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

[38] Hadid, A., Ghahramani, M., & Kellokumpu, V. 2012. Can gait biometrics be spoofed? In *21st International Conference on Pattern Recognition*.

[39] Auvinet, E., Multon, F., Aubin, C.-E., Meunier, J., & Raison, M. 2015. Detection of gait cycles in treadmill walking using a kinect. *Gait and Posture*, 41, 722 – 725.

[40] Derawi, M. & Bours, P. 2013. Gait and activity recognition using commercial phones. *Computers & Security*, 39, 137 – 144.

[41] Ergunay, S. K., Khoury, E., Lazaridis, A., & Marcel, S. 2015. On the vulnerability of speaker verification to realistic voice spoofing. In *IEEE 7th International Conference on Biometrics Theory, Applications and Systems*.

[42] Wu, Z. & Li, H. 2013. Voice conversion and spoofing attack on speaker verification systems. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*.

[43] Alegre, F., Amehraye, A., & Evans, N. 2013. Spoofing countermeasures to protect automatic speaker verification from voice conversion. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

[44] Ballard, L., Monrose, F., & Lopresti, D. 2006. Biometric authentication revisited: Understanding the impact of wolves in sheep's clothing. In *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, USENIX-SS'06, Berkeley, CA, USA. USENIX Association.

[45] Mondal, S. & Bours, P. 2015. A computational approach to the continuous authentication biometric system. *Information Sciences*, 304, 28 – 53.

[46] Pfost, J. 2007. The science behind keystroke dynamics. *Biometric Technology Today*, 15, 7.

[47] Alves, D. D., Cruz, G., & Vinhal, C. 2014. Authentication system using behavioral biometrics through keystroke dynamics. In *Computational Intelligence in Biometrics and Identity Management (CIBIM), 2014 IEEE Symposium on*.

[48] Karnan, M., Akila, M., & Krishnaraj, N. 2011. Biometric personal authentication using keystroke dynamics: A review. *Applied Soft Computing*, 11, 1565 – 1573.

[49] Bours, P. & Komanpally, V. 2014. Performance of keystroke dynamics when allowing typing corrections. In *International Workshop on Biometrics and Forensics (IWBF)*.

[50] Killourhy, K. S. & roy A. Maxion. 2009. Comparing anomaly-detection algorithms for keystroke dynamics. In *International Conference on Dependable Systems & Networks*.

[51] Araujo, L. C. F., Sucupira, L. H. R., Lizarraga, M. G., Ling, L. L., & Yabu-uti., J. B. T. 2004. User authentication through typing biometrics features. In *Proceedings of the 1st International Conference on Biometric Authentication*, volume 3071, 694 – 700.

[52] Bleha, S., Slivinsky, C., & Hussien, B. 1990. Computeraccess security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12, 1217 – 1222.

[53] Duda, R. O., Hart, P. E., & Stork, D. G. 2000. *Pattern Classification*. John Wiley & Sons, Inc.

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

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

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

[57] Rahman, K. A., Balagani, K. S., & Phoha, V. V. 2013. Snoop-forge-replay attacks on continuous verification with keystrokes. *IEEE Transactions on Information Forensics and Security*, 8, 528 – 541.

[58] Monaco, J. V., Ali, M. L., & Tappert, C. C. 2015. Spoofing key-press latencies with a generative keystroke dynamics model. In *IEEE 7th International Conference on Biometrics Theory, Applications and Systems*.

[59] Killourhy, K. & Maxion, R. 2010. Why did my detector do that?! In *Recent Advances in Intrusion Detection*, 256 – 276.

[60] Stefan, D., Shu, X., & Yao, D. D. 2012. Robustness of keystroke-dynamics based biometrics against synthetic forgeries. *Computers & Security*, 31, 109 – 121.

[61] Rahman, K. A., Balagani, K. S., & Phoha, V. V. June 2011. Making impostor pass rates meaningless: A case of snoop-forge-replay attack on continuous cyber-behavioral verification with keystrokes. In *2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE.

