Combining Periodic and Continuous Authentication using Keystroke Dynamics

Per-Kristian Nilsen

Master in Information Security
Submission date: June 2018
Supervisor: Patrick Bours, IIK

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

Preface

This is a master's thesis written during the spring semester of 2018 to complete a master's degree in information security. It documents a project that was conducted after being suggested as a research topic by Professor Patrick Bours, who also supervised the project.

The thesis dives into rather technical details surrounding keystroke dynamics. Experts in biometrics are the target audience in mind. However, I have attempted to write it in a way that is understandable and easy to follow for others who might also be interested. This means that biometric terms and concepts are explained, but not to an extent that would bore the target audience. Therefore, I hope that the reader will enjoy the work presented here, regardless of their background.

Acknowledgement

I would like to extend my sincere gratitude to Professor Patrick Bours who has been my supervisor for three projects, including this thesis. Your passion for our field of study is highly motivating, and I cannot begin to describe how important your guidance and encouragement has been to me. Even in your most hectic workdays, you have always been available and found time to help me brainstorm and focus on what mattered the most. Thank you.

To my family, I think you know how much you mean to me. You have always encouraged me throughout the five years I have spent in Gjøvik. The time I have spent with you during my trips back home has been invaluable. Thank you. To my parents, Anita and Per-Eirik, thank you for your support and patience especially during the last 8 months.

To Sigve, who tempted me into coming along to Gjøvik, thank you for the years we spent together as roommates and for dragging me along. You are truly awesome. To my other friends back home, there are (un?)fortunately too many of you to mention everyone by name. Still, you know who you are. Thanks for being great. I do however want to mention Pål, who has given me advice and boosts of confidence throughout my education. Thank you.

To the guys in my study group, that being Audun, Emil, Erik, Jardar, Jørgen and Kristian, thanks for making these last couple of years of study so much more enjoyable. Studying for all our exams would have been a huge pain in the neck without your company. Lastly, I would like to thank my friends in Klatregruppa Grad 9. Climbing and hanging out with you guys has been a ton of fun.

P-K.N.

Abstract

If a user leaves their computer without locking it, any nearby person can simply walk over and take control over the machine as if they were the genuine user. If the imposter also has malicious intentions, the genuine user could face serious consequences such as identity theft or blackmailing. Keystroke dynamics enables the system to repeatedly authenticate the user in the background by recognizing their personal typing pattern. Seen from another perspective, the system can lock the imposter out based on detecting unfamiliar typing patterns.

The aim of this project was to combine two such authentication mechanism, namely *continuous* and *periodic* authentication. Continuous authentication (CA) systems react to every single keystroke action performed by the user, though such systems base their decisions on very limited amounts of information derived from a couple of keystrokes at a time. Periodic authentication (PA) systems base their decisions on statistics from samples containing a large number of keystrokes, however, they only perform checks after that large number of keystrokes has been collected. This gives the imposter a certain period of freedom before being locked out. By integrating a PA system into a CA system, we eliminated the disadvantages of both authentication mechanisms while still benefiting from their advantages, in addition to improving the CA system's imposter detection performance by over 18%. Alternatively, the combination of the systems could give the genuine user an increase in the average number of keystrokes before being wrongfully locked out by over 8%.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Topic covered by the project

When a person attempts to access certain resources or systems, they may need to verify that they are in fact authorized to do so, often through an authentication mechanism. Authentication is the act of verifying that the current user matches the identity they are claiming ownership of. After claiming an identity, for example by providing a username, the current user may support their claim by presenting something only the true owner of the identity is supposed to *know* or *have*. This could for example be a password or a token such as a key card. The user may also give a representation of what they *are*, which bring us to the topic of *biometrics*.

Biometric systems measure human characteristics to determine the identity of users. While biological biometrics is now widely embedded in our everyday lives such as fingerprint scanning and lately face recognition in our smart phones, we can also be identified by the way we behave, i.e. by means of *behavioral biometrics*. Voice and signature recognition are examples of behavioral biometrics, however the topic of this project revolves around *keystroke dynamics* which involves measuring a user's typing patterns on a keyboard. Every individual has their own way of typing on computer keyboards, and this can be taken advantage of by authenticating users by measuring for example the pressure or timings of keypresses, of which the latter will be our focus. Examples of such timings can be the time of when keys are pressed or released.

Authentication can be used both for giving access (*static authentication*) and removing access (*dynamic authentication*). When a user claims an identity and writes the correct password, the biometric system can compare the typing pattern in the written password to the way the true owner of the identity writes the same password. If the patterns match, the user will be given access. This is an example of static authentication using *fixed-text* keystroke dynamics.

Keystroke dynamics can also be used for dynamic authentication. Even after a user is logged into a given system using *static authentication*, they can continue being authenticated by a background process after the initial log-in, removing their access if they are believed to be an imposter. Dynamic authentication based on keystroke dynamics can be done by two different methods: *periodic* or *continuous authentication*. With periodic authentication (PA), the system collects keystroke timing information over a period of time, and retroactively analyzes the collected data. It will then analyze the statistical properties derived from the data, decide if they fit the properties of the genuine user and remove access if they do not fit. Systems with continuous authentication (CA) will check if the user is genuine after each keystroke action they perform.

Authors of PA systems generally refer to their systems as *continuous authentication*, however we will in this project refer to them as *periodic authentication*. The reason for this is that the word "continuous" implies checks being performed after every action, while PA systems require a greater number of actions before the check is initiated. The distinction between CA and PA was described by Dowland et al. in 2002 [2], and Bours [3] described a similar but more specific distinction in his CA study in 2012.

A great benefit to CA is that the system can make a decision every time the user presses a key, whereas impostors have time to perform a certain amount of potentially harmful keystroke actions in-between identity verifications in PA systems. On the other hand, CA systems can not take advantage of statistical analysis like PA systems can. Therefore, we would like to investigate if extending an existing CA system with a PA system can remove the inherent disadvantages of both types of systems as well as significantly improve the performance of the CA system.

1.2 Keywords

Behavioral biometrics; keystroke dynamics; continuous authentication; periodic authentication; distance-based classification; machine learning

1.3 Problem description

While many systems and applications rely on static authentication, such as log-in processes, most systems do not perform further authentication to ensure that the current user has not changed since logging in. Physically leaving an unlocked computer unattended is not an uncommon practice in many work environments, which opens up for free and unauthorized access by anyone willing to seize the opportunity. The longer the genuine user is absent, the more time unauthorized users have to access information or cause damage to any part of the system. Restricting the amount of actions intruders can perform is therefore needed in order to reduce the damage potential.

To the best of our knowledge, there has been no research conducted on combining CA and PA for keystroke dynamics. Because of this, the drawbacks of both types of systems are present in literature. As stated earlier, a disadvantage to PA systems is that there is a window of time where an imposter can use the system before the next authentication process is performed. Most of the existing PA systems need several hundreds to over one thousand keystrokes for every periodic authentication. This leaves the imposter with too much freedom before their access is removed.

The main problem with CA systems is that they need to base their decisions on a very small amount of information about the current user's typing pattern. Every action is continuously classified as an imposter action or a genuine action. That means that CA systems are not able to rely on statistics from the current user's typing pattern.