[62] Mondal, S. & Bours, P. March 2017. A study on continuous authentication using a combination of keystroke and mouse biometrics. *Neurocomputing*, 230, 1 – 22.

[63] Czitrom, V. & Spagon, P. D. 1997. *Statistical Case Studies for Industrial Process Improvement (ASA-SIAM Series on Statistics and Applied Probability)*. Society for Industrial and Applied Mathematics.

[64] Grafarend, E. & Awange, J. L. 2012. *Applications of Linear and Nonlinear Models: Fixed Effects, Random Effects, and Total Least Squares*. Springer Geophysics.

[65] Letter frequencies for various languages.

[66] Jones, M. N. & Mewhort, D. J. K. 2004. Case-sensitive letter and bigram frequency counts from large-scale english corpora. *Behavior Research Methods, Instruments, and Computers*, 36(3), 388 – 396.

A German Wikipedia on Biometrics

Die Biometrie (auch Biometrik - von altgriechisch Ssioc bios "Leben" und uetpov metron "Mass, Massstab") ist eine Wissenschaft, die sich mit Messungen an Lebewesen und den dazu erforderlichen Mess- und Auswerteverfahren beschaeftigt.

Je nach Anwendungsbereich gibt es unterschiedliche Detaildefinitionen. Christoph Bernoulli benutzte 1841 als einer der ersten Wissenschaftler den Begriff Biometrie in einer sehr woertlichen Interpretation fuer die Messung und statistische Auswertung der menschlichen Lebensdauer.

Der Begriff der Biometrie besitzt die zwei Facetten der biometrischen Statistik und der biometrischen Erkennungsverfahren, die auch in der Praxis getrennt sind.

Bei biometrischer Statistik geht es um die Entwicklung und Anwendung statistischer Methoden zur Auswertung von Messungen aller Art an lebenden Wesen. Sie wird intensiv von allen Lebenswissenschaften genutzt. Wegbereiter der wissenschaftlichen Methodik war Karl Pearson (1857-1936). In diesem Kontext wird Biometrie auch als Synonym fuer Biostatistik verwendet.

Als Erkennungsverfahren setzte man schon frueh die Biometrie zur Personenidentifikation ein. So entwickelte Alphonse Bertillon 1879 ein spaeter Bertillonage genanntes System zur Identitaetsfeststellung, das auf 11 Koerperlaengenmassen basierte (Anthropometrie). 1892 legte Francis Galton den wissenschaftlichen Grundstein fuer die Nutzung des Fingerabdrucks(Daktyloskopie).

Heute definiert man Biometrie im Bereich der Personenerkennung auch als automatisierte Erkennung von Individuen, basierend auf ihren Verhaltens- und biologischen Charakteristika.

Weitere Anwendungsgebiete der Biometrie sind beispielsweise automatisierte Krankheits-Diagnoseverfahren.

Biometrie lebt vom Zusammenspiel der Disziplinen Lebenswissenschaften, Statistik, Mathematik und Informatik. Erst die heutige Informationstechnologie macht es moeglich, die hohen Rechenleistungsanforderungen ueblicher biometrischer Verfahren zu bewaeltigen.

Biometrische Statistik

Hauptartikel: Biostatistik Biometrie als Entwicklung und Anwendung statistischer Methoden im Rahmen empirischer Untersuchungen an Lebewesen dient dem wissenschaftlichen Erkenntnisgewinn, der Entscheidungsfindung und der wirtschaftlichen Optimierung von Produkten. Hier einige Beispiele:

- Biologie
- Epidemiologie: Erforschung von Krankheitsursachen, Verbreitungswegen und Umwelteinfluessen, z. B. zur Unterstuetzung einer effektiven Gesundheitspolitik und Krankheitsvorbeugung
- Forstwirtschaft

- Genetik: Untersuchung der genetischen Komponenten von Krankheiten zur besseren Vorbeugung und Steigerung der Heilungschancen
- Landwirtschaft: Futtermittelentwicklung und -optimierung; Pflanzenzucht, Ertragsoptimierungen in Abhaengigkeit von Umweltparametern
- Medizin: Ermittlung von Risikofaktoren bei bestimmten Krankheiten; Klinische Studien im Vorfeld von Arzneimittelzulassungen zur Bestimmung von Wirkungen und Nebenwirkungen, Bewertung des Nutzen-Risiko-Verhaeltnisses
- Versicherungsmathematik: Berechnung und Prognose der relevanten Parameter fuer Lebensversicherer, z. B. der Sterbetafel.
- Veterinaermedizin: Abbauverhalten von Arzneimitteln; Erforschung von Krankheitsursachen, Verbreitungswegen und Umwelteinfluessen

Biometrische Erkennungsverfahren

Biometrische Erkennungsmethoden haben in den letzten Jahren einen enormen Aufschwung erlebt. Der technologische Fortschritt erlaubt in zunehmendem Masse die rasche Messung biologischer Charakteristika und deren Auswertung mit vertretbarem Aufwand und hoher Qualitaet. Der Einsatz von Biometrie ist ein vielversprechender Ansatz, das ungeloeste Problem vieler Sicherheitskonzepte zu loesen: Wie verbindet man Identitaeten und die dazugehoerigen Rechte mit den die richtige Identitaet aufweisenden physischen Personen?

Das im Jahr 2001 in Australien gegruendete Biometrics Institute hat satzungsgemaess die Aufgabe, den verantwortungsbewussten Gebrauch von biometrischen Technologien zu foerdern.

Biometrische Charakteristika

Beim Einsatz der Biometrie zur automatisierten Erkennung von Personen kommt es darauf an, individuelle biometrische Verhaltens- oder Koerpercharakteristika zu finden, die sich u. a. durch folgende Eigenschaften auszeichnen:

- Einmaligkeit: Der Messwert des Charakteristikums ist fuer moeglichst alle Personen unterschiedlich
- Konstanz: Der Messwert haengt nicht vom Alter der Person oder dem Messzeitpunkt ab
- Messbarkeit: Es sollte eine gut definierbare Messgroesse existieren, fuer die es geeignete Sensoren gibt
- Universalitaet: Das Charakteristikum kommt bei moeglichst vielen Personen vor.

Biometrische Charakteristika werden haeufig unterschieden in aktiv/passiv, verhaltens-/physiologiebasiert oder dynamisch/statisch. Zu den langfristig stabilen verhaltensbasierten Charakteristika zaehlen die Stimme, die Hand- oder Unterschrift, das Tippverhalten und die Gangdynamik. Langfristig stabile physiologische Charakteristika sind beispielsweise der Fingerabdruck, die Iris oder die Handgeometrie. Diese Unterscheidung ist zwar

weitgehend akzeptiert, es existieren aber Grenzbereiche. So sind die meisten verhaltensbasierten biometrischen Charakteristika beeinflusst durch die Physiologie, etwa die Stimme durch den Sprachapparat des Menschen.

Als biometrische Charakteristika koennen u. a. verwendet werden:

- DNA (mobiler DNA-Test, genetischer Fingerabdruck)
- Fingerabdruck (Fingerlinienbild)
- Gangstil (engl. automatic gait recognition)
- Gesichtsgeometrie
- Handgeometrie
- Handlinienstruktur
- Handvenenstruktur
- Iris (Regenbogenhaut)
- Koerpergeruch
- Koerpergroesse (Anthropometrie)
- Lippenbewegung, meist im Zusammenhang mit Stimmerkennung (Klangfarbe)
- Nagelbettmuster
- Ohrform
- Retina (Augenhintergrund)
- Stimme (nicht zu verwechseln mit Spracherkennung)
- Tippverhalten auf Tastaturen (engl. keystroke dynamics)
- Unterschrift (statisch, dynamisch, auch Handschrift)
- Zahnabdruck