CA and PA systems share another common problem, being that they can make errors. More specifically, they may wrongfully believe that the genuine user is an imposter, or they may mistake an imposter for being the genuine user. In real-time implementations of such systems, the first case would lead to access being removed from the genuine user, which would be a source of irritation or frustration. The second case would give an imposter time to perform more keystrokes before (hopefully) having their access removed at a later point.

Both systems take a certain amount of time to detect imposters. While PA systems generally need a fixed amount of recorded keystrokes before analyzing them, CA systems remove access when they no longer trust the genuineness of the user. Every keystroke increases or decreases the CA system's trust level, and the more similar the current user's typing pattern is to that of the genuine user, the longer they are allowed to remain logged in. Therefore, an important problem to solve is to decrease the number of keystrokes imposters are allowed to perform before having their access removed, while also allowing genuine users to perform as many keystrokes as possible before being wrongfully locked out.

1.4 Justification, motivation and benefits

This project was conducted for its potential to increase the viability of free-text keystroke dynamics in industries and sectors where a higher level of information security is needed or desired, such as the health sector or other critical infrastructures. This does not exclude other work environments or even private computers, as any owner of a system where security is essential could benefit from imposters being automatically detected and locked out by typing on the keyboard. As improving the CA system's detection performance would lead to imposters being detected more quickly while genuine users are locked out less frequently, it would result in a higher level of security and a better user experience.

1.5 Research questions

The main objective of this project was to answer the following research question:

- *Can incorporating periodic authentication methods into a continuous authentication system using keystroke dynamics improve the original system's imposter detection performance?*

In order to answer this research question, a couple of sub-questions were also considered. They are addressed throughout the different chapters of this thesis. The sub-questions were as follows:

1. *What is the impact on computational performance when incorporating a PA system into the CA system?* Continuous and periodic authentication systems are meant to operate transparently in the background. Therefore, we wanted every authentication process to be performed quickly in order to avoid slowing down the user's machine.
2. *How can the decision of the PA system be used by the CA system?* If the PA system believes the current user is an imposter, it can either remove access immediately or cause the CA system to place less trust in the user.

1.6 Contributions

In this project, we have developed a CA system and extended it with a PA system, improving its detection performance. Two architectures for combining the systems are proposed. We also provide performance analyses of both individual systems, as well as both of the proposed architectures. The computational impact of combining the system is also addressed. The end result was a system that can react after every keystroke action while also utilizing statistics derived from larger keystroke samples in order to make more accurate decisions. The developed software will be delivered for continued research on this topic.

2 Related work

This chapter aims to describe the literature relevant to the project. In order to discuss the available literature, a few concepts from the field of biometrics must first be explained. In order to compare the current user's characteristics to those of the genuine user, a *reference* and a *probe* is needed. In the context of continuous or periodic authentication and keystroke dynamics, the reference is a stored template of the genuine user's typing behavior recorded during the enrollment phase. This is the period where the biometric system learns the genuine user's characteristics. The probe is a template of the current user's behavior, based on the keystrokes recorded during the user session. Both of these templates are generated by extracting *features* from the recorded keystrokes.

As mentioned in Section 1.1, we will be focusing on the timing information of key actions. The available *timing feature* from a single keystroke is its *duration*, which is a measurement of how long the key is held down. Consecutive keystrokes are called *n*-graphs, where *n* is the number of keystrokes. Single keystrokes can similarly be referred to as *monographs*. Features can also be extracted from these by measuring the *latency* from the press/release of one key to another. Using digraphs (or 2-graphs) as an example, the available latencies are as follows [4]:

- Press-Press(PP): The time elapsed from pressing down the first key to pressing the second key.
- Release-Release(RR): The time between releasing the first key and releasing the second key.
- Release-Press(RP): The time between releasing the first key and pressing the second key.
- Press-Release(PR): The time from pressing the first key and releasing the second key.

In the following chapters, the term *duration* will refer to the timing of a monograph, while *latency* will be a general reference to timings of consecutive keystrokes, unless further specified by using the above acronyms such as "PR-latency".

We would also like to address the use of the term *authentication*. According to the vocabulary specified in the ISO/IEC 2382-37 standard [5], using this term as a synonym for *biometric verification* is deprecated, and the term *biometric recognition* is preferred. However, the *biometric recognition* term has not been widely adopted in the literature on keystroke dynamics, and *continuous authentication* is still largely used. Therefore, we have chosen to also use the term *authentication* in the context of verifying a claimed identity through biometric comparison. Other than that, we intend to closely follow the standardized vocabulary.

With these concepts now explained, the related literature can be presented. The CA system design we have based our system on is first summarized in Section 2.1 in order to allow for further discussions on what PA techniques may result in a good fit for our project. Relevant PA systems are then discussed in Section 2.2 with focus on classification methods, before a quick overview of other important aspects of the literature is given in Section 2.3.

2.1 Continuous authentication

The design we have based our CA system on was a part of the doctoral thesis of Mondal [4], where a *trust model* was used to lock imposters out. Similarly to Bours' CA study [3], Mondal's trust model worked by comparing monographs and digraphs to the genuine user's reference and having the result impact the current *trust level* by means of a penalty-and-reward system.

After the initial static authentication, the user's trust level was set to 100, being the highest achievable level. Probe typing patterns deviating from the reference would cause penalties in the form of lowering the trust level, while probe patterns complying with the reference would cause rewards to be given in the form of increasing the trust of the user's genuineness. A *lockout threshold* was set to a value below 100, and should the current user's trust level fall below said threshold, they would be locked out. Ideal results would have had genuine users' trust levels never dropping below the threshold, while all imposters' levels would drop below the threshold after a small amount of actions performed.

An important part of the trust model was to determine how big a reward or penalty should be given per action performed. For CA based on keystroke dynamics alone, Mondal [4] presented and used an implementation of a trust model referred to as *Dynamic Trust Model (DTM)*. The size of the reward or penalty was determined by a single continuous function based on a *comparison score* computed by comparing the probe to the reference. The larger the difference between the comparison score and comparison threshold (not to be confused with the lockout threshold), the larger the penalty or reward became. For example, an action with a comparison score just below the comparison threshold would only result in a small decrease in trust level.

For keystroke action classification, Mondal followed a machine learning approach and two statistical approaches. The first statistical approach (SA-1) calculated the classification score to be used in the DTM by using Scaled Euclidean Distance (SED) for monographs and a combination of SED and Correlation Distance for digraphs. The second statistical approach (SA-2) used the same distance metrics, but converted the distances into the classification score using fuzzy logic. It is also worth mentioning that Bours used Scaled Manhattan Distance in his CA research [3]. In Mondal's [4] machine learning approach, an *Artificial Neural Network*, a *Counter-Propagation Artificial Neural Network* and a *Support Vector Machine* were combined in a *Multi-Classifier Fusion* architecture.

The overall best machine learning results were achieved by training the classifier with data from the genuine user and from a set of imposter users, which in the original study [4] was called Verification Process 3 (VP-3). Testing was done with data from the genuine user which was not used for training and with data from the remaining imposters not involved in training. This scenario is applicable in many cases, including the use on personal computers, as it shows the performance when imposters are not other users of the same system. The performance was measured in terms of *Average Number of Imposter Actions (ANIA)* and *Average Number of Genuine Actions (ANGA)*, as well as number of imposters going undetected. The ANIA rate represents the number of keystroke actions needed on average before imposters are detected, while the ANGA rate tells how many keystroke actions genuine users can perform on average before they are mistaken for being an imposter.

VP-3 achieved an ANGA rating of 16057 and an ANIA rating of 499, with 1.3% of imposters not being detected. When compared to the best statistical approach (SA-1) having an ANGA rating of 14096 and ANIA rating of 686 with 0.9% of imposters not detected, one could argue that the VP-3 machine learning approach performed better due to imposters being rejected faster on average, and the genuine user being rejected less often. However, SA-1 catches a larger percentage of imposters than what VP-3 does, which certainly is an important result.

Mondal's dataset consisted of mouse and keystroke data collected from 53 participants who were either students or university staff. The data was collected in a completely unconstrained manner by having the participants install a tool for logging keystrokes and mouse events on their own computers. They were not given any specific task, ensuring that the collected data represented the participants' natural behavior. Mondal reported that the dataset had an average of 47600 keystroke events per participant. At the start of our project, the dataset had grown in size to 57 users.

This dataset was also used for testing our CA and PA combination, although the recorded mouse activity was not utilized as mouse dynamics was beyond the scope of the project. In his approach, Mondal used 35% of a user's recorded keystrokes for training, up to a maximum of 20000. This was a sufficient amount of data seen in relation to the sizes of references used in state-of-the-art PA studies. Furthermore, 10% was used for adjusting the parameters of the algorithms, and the rest was used for testing.

2.2 Periodic authentication

There is a significant amount of available literature on PA systems. Discussing it all is beyond the scope of this report. The focus will therefore mostly be on research achieving viable results using free-text authentication and having potential for being incorporated into the CA system. This section will present the various options available to us from literature regarding methods used for PA. When developing the PA system, we used a custom combination of properties from existing research, as opposed to using an entire PA system as it is described by its original authors.

Periodic *identification* systems are also included in this section. These systems attempt to recognize who the user is without them claiming an identity first. While generally having more computationally expensive matching algorithms than authentication systems, they may still have other relevant properties such as feature comparison methods which can also be used for authentication.

The authors of the literature discussed in this section usually refer to their own solutions as CA, however we will refer to them as PA if they are not truly continuous, due to the reason stated in Section 1.1. An extensive and detailed literature study of PA systems [1] was delivered in IMT4215 Specialization Project in December 2017. This section further builds upon the knowledge collected in said study.

2.2.1 Statistical approaches

To the best of our knowledge, incorporating a PA system into a CA system has not been done before. We could therefore not know the answers to our research question and subquestions before performing our own analysis of the CA/PA combination. We could however look at what promising results had been achieved earlier, and how they were achieved. This gave us indicators for how we could assemble a reasonable combination of CA and PA.

One of the most cited articles on PA systems was written by Gunetti and Picardi [6] and published in 2005. They introduced the R- and A-distances, which were relative and absolute distances used for comparison, and they used 2-, 3- and 4-graph latencies in their distance calculations. Their solution is interesting due to how it accounts for variation in genuine users' typing behavior. If a genuine user for some reason types slower than usual, for instance due to cold fingers, their typing pattern is likely to stay relatively similar to the regular pattern, only at a slower speed. The relative distance accounts for this when comparing a probe to a reference, and is used in combination with the absolute distances of speed between the samples. Both the R- and A-distances are explained in detail in Section 3.3.

They achieved a False Match Rate (FMR) of 0.005% and False Non-Match Rate (FNMR) of 4.833%, meaning imposters were undetected in 0.005% of authentication processes, while genuine users were wrongfully believed to be imposters in 4.833% of all cases. This was using a *block size* of 700-900 keystrokes, meaning 700-900 recorded keystrokes were used to form each probe. Block sizes this large give imposters a fairly large window of unauthorized access, and for the CA/PA combination, we wanted a block size similar to our CA system's ANIA rate, or smaller. This is more easily expressed by converting the FMR and FNMR rates into respective ANIA and ANGA rates by means of the formulas presented in [7], where Bours and Mondal first introduced the ANIA and ANGA rates. A middleground block size of 800 keystrokes will be used for simplicity's sake:

$$ANIA = \frac{block\ size}{(1 - FMR)} = \frac{800}{(1 - 0.00005)} \approx 800 \tag{2.1}$$

$$ANGA = \frac{block\ size}{FNMR} = \frac{800}{0.04833} \approx 16553 \tag{2.2}$$

Genuine users are rarely rejected with this ANGA rate, which is also the case in Mondal's [4] CA system. The ANIA rate is higher than that of the Mondal's CA system which was 499, however it is difficult to compare systems using completely different datasets. For our PA system, we wanted a lower ANIA rate than 800 so that it would in more cases have a chance to make a decision before the CA system had already removed access from the current user. This way we could make more use of both authentication systems.

Apart from detection performance, we must also take computational performance into account to discuss *research sub-question 1*. Gunetti and Picardi's [6] system used 140 seconds per authentication, which was a clear issue. Granted, this was on a Pentium 4 processor, and more modern CPUs should provide significantly better performance. The reason for the suffering computational performance was a sub-optimal classification algorithm which compared a probe sample to the references of every single user in the system, which in their experiment was 40 users. This is useful for periodic *identification*, where the system attempts to recognize who the user is without them claiming an identity first. We avoided using such an algorithm in our project in order to keep processing costs at a minimum. Simply modifying the algorithm to not consider all users per verification process could have been an option for increasing the speed, however we could not predict the impact that would have had on the detection performance.

Several other researchers have also used Gunetti and Picardi's R- and/or A-distances [8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. Of these, especially Ferreira and Santos [10] stand out as they attempted to tackle both the block size and computational performance problems of Gunetti and Picardi's [6] study. They used a block size of 250 keystrokes, achieving an Equal Error Rate (EER) of 1.4%, meaning the FMR and FNMR are equal at that percentage. They also mentioned that a specific setting gave a result of 0.5% FMR and 2.7% FNMR, which corresponds to an ANIA of 251 and an ANGA of 9259. With such a low ANIA, we saw this system as a good candidate to use as inspiration for our own PA system to be incorporated into our CA system. The mentioned ANGA was also in an acceptable range which generally would not lead to very many false rejections per day.

As opposed to Gunetti and Picardi's solution, Ferreria and Santos' system [10] only compared probes to the reference of the claimed identity, ensuring fast computation. The size of the reference used was 11250 keystrokes, which is comparable to Mondal's [4] training sets, as mentioned in section 2.1. Their method involved extracting monograph durations and digraph RP-times. Furthermore they also used PP-latencies of 2-, 3-, and 4-graphs, similarly to Gunetti and Picardi [6].