Realisierung und Funktionsweise

Ein biometrisches Erkennungssystem setzt sich im Wesentlichen aus den Komponenten Sensor (Messwertaufnehmer), Merkmalsextraktion und Merkmalsvergleich zusammen. Welche Arten von Sensoren zum Einsatz kommen, haengt stark vom biometrischen Charakteristikum ab. So ist eine Videokamera fuer die meisten Charakteristika geeignet; fuer die Fingerabdruckerkennung kommen auch andere bildgebende Verfahren in Frage. Die Sensorkomponente liefert als Ergebnis ein biometrisches Sample. Die Merkmalsextraktion entfernt mittels komplexer Algorithmen alle vom Sensor gelieferten Informationen, die nicht die geforderten Merkmalseigenschaften erfuellen und liefert als Ergebnis die biometrischen Merkmale. Der Merkmalsvergleicher errechnet schliesslich einen Vergleichswert (Score) zwischen dem in der Einlernphase gespeicherten biometrischen Template und dem aktuellen, von der Merkmalsextraktion gelieferten Datensatz. Ueber- bzw. unterschreitet dieser Vergleichswert eine (einstellbare) Schwelle, gilt die Erkennung als erfolgreich.

In der "Einlernphase", dem Enrolment, werden die biometrischen Merkmalsdaten als

Referenzmuster in digitaler Form verschluesselt abgespeichert. Beim naechsten Kontakt mit dem biometrischen System wird ein aktuelles Sample aufgenommen und mit dem Referenzmuster (Template) verglichen. Das System entscheidet dann, ob die Aehnlichkeit der beiden Muster hinreichend hoch ist und damit beispielsweise ein Zutritt erfolgen darf oder nicht.

Die wichtigsten Erkennungsarten sind die Verifikation und die Identifikation. Bei der Verifikation muss die zu verifizierende Person dem System zunaechst ihren Namen oder ihre User-ID mitteilen. Danach entscheidet das biometrische System, ob die Person zum zugehoerigen Referenzmerkmalsdatensatz gehoert oder nicht. Bei der Identifikation offenbart die zu erkennende Person ausschliesslich ihr biometrisches Charakteristikum, das System ermittelt daraus durch Vergleich mit den Referenzmerkmalsdatensaetzen aller Nutzer den zugehoerigen Namen bzw. die User-ID.

Leistungskriterien

Da die vom biometrischen Sensor gelieferten Samples starken statistischen Schwankungen unterliegen, kann es gelegentlich zu Falscherkennungen kommen. Die Zuverlaessigkeit der Identifikation bzw. Verifikation wird hauptsaechlich nach zwei Kriterien beurteilt: nach der Zulassungsrate Unberechtigter und nach der Abweisungsrate Berechtigter:

- Falschakzeptanzrate (FAR) = Zulassungsrate Unberechtigter
- Falschrueckweisungsrate (FRR) = Abweisungsrate Berechtigter

Beide Raten haengen gegenlaeufig vom Entscheidungsschwellwert ab: Eine hoeher gewaehlte Schwelle verringert zwar die FAR, erhoeht zugleich aber die FRR und umgekehrt. Deshalb ergibt z. B. die alleinige Angabe der FAR ohne zugehoerige FRR keinen Sinn. Bei einer FRR von 10 % kann die (Verifikations-) FAR bei guten biometrischen Systemen je nach Charakteristikum Werte von 0,1 % bis < 0,000001 % erreichen.

Waehrend die FAR bei Verfikationssystemen bei gegebener Entscheidungsschwelle eine Konstante ist, waechst sie bei Identifikationssystemen mit der Zahl der gespeicherten Referenzdatensaetze. Naeherungsweise ergibt sich die resultierende Gesamt-FAR aus der Multiplikation der zugrunde liegenden Verifikations-FAR mit der Zahl der Datensaetze. Dies ist der Grund, warum nur stark distinktive Charakteristika wie Iris und Zehnfingerprint eine zuverlaessige Identifikation ueber grosse Datenbasen mit Millionen von Eintraegen ermoeglichen.