An interesting aspect of their system was to identify the 10% most consistent n-graphs with regards to extracted features. In other words, these were the n-graphs which the user would type in a similar manner most of the time. Then, during an authentication process, the system would place more strict expectations onto these n-graphs when they were typed by the current user. All in all, this PA system consisted of several properties and mechanisms which could potentially be useful for our project, as it provided solutions to the block size and computational performance issues in Gunetti and Picardi's [6] system. They also performed their experiments on data collected in an unconstrained manner, which matches the setting used in Mondal's [4] dataset. This increased the possibility of achieving a high performance in our implementation. However, our focus was not on achieving optimal or state-of-the-art performance for our CA or PA systems, but rather on improving the CA system's performance. Therefore, not all properties of Ferreira and Santos' [10] system were seen as necessary for our own PA system.

Other statistical methods than the R- and A-distances have also been used in literature. Regarding PA systems, examples include Euclidean distance [18, 19, 20, 21], Kolmogorov-Smirnov Test [17, 22], multivariate testing [17], Chi-square test [23], Manhattan distance [18, 20, 24] and Scaled Manhattan distance [20]. Some studies have applied several classifiers [4, 11, 17, 18, 20, 24]. For instance, Kaneko et al. [18] applied Euclidean distance, Manhattan distance, a proposed custom distance and Gaussian probability density function. Reported results showed that Euclidean distance performed best, however the experiment setting was writing a fixed Japanese text of around 200 keystrokes. It is hard to know whether the results would be similar for the dataset used in our project, due to the large differences in data collection methods.

2.2.2 Machine learning approaches

Machine learning has also been utilized in recent years, with some of the research presenting promising results. An interesting example of this is Ahmed and Traore's [25] work from 2007, where neural networks combined with a key mapping technique were used in order to predict digraphs missing from a user's reference. This meant that a much smaller amount of different digraphs needed to be recorded in the enrollment phase. If the current user typed a digraph which was never recorded for the reference, it could still be compared to the *approximated* values of the missing digraph, based on the genuine user's actual recorded digraphs. They achieved an FMR of 0.0152% and FNMR of 4.82% with a block size of 500 keystrokes and by considering monograph durations and digraph RP-latency. This can be converted to an ANIA of 500 and ANGA of 10373.

In addition to neural networks being used in literature [4, 20, 25], other machine learning methods have also been used both in CA and PA systems, such as k-means clustering [26, 27], kernel ridge regression [28], decision trees [29], random forest [30], support vector machine [4] and k-nearest neighbor [11, 17, 21, 31, 32].

2.3 Overview

The previous sections described the methods used in literature as well as highlighting some particularly interesting studies. Since we were not restricted to using the entire systems as they were described by their authors, it was beneficial to look at a general overview of the literature, and to compare certain properties of the studies.

2.3.1 Data collection

The approaches used for experimental data collection is interesting for the project, as some are more similar to Mondal's [4] unconstrained collection approach than others. There are several other studies with unconstrained data collection [3, 10, 12, 15, 23, 25, 33, 34, 35], whereas some studies constrained the participants to typing freely into a textbox such as in a webform [6, 8, 9, 16, 17, 27, 36]. Other studies had the participants perform specific tasks [19, 22, 28, 31], such as writing a long fictional text. Participants wrote static text in [11, 13], and manually copied various texts in [16, 26, 29]. While the focus of our project was on keystrokes from physical keyboards, it is worth mentioning that keystroke dynamics using soft or touch keyboards also has been researched, such as is Kang and Cho's publication from 2015 [17].

With regards to participants included in experiments, 15 researches had 30 or more participants [6, 10, 12, 14, 16, 17, 19, 22, 24, 25, 26, 28, 31, 32, 34]. One of these had 2000 participants [28]. Since the dataset we used [4] contains data from 57 users, the variance in inter-user behavior should be more than high enough to be comparable to other studies.

2.3.2 Feature extraction

Looking at how feature extraction is performed was also of value, in order to see viable approaches we could use. The related studies extracted various latencies from n-graphs, however some also considered monograph durations [3, 10, 13, 15, 19, 21, 23, 24, 25, 26, 31, 32, 33, 37], which

Mondal's [4] CA system also does. When considering consecutive keystrokes, some studies [3, 8, 9, 16, 19, 20, 21, 22, 23, 25, 26, 27, 30, 31, 32, 33, 35, 36, 37] restricted themselves to considering digraphs only. This was also done by Mondal [4]. One study [28] used only trigraphs, while the rest used several types of n-graphs. Locklear et al. [24] used cognition-centric features in addition to keystroke timings, and six studies [21, 29, 31, 32, 35, 37] included other statistical features such as the rate of certain key presses, words per minute and/or rate of typing errors. Of these, one of the studies [32] presented an extension of an existing system [21] where the new system also utilized stylometry.

Block size in periodic systems is also relevant to look at, as a small block size was wanted for the CA/PA combination. Ten studies used a block size of 500 keystrokes or less [10, 11, 14, 15, 20, 22, 23, 25, 30, 33, 36]. Studies achieving good performance using such a small amount of keystrokes were important to consider when we were to implement the PA part of the project. However, more factors must also be taken into consideration when evaluating the performance of CA and PA studies. For example, one of the studies [23] tested the performance of their system using test data that was already included in the users' references, which artificially skews the results in a positive direction. Another example is one of the studies with a small block size using static text instead of free-text [11]. Therefore, this chapter is concluded with Table 1, where an overview of the properties of the related studies can be found.

Paper	Block size	Users	Task	Method	Performance	Features	DB
[6]	700-900	40	Webform	R- and A-distances	FMR 0.005% FNMR 4.833%	2-, 3- and 4-graph latency	Own*
[14]	50 - 150	50	E-mail	R-distances	FMR 2.02% FNMR 1.84%	n-graph latency	Own
[10]	250	60	Unconstrained	R- and A-distances	EER 1.4%	Duration, Digraph RP, PP for {2-4}-graphs	Own
[15]	150	10	Unconstrained	R- and A-distances	FMR ∼2% FNMR ∼2%	Duration, Digraph RP, PP for {2-4}-graphs	Own
[8]	700-900	21	Webform	Modified R-distance	FMR 0.08% FNMR 18.8%	Digraph latency	[6]
[9]	700-900	21	Webform	Weighted R-distance	FMR 0.07% FNMR 15.2%	Digraph latency	[6]
[25]	500	53	Unconstrained	Neural Network (NN)	FMR 0.0152% FNMR 4.82% EER 2.13%	Duration, digraph latency	Own
[11]	36	19	Webform, static text	R- and A-distances, k-Nearest Neighbor	FMR 0.045% FNMR 0.005%	n-graph latency	Own
[28]	900 words	2000	Pre-defined tasks	Kernel Ridge Regression, truncated RBF kernel	EER 1.39%	Trigraph latency	[38]
[26]	1000	150	Copytask	K-means clustering	EER 0.44%	Duration, digraph latencies	Own
[27]	700-900	14	Webform	K-means clustering	Accuracy 100%	Digraph latency	[6]
[33]	Minimum 2	22	Unconstrained	Bhattacharyya distance	Accuracy 70-100%	Duration, digraph latency	Own
[19]	Unknown	31	Pre-defined tasks	Euclidean distance, weighted probability	Accuracy 23%	Duration, digraph latency	Own
[12]	1 min sliding window	56	Unconstrained	R- and A-distances	FMR 1% FNMR 11.5%	n-graph latency	Own*
[3]	Continuous	25	Unconstrained	Scaled Manhattan distance	ANIA 182	Duration, digraph latency	Own
[4]	Continuous	53	Unconstrained	Scaled Euclidean Distance, Correlation distance, NN, Support Vector Machine	ANIA 499 ANGA 16057	Duration, digraph latencies	Own
[31]	775 on average	119	Pre-defined tasks	k-Nearest Neighbor	EER 3.7%	Duration, digraph latencies, statistical features	Own
[22]	300	35	Pre-defined tasks	Kolmogorov-smirnov test	EER 0.09%	Digraph latency	Own
[23]	150	26	Unconstrained	Chi-square test	FNMR 5%	Duration, digraph latency	Own
[13]	600	10	Webform, static text	R- and A-distances	FMR 4.09% FNMR 5.17%	Duration; 2-, 3-, 4- and 5-graph latency	Own
[17]	100-1000	35	Textbox	12 different classifiers	EER 5.64-14.53%	Digraph latency	Own
[34]	Continuous	35	Unconstrained	Unknown distance	ANIA 6390 ANGA 68755	Digraph, trigraph and word latency	Own
[36]	110	15	Textbox	Markov chain	EER 12.7%	Digraph latency	Own
[20]	110	15	Textbox	NN and 5 distances	EER 22.9%	Digraph latency	[36]
[24]	Time based blocks of diff. lengths.	486	Pre-defined tasks	Manhattan distance, Fisher score	EER 4.55-13.37%	Duration, digraph latency, cognition- centric features	Own
[35]	Minimum 500	10	E-mail	Custom one-class classifier	FMR 4.13% FNMR 12.39%	Statistical features incl. digraph latencies	Own
[16]	∼850-1800	50	Textbox; copytask and free text	R and Similarity measures	EER 10-15%	Digraph latency	Own
[32]	∼3000-6000	30	Electronic university exam	k-Nearest Neighbor	EER 0.55-1.4%	Digraph latencies, stylometry.	Own
[30]	250	21	Webform	Clustering, random forest	FMR 3.47% FNMR 0%	Digraph latency	[6]
[29]	1000	30	Textbox; copytask	Decision trees	FMR 1.1% FNMR 28%	Statistical features	Own
[37]	Sliding window of length 500	55	Pre-defined tasks	Partially Observable Hidden Markov Model	ANIA 55.18	Duration, digraph latencies, statistical features	[39]

Table 1: Summary of relevant periodic and continuous systems. Datasets marked with an ampersand (*) in the database (DB) column are available publicly or by request. This table is an extension of an earlier version in the author's IMT4215 Specialization Project report [1].

3 System design

This chapter describes the individual systems we have developed, as well as the architecture of the combined system. The dataset we have used for testing system performances is described in Section 3.1, while Sections 3.2 to 3.4 present the system architectures.

3.1 Dataset

Regarding separating every user's data for training, validation and testing, we followed the same approach as Mondal [4]. This means that up to 35% of the keystrokes were used for training, 10% were used for calculating certain user specific parameters, and the rest was used for testing. 20000 keystrokes were used as a cutoff for the training set. Therefore, if a user's dataset contained a large amount of keystrokes, the leftover keystrokes initially meant for training were used for testing instead. We have tested the performance of the individual systems with and without the cutoff, and the results can be seen in Section 4.1.3.

Table 2 shows an example of the dataset structure. Each row contains the keycode and duration of a pressed key, as well as the keycode of the next key and the RP-latency to that key. In the example, the RP latency of the digraph "da" is a negative value, which is not uncommon. This means that the user was still holding down the "d"-key for 45 milliseconds after pressing the "a"-key. We can also see that the next key pressed after writing the word "data" is the space key, however we cannot see its duration as the example does not include the next row.

Key	Duration	Next key	RP-latency
d	176	a	-45
a	120	t	16
t	221	a	80
a	137	|space|	102

Table 2: Fictional example of dataset structure where a user wrote the word "Data".

Using the values in the dataset, we can achieve any and all of the four latencies described in Chapter 2 by means of summing durations and/or latencies. Following are the latencies for the digraph "Da".

- **PP**: 176 ms duration + (-45) ms RP-latency = *131 ms*
- **PR**: 176 ms duration + (-45) ms RP-latency + 120 ms duration = *251 ms*
- **RP**: = *-45 ms*
- **RR**: -45 ms RP-latency + 120 ms duration = *75 ms*

These latencies, along with the monograph durations, could then be stored in the reference or used for validation/testing, depending on the example's location in the user dataset.

An issue we had to consider was to decide which combinations of keys were to be considered as actual digraphs. The user may have periodically stopped typing, for example when reading, watching a video or temporarily leaving the computer. In such cases, the last key they pressed before stopping would be unlikely to have a meaningful relation to the first key they pressed when they resumed. Even if there were a meaningful relation, pausing in the middle of a word would probably be uncharacteristic behavior. Therefore, we chose to only regard consecutive keystrokes as digraphs if their RP-latencies were less than 1500 ms.

Another consideration was that the amount of unique digraphs greatly outnumbers the amount of unique monographs. Naturally, this means that user datasets tend to contain a large number of different digraphs, though each digraphs generally have a small amount of *occurrences*. With a small amount of occurrences, we can assume that the timing values of digraphs are more prone to represent a behavior that is not truly representative of the user. With larger user datasets, more occurrences can in general be registered per digraph. Therefore, a decision was made to exclude datasets consisting of less than 10000 keystrokes, in order to ensure more accurate measurements of the system's true performance. 11 users were excluded, leaving us with 46 users whose keystroke data was used for analysis. The average number of recorded keystrokes from the remaining users was 43338.

Lastly, some of the users' datasets occasionally showed very high monograph durations, which were consistently lasting several minutes. While activities such as gaming can cause high durations, the consistent nature of the values seemed unnatural, and may have been caused by a recording error during enrollment. These values were removed from the datasets.

3.2 CA system

The developed CA system is based on the *system architecture* proposed by Mondal [4], though our implementations are not identical. For instance, the classifiers are different. We also use *dissimilarity scores* instead of *similarity scores*. Still, the general architecture remains similar, and is depicted in Figure 1. Features are extracted from the raw keystroke data in the dataset. The training portion of the dataset is used to build the user's reference. The users' testing data is compared againsttheir own and other users' references by the Keystroke Comparison Module. From these comparisons, comparison scores are produced, and are used as input for the Dynamic Trust Model (DTM), as described in Section 2.1. Another optional input for the DTM is a set of personal parameters, which can be calculated using the genuine user's validation data. Based on the new trust level produced by the DTM, the Decision Module decides whether the current user should be allowed to continue or be locked out. The following subsections describe the CA system's architecture, as well as our implementation.