Schliesslich beschreibt die

- Falschenrolmentrate (FER) = Rate erfolgloser Enrolments

den Umstand, dass nicht jedes biometrische Charakteristikum bei jedem Menschen jederzeit in ausreichender Qualitaet zur Verfuegung steht. Die FER haengt nicht nur von der jeweiligen Verfassung des biometrischen Charakteristikums ab, sie wird wie die anderen Fehlerraten auch durch die Leistungsfaehigkeit der Technik und die Mitwirkung der enrolten Testperson beeinflusst.

In der Regel lassen sich die beschriebenen Fehlerraten nicht theoretisch berechnen, sondern sind in aufwaendigen statistischen Untersuchungen zu ermitteln. Dabei steigt der Aufwand mit abnehmenden Fehlerraten umgekehrt proportional an. Verfahren zur Leistungspruefung und -auswertung fuer biometrische Systeme beschreibt die Norm ISO/IEC

19795.

Bei biometrischen Systemen spielt aber auch die Erkennungszeit eine grosse Rolle. Neben der Sicherheit und Zuverlaessigkeit sind die Benutzerakzeptanz und die Gebrauchstauglichkeit (usability) bei der Beurteilung eines biometrischen Systems entscheidende Kriterien.

B Lorem Ipsum

Nunc urna nunc, gravida et leo nec, ornare hendrerit nisl. Maecenas non enim lacinia, ultrices lectus et, egestas diam. Morbi fringilla semper nibh non ornare. Nulla efficitur eleifend odio, sed molestie nisi. Cras fermentum eleifend dignissim. Interdum et malesuada fames ac ante ipsum primis in faucibus. Quisque et ipsum sit amet nibh luctus euismod non quis nunc. Suspendisse ornare non ipsum a fermentum. Curabitur accumsan, risus nec consequat accumsan, erat velit maximus ipsum, vitae auctor justo dolor vel est. Integer laoreet nulla ornare, tempus urna ac, pharetra tortor.

Nunc felis enim, gravida semper est quis, cursus congue nisl. Morbi sagittis leo in lacus fermentum, a tempor mauris commodo. Sed sodales rhoncus nibh, sit amet malesuada neque congue vel. Nam feugiat molestie nibh feugiat feugiat. Vivamus eget interdum enim, sed dapibus dui. Nunc ullamcorper leo ornare euismod rutrum. Quisque pulvinar nunc diam. Ut auctor purus sed sagittis consectetur. Morbi feugiat eros a ante feugiat ultricies. Aenean a euismod lacus. Mauris tempor efficitur lectus, a scelerisque dolor cursus non. Aenean semper tincidunt lobortis. Quisque mollis nisl quis fringilla facilisis. Donec efficitur aliquet ante ut volutpat. Integer posuere ex magna, a volutpat erat cursus at. Nulla facilisi. Etiam in consectetur diam.

Ut cursus mauris ac diam euismod pharetra. Duis at lacinia odio, nec varius libero. Aliquam egestas scelerisque eros vitae rhoncus. Sed sed rutrum augue, non tempor quam. Quisque quis ante et sem tempus interdum. Proin suscipit sagittis rhoncus. Aliquam erat volutpat. Vivamus at diam tincidunt, congue lectus eu, malesuada purus. Integer in libero tellus. Cras suscipit, nulla non fringilla porttitor, velit metus tincidunt libero, eu pretium massa quam vel purus. Maecenas ornare scelerisque ipsum eu suscipit. Sed sodales turpis vehicula odio pretium, sed euismod tellus faucibus. In eleifend, leo eu congue blandit, libero purus ultricies justo, ut tristique nisl mauris a sapien. Ut facilisis ac est quis tempor. Donec vitae laoreet tortor.