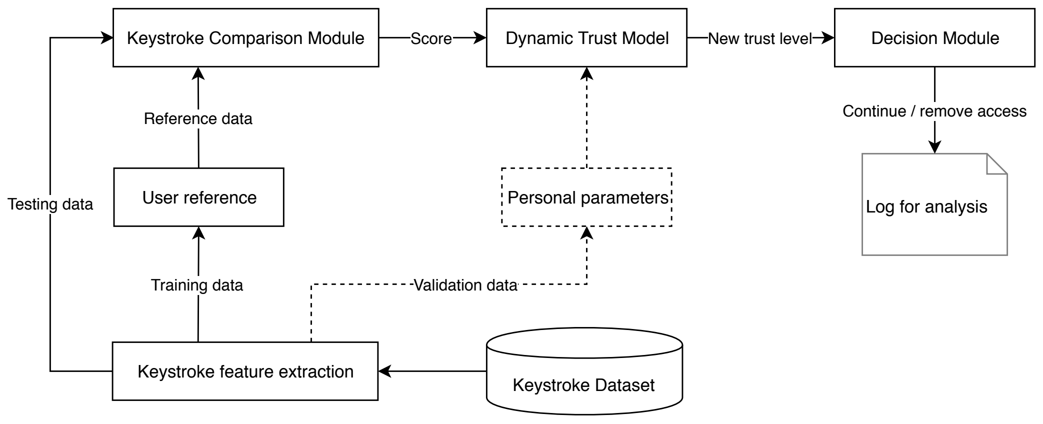


Figure 1: Generalized diagram of the CA system's structure.

3.2.1 Feature extraction and references

Six features are utilized in the CA system, namely keycodes, monograph durations as well as PP, PR, RP and RR latencies from digraphs. While certain systems in literature exclude monographs in their analysis, we found it necessary to consider them in our system. If monographs were ignored, an attacker could wait for 1500 ms between keystrokes in order to avoid typing digraphs. This would result in the system having little to no features available for comparison, even if the user typed a full block of monographs. Including monographs features helps mitigate this security issue.

Monograph	Digraph
- Keycode	- Keycode
- Hit count	- Hit count
- Duration$_{\text{Mean},\sigma}$	- PP$_{\text{Mean},\sigma}$
	- PR$_{\text{Mean},\sigma}$
	- RP$_{\text{Mean},\sigma}$
	- RR$_{\text{Mean},\sigma}$

Table 3: The structure of the references used in our system, where σ is the standard deviation of recorded durations/latencies.

Table 3 shows how these features are used in the reference. For every mono- or digraph occurring during enrollment, the system only stores the *mean* and *standard deviation* of their recorded timing values, as well as their keycode and hit count. The hit count keeps track of how many times each mono- or digraph occurred in enrollment. All features are extracted as described in Section 3.1.

As human behavior is prone to inconsistencies, removing outlier values can give more accurate representations of how a user typically behaves. In PA systems, it is possible to remove outlier timing values from probe blocks, as each n-graph can have multiple recorded occurrences. This is generally not the case for CA systems, as the trust level must be adjusted after every recorded keystroke. However, it is possible to remove outliers while constructing the CA reference. We chose to do so since our classifier relies on standard deviation, which causes it to be negatively affected by outliers. For every feature in the reference, values more than 1.5 interquartile ranges (IQR) below the lower quartile or above the upper quartile were considered as outliers and removed. This was done within the scope of one feature. In other words, if a key were pressed only once and its duration were an outlier compared to other keys in the reference, it would still not be removed. However, if the key had several occurrences, and therefore several durations, outlier durations would be removed if present. The system was also tested without removing outliers, and the results are discussed in Section 4.1.2.

3.2.2 Comparison

When processing a keystroke during testing, its extracted features are handled by the Keystroke Comparison Module which compares them to the genuine user's reference using a distance measure. The comparison is performed by computing the dissimilarity score sc between the probe timing features p and the mean duration μ of the corresponding features from the same keycode in the reference. This is calculated using the following formula:

$$sc = \frac{1}{n}\sum_{i=1}^{n}\frac{|p_i - \mu_i|}{\sigma_i}$$

where i is a specific feature, n is the amount of available *timing features* and σ is the *standard deviation* of the reference features. Our implementation makes a distinction between monographs and digraphs in the sense that they produce two separate scores. In practice, this causes n to either be 1 (monograph duration) or 4 (digraph latencies). For digraphs, this means that the score is the mean of the distances computed for the PP, PR, RP and RR latencies.

The formula is a variant of the Scaled Manhattan Distance, as described by Killourhy and Maxion [40]. They compared the performance of 14 different classifiers on static text keystroke dynamics, where Scaled Manhattan Distance achieved the best Equal Error Rate. This distance metric shows the dissimilarity between the probe and reference, and is used directly as the score input to the trust model.

3.2.3 Trust model and decision module

The trust model used in our system is a variant of the one presented by Mondal [4]. We have made certain changes to make it fit our system, and eliminated one parameter which was not used in our analysis. Algorithm 1 shows our implementation of the trust model. Perhaps the most notable change is the fact that the sigmoid function is inverted horizontally, as seen in Figure 2. This was a necessary feature due to having a dissimilarity score as input.

As mentioned in Section 3.2.2, monograph scores and digraph scores are separated. In practice, this means that the 2$^{\text{nd}}$ monograph of a valid digraph produces *two* scores. One score represents the monograph, while the other represents the digraph. Therefore, a digraph produces *three* scores in total: 2 monograph scores + 1 digraph score. This brought in the question of how to handle digraph actions at the lockout threshold. For example, let the current trust level $Trust_{i-1} = 90.3$ and lockout

Algorithm 1 Algorithm for the Trust Model

```
Data:
$sc_i \leftarrow$ Dissimilarity score for $i^{th}$ action
$A \leftarrow$ Threshold for reward or penalty
$B \leftarrow$ Width of sigmoid function
$C \leftarrow$ Maximum reward and penalty
$Trust_{i-1} \leftarrow$ Trust level after $(i-1)^{th}$ action
Result:
$Trust_i \rightarrow$ Trust level after $i^{th}$ action
begin
    $\Delta_T(sc_i) = \min\{-C + (\frac{C \times 2}{1 + \exp(\frac{sc_i - A}{B})}), C\}$    (3.1)
    $Trust_i = \min\{\max\{Trust_{i-1} + \Delta_T(sc_i), 0\}, 100\}$    (3.2)
end
```

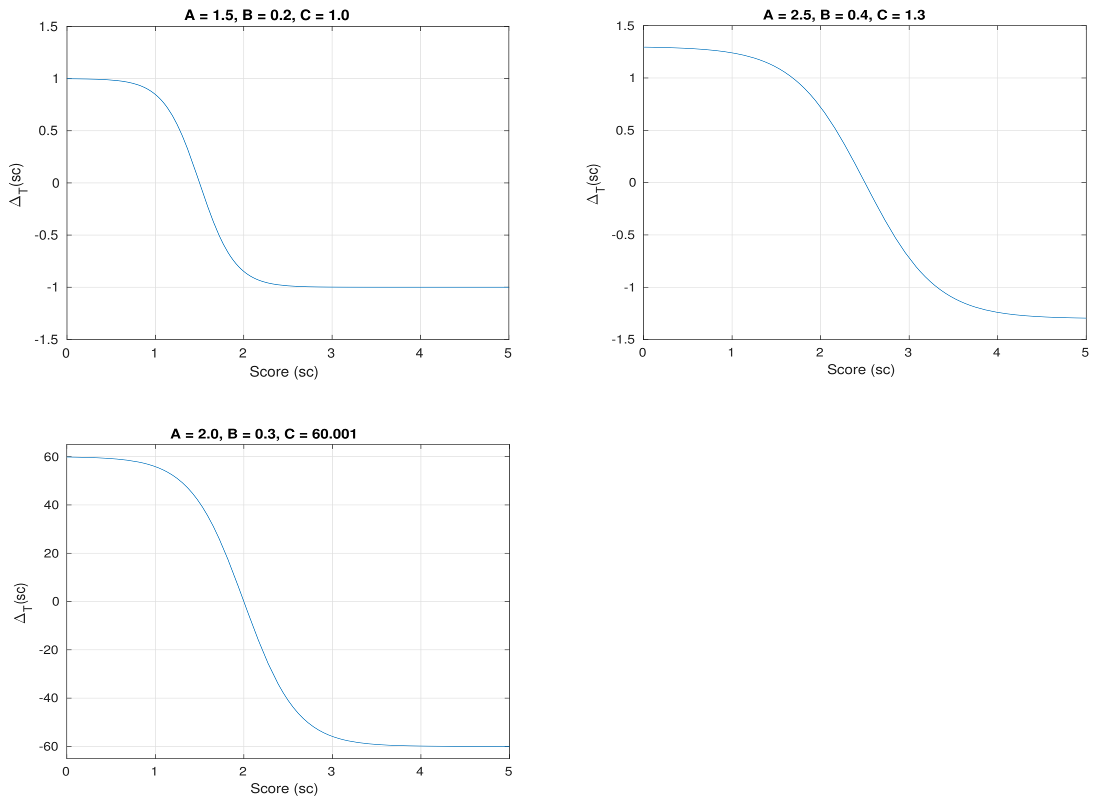


Figure 2: Examples of how the score would affect the trust level using different parameters for Equation (3.1).

threshold $T_{\text{lockout}} = 90$. It is then possible that the next monograph score causes $Trust_i = 89.6$. However, if the system locks the user out at that point, it disregards the fact that the monograph may be the 2nd component of a digraph. This issue is highlighted when the digraph score would have raised the trust level back to a value above T_{lockout}, for example $Trust_i = 90.2$. Therefore, when observing digraph actions, the Decision Module locks the user out only if $Trust_i < T_{\text{lockout}}$ after considering both the monograph *and* digraph scores of the keystroke.

As seen in Figure 1, the DTM can accept personal (user specific) parameters for the sigmoid function. In our analysis, we have tested the system both using global parameters as well as personalizing the threshold for reward or penalty, which is the "A" parameter in Algorithm 1. To adjust this threshold, the genuine user's validation set is used to find their average dissimilarity score against their own reference, before adding a *tolerance level* to that average score. Outlier values are removed from the list of scores calculated from all the keystrokes in the validation set. This ensures that the average of the remaining values is more accurate compared to the user's regular behavior. This lets the system account for the fact that some users type more consistently than others. If a genuine user types less consistently, thus achieving higher dissimilarity scores from their validation set, their threshold will be higher in order to avoid being locked out too often. Consequently, imposters have a slightly higher chance of going undetected in such cases. We see this as a fair tradeoff, as *acceptability* is very important for a biometric system to be viable, and being locked out on a regular basis is an annoyance to the genuine user.

The tolerance level is a system level parameter. With higher tolerance levels, users are allowed to deviate more from the genuine user's expected behavior without being locked out. Lowering the tolerance level would cause the system to be more strict.

3.3 PA system

The foundation of our PA system is based on that of Ferreira and Santos [10]. Some of their features, such as "progressive learning" has been excluded from our implementation. When three consecutive probe blocks were accepted, their system would update the user's reference, infusing said probes into it in order to keep the reference up to date with the genuine user's most recent behavior. The reason for not including this feature is threefold:

- The dataset we used for analysis was collected over the course of about one week per user. The user's typical behavior is not expected to be significantly changed that quickly, eliminating the need for an adaptive reference.
- Even if this would slightly improve performance, our goal is not to necessarily create high performing CA or PA systems. Instead, it is to observe the effect of combining them regardless of what their individual performances are.
- Updating the user's reference several times when testing system performance would severely impact processing time for full test runs. This would in turn impact the amount of different parameter sets we would be able to test in the project's analysis phase. Therefore, we chose to prioritize testing as many sets of parameters as possible over including this feature. In a

real-time system, the extra computational work would be trivial due to larger time periods between each update.

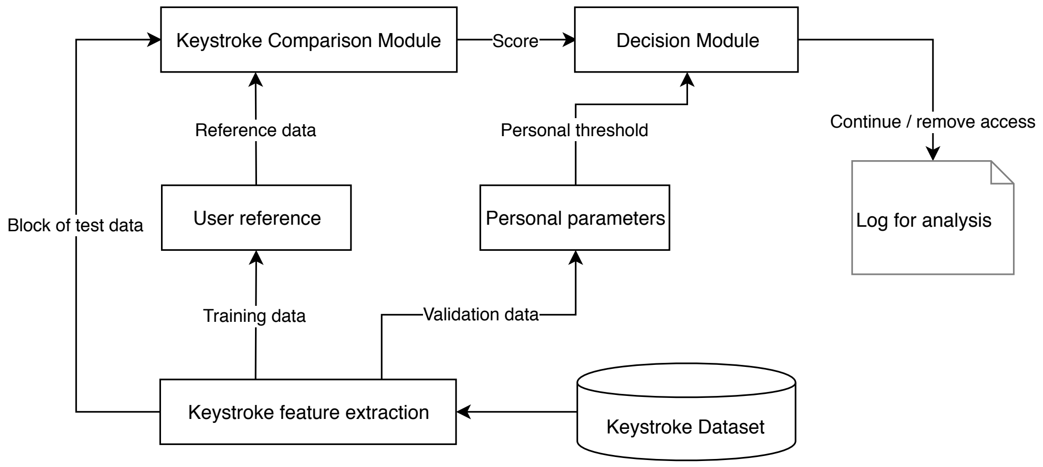


Figure 3: Generalized diagram of the PA system's structure.

The PA system is displayed in Figure 3. Its structure is similar to that of the CA system, though the PA system has no Trust Module. Also, a block of keystroke features is fed as input to the Keystroke Comparison Module, as opposed to one keystroke at a time in the CA system. The following subsections further describe the different components of the PA system.

3.3.1 Feature extraction and references

We have used the same reference for our PA system as described in Section 3.2.1, however, a different set of features is considered. Ferreira and Santos [10] consider monograph durations, digraph PP and RP latencies, as well as PP latencies for trigraphs and tetragraphs (4-graphs). In order to soften computational impact, trigraphs and tetragraphs were excluded from our system. This left us with the features listed in Table 4. Probes are formed using the same structure, and since the PA system has significantly more data to base its decision on than the CA system, outlier values can be removed from the probe as well as the reference. Outlier removal is performed using IQR, as in the CA system.