Nam ultricies condimentum mi, tempor aliquet libero efficitur quis. Integer tristique risus et pretium consequat. Ut eros augue, mattis nec erat a, rutrum tincidunt dolor. Proin non bibendum sapien. Nunc auctor augue et leo fermentum bibendum. Etiam non neque gravida, faucibus enim a, tristique justo. Ut tincidunt tristique dictum. Sed egestas arcu eu nisi auctor interdum. Vestibulum non pellentesque felis, non pharetra mauris. Suspendisse eget hendrerit nulla, sit amet vestibulum metus. Quisque vel nulla erat. Sed viverra turpis id metus feugiat, sit amet imperdiet nulla ultricies.

Mauris tempor tortor at sapien porttitor, sed volutpat erat consectetur. Aliquam et dignissim quam. Etiam auctor, mi quis sollicitudin convallis, augue elit porta lectus, ac pulvinar ligula sem eget diam. Suspendisse eu fermentum est. Etiam sodales erat vitae urna pharetra cursus sit amet id eros. Quisque ut tincidunt ipsum. In hac habitasse platea dictumst. Fusce libero ligula, gravida vel diam non, hendrerit convallis sapien. Nullam volutpat hendrerit enim non hendrerit. Duis tincidunt arcu ipsum, tempus fermentum nisi porttitor eget. Quisque eleifend ullamcorper condimentum. Pellentesque luctus pellentesque nulla, non congue mi sagittis sed. Vivamus suscipit eleifend rutrum. Aenean placerat dictum magna, nec aliquam massa aliquam vitae.

Sed fringilla est sit amet est vestibulum, vel facilisis elit interdum. Duis nec suscipit erat. Vestibulum vehicula sed dolor non ullamcorper. Morbi quis tristique diam, nec consequat lorem. Nam aliquet tincidunt tempus. Nullam ac lorem nec est scelerisque auctor. Sed facilisis odio tellus, et condimentum ex consequat tempus. Praesent scelerisque dolor eu mollis pharetra. Integer hendrerit quam lorem. Morbi euismod enim quis scelerisque varius. Nunc luctus ex nec quam scelerisque semper. Suspendisse ut rutrum ex. Aenean fermentum mi et vestibulum fringilla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Donec vel nibh risus. Aliquam interdum at lectus ut cursus. Maecenas pellentesque, nisl at scelerisque laoreet, felis quam finibus elit, eget interdum nunc turpis nec magna. Sed pellentesque egestas nunc nec dapibus. Vivamus sed leo laoreet, lobortis augue quis, laoreet odio.

Aliquam blandit fermentum viverra. Integer egestas consectetur tellus quis mattis. Nunc ultrices massa ut ante porta dictum. Fusce ex sem, pharetra ac lorem a, tincidunt cursus ipsum. Suspendisse potenti. Phasellus ac feugiat sem. Aliquam est lacus, auctor quis odio eu, blandit consectetur nibh. Praesent at nibh et felis efficitur sollicitudin. Ut sapien dui, fermentum eget ultrices et, imperdiet a quam. Praesent nec lorem ac quam imperdiet bibendum at nec magna. Suspendisse ullamcorper diam non nulla congue, et laoreet tellus consectetur. Cras posuere suscipit risus eu accumsan. Morbi ac iaculis sapien. Mauris et faucibus neque. Aenean tortor est, dictum ac molestie quis, congue sit amet ex. Morbi tincidunt eu dolor at vestibulum.

Aliquam sagittis commodo dolor, sed dictum odio fringilla sed. Etiam pellentesque laoreet leo, et pulvinar enim euismod eget. Integer at nulla ac velit sagittis dapibus. Praesent quis mi ut risus feugiat maximus vel sit amet sem. Quisque tempor leo eget neque pulvinar facilisis in eu nisl. Vivamus et mi euismod, tincidunt dui sit amet, sodales turpis. Aenean enim ligula, pretium id sollicitudin et, gravida vitae orci. Mauris non nunc pretium, pellentesque metus sed, eleifend eros. Vivamus maximus est ac est condimentum, a vulputate tortor dapibus.

Curabitur iaculis nulla vitae sem molestie dictum. Ut maximus nisi nec elit mattis sollicitudin. Donec convallis ex nibh, nec luctus felis efficitur eleifend. Nunc eu feugiat lacus. Aenean tincidunt nulla sed turpis imperdiet interdum. Fusce semper massa at sem dignissim euismod. Morbi at sodales tortor. Cras sodales volutpat nisl non convallis. Cras nec ultricies tortor. Donec sit amet dictum lorem. Aliquam sed purus bibendum, euismod sem nec, lacinia leo. Sed sed pretium odio. Fusce feugiat nisi sit amet urna pretium elementum. Sed faucibus dolor vitae magna maximus, a aliquet ligula placerat. Maecenas posuere elit vitae consectetur gravida. Cras et diam dictum, ullamcorper justo et, feugiat libero. Cras non felis faucibus, luctus erat sit amet, bibendum nisi. Phasellus a augue ac mauris hendrerit fringilla. Integer interdum risus tincidunt ultrices congue. Nam rutrum sit amet sem eget feugiat.

Suspendisse potenti. In quis pulvinar neque. Integer fringilla neque elit, eget aliquet ante aliquet eu. Nullam fermentum mattis massa vel lobortis. Vestibulum purus sem, porta at porta eget, porttitor sit amet odio. Aenean blandit nunc non nunc varius fermentum. Aliquam condimentum accumsan accumsan. Etiam turpis neque, placerat et lacus in, convallis malesuada elit. Pellentesque fringilla arcu arcu. Praesent et pulvinar purus. Sed at augue ac purus ornare luctus. Ut vulputate ultrices quam, sed mollis mi. Suspendisse porta pulvinar quam, vitae tristique felis pellentesque eget. Cras fermentum dolor pellentesque, convallis tortor ac, commodo massa. Pellentesque imperdiet risus ut

lorem euismod, aliquet blandit tellus ullamcorper.

Duis id mauris quis turpis consectetur finibus non consectetur orci. Cras sed blandit libero, sit amet rutrum mauris. Praesent volutpat tellus quam, id dignissim ante porttitor in. Proin facilisis aliquam gravida. Aliquam laoreet porttitor justo, at venenatis tortor ultrices eu. Cras in magna lobortis, condimentum lectus eget, bibendum lacus. Donec eu sapien vel dui pellentesque posuere. Aliquam sit amet convallis neque, et lobortis est. Pellentesque non sapien id sem suscipit finibus. Pellentesque viverra massa dolor, ac mollis nibh hendrerit in. Suspendisse ultrices viverra massa, id ornare ipsum placerat eget. Sed pharetra mi ac nisi mattis, ut sodales lorem accumsan. Curabitur sed turpis massa. Aliquam maximus id velit vitae cursus.

Nulla sollicitudin libero vel elit viverra, aliquet fermentum arcu faucibus. Nam ullamcorper egestas nisi ac lobortis. Phasellus in turpis in odio finibus pharetra a nec magna. Quisque molestie ut arcu tincidunt vulputate. Morbi tincidunt mi felis, vel tempus nisi ullamcorper non. Aliquam rhoncus est faucibus, iaculis purus et, venenatis odio. Ut lorem felis, cursus et sollicitudin vitae, interdum a arcu. Cras imperdiet feugiat quam, ornare pulvinar augue tempor ut. Nunc efficitur, metus tempus malesuada aliquet, ante neque sollicitudin libero, ac tempus nisl nulla vel quam. In eu porta elit. Mauris feugiat, dui a cursus iaculis, erat augue blandit libero, in posuere justo magna ac ipsum. Duis volutpat viverra pretium. Fusce finibus vitae lacus vitae porta. In eu augue non felis ornare sollicitudin ac quis urna. Nulla ut velit nisl. Vivamus et fermentum est, ut rhoncus ex. Duis porttitor elementum elit, eget condimentum sem consectetur a. Integer eu scelerisque ex.