Monograph	Digraph
- Keycode - Hit count - Duration$_{\text{Mean}}$	- Keycode - Hit count - PP$_{\text{Mean}}$ - RP$_{\text{Mean}}$

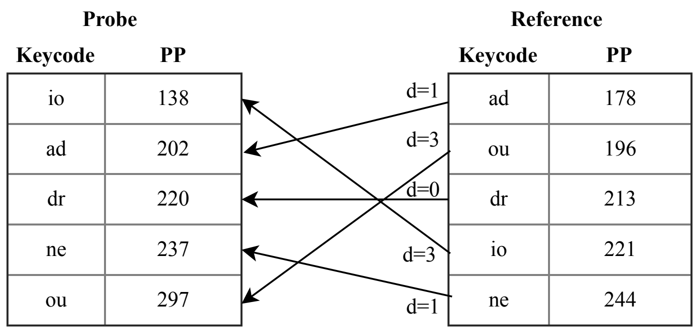
Table 4: The structure of the probes and references used in our PA system.

3.3.2 Comparison

The Comparison Module uses the R- and A-distances presented by Gunetti and Picardi [6], which have been adopted by several other authors, including Ferreira and Santos [10]. These two distances are calculated for every sample, and are *summed* to produce a final distance. Following is a description of how these distances are produced in our system.

R-distance

After the current user has typed an amount of keystrokes equal to the block size used by the system, the features in Table 4 are extracted from the block sample to form a probe. The Keystroke Comparison Module then finds all n-graphs that are shared between the probe and the genuine user's reference. The shared n-graphs are separated into one set for monographs and one set for digraphs which are processed individually. To calculate the R (relative) distance of a probe, the shared monographs and digraphs are extracted into probe and reference feature vectors sorted by durations and latencies, respectively. For every feature vector (monograph durations, digraph PP and digraph RP), the position of each n-graph in the probe is compared to the same n-graph's position in the reference vector. This results in a *position distance* being produced for each n-graph. The R-distance is then calculated by summing the position distances of all n-graphs per feature vector, and normalizing the result. The normalization allows us to compare and combine R-distances calculated from feature vectors of different lengths. This is useful as the number of shared digraphs may be different from the number of shared monographs. The normalization is performed by dividing the summed position distances by the maximum possible disorder in an array of the same length as the respective feature vector.



Probe

Keycode	PP
io	138
ad	202
dr	220
ne	237
ou	297

Reference

Keycode	PP
ad	178
ou	196
dr	213
io	221
ne	244

Table 5: Calculation of the relative distance between digraphs shared between a probe and reference.

An example of R-distance calculation for PP latencies is shown in Table 5. The example is based on that of the R- and A-distances' original authors [6]. Five of the digraphs from the probe were found in the reference, and so the shared digraphs are sorted by their latencies. The digraphs' summed position distances is $1 + 3 + 0 + 3 + 1 = 8$. With there being five digraphs shared between the probe and reference, the maximum order of the feature vector is $(5^2 - 1)/2 = 12$. Thus, the R-distance is $8/12 = 0.66$. In our PA system, the same procedure would be performed for monograph durations and RP latencies as well.

When an R-distance is produced for each of our three probe feature vectors, the distances are combined. This is done by means of weighed summation. The more durations/latencies are available in a feature vector, the higher weight it is given. This ensures that the feature which has more data to base its R-distance on, and thus is likely to be more accurate, is prioritized.

Equation (3.3) shows the weighted summation where R_n, R_m and R_p are R-distances of durations, PP latencies and RP latencies, respectively. Furthermore, $X = \max(N, M, P)$ where N is the number of recorded durations, while M and P are the respective number of recorded PP and RP latencies divided by 2. This division is performed to avoid an unfair weighing of digraphs, as digraphs produce *two* timing features.

$$R_{\text{total}} = R_n \times \frac{N}{X} + R_m \times \frac{M}{X} + R_p \times \frac{P}{X} \tag{3.3}$$

Where we have based our weighting on total number of durations/latencies per feature vector, the original authors use the *number of shared n-graphs*. The reason for changing this, is that our dataset contains a large variety of behaviors, as the data was collected in an unconstrained environment. A consequence is that some participants were for example playing games during data collection. When playing these games, very few unique keys were pressed, though they were pressed rapidly over long periods of time. This resulted in a high number of occurrences for a single unique digraph.

In the case where a block would consist mostly of only one character being pressed repetitively, which is a realistic situation with our dataset, only a few unique monographs other than the repeated key could heavily shift the weight in favor of monographs if the user waited a couple of seconds before pressing those other keys. This would happen due to the system not recognizing consecutive keypresses with more than 1500 ms RP-latency as digraphs. Therefore, pressing those other keys would only increase the number of unique *monographs* and not unique *digraphs*. A weighting like this could be detrimental, as a great amount of valuable information from the recorded digraphs of the repeated key would have only a small impact on the result.

On the other hand, a consequence of our weighting scheme is that monographs will always be prioritized over digraphs. With a block size of 100 keystrokes, the amount of available monographs would be 100, while there would be at most $100 - 1 = 99$ digraphs. Prioritizing monograph durations is supported by Pinto et al. [15], who found that using the following weights produced the overall best results: 42% for monograph durations, 24% for digraph RP, 16% digraph PP, 10% trigraph PP and 8% tetragraph PP latencies. However, their study was performed on a limited dataset of 10 participants where most of them were software developers.

A-distance

As the R-distance only accounts for *relative* speeds as mentioned in Section 2.2.1, the A-distance considers the *absolute* timing values, and measures the distances between these. When combining the R- and A-distances, we achieve a classifier which both considers *typing rhythm* using the R-distance, as well as raw typing speed.

When calculating the A-distance between a probe and a reference, we reuse the feature vectors from the R-distance calculation, however the fact that they are sorted by duration/latency is irrelevant. To calculate the A-distance, the system first has to decide which n-graph durations/latencies from the probe are to be considered as *similar* to those in the reference. When comparing for instance monographs, the system would regard durations as *similar* if the longer duration divided by the shorter duration is between 1 and a specific threshold. Using standard deviation instead of a fixed threshold would be a viable option, but the fixed threshold allows comparison between monographs with only one occurrence.

Gunetti and Picardi [6] used 1.25 as the threshold for similarity, after testing different thresholds on a subset of their users. While another threshold may be optimal for our dataset, we have also decided to use 1.25 for our system, as achieving an optimal PA or CA performance was not the purpose of this project. The A-distance between a probe and reference for a duration or latency X of an n-graph is defined as follows:

$$A_X = 1 - \frac{\text{number of similar } X\text{-graphs}}{\text{number of shared } X\text{-graphs}}$$

Following the earlier example, Table 6 shows how digraphs from Table 5 are deemed as similar using a threshold of 1.25. Three out of five digraphs are similar, and so the A-distance becomes $A_{\text{PP}} = 1 - 3/5 = 0.4$. Finally, this A-distance would be combined with the A-distances of the monograph durations and the digraph RP-latencies in the same fashion as the R-distance. The R- and A-distances would then be summed to produce the final dissimilarity score of the probe, which would be sent to the decision module. In our example using only PP-latencies, the final distance would be $Dist_{R,A} = 0.66 + 0.4 = 1.06$.

Digraph	**Probe**	**Ref.**	**Calculation**	
io	138	221	$221/138 = 1.60$	
ad	202	178	$202/178 = 1.13$	(similar)
dr	220	213	$220/213 = 1.03$	(similar)
ne	237	244	$244/237 = 1.03$	(similar)
ou	297	196	$297/196 = 1.52$	

Table 6: Similarities between digraphs from Table 5 using a threshold of 1.25.

3.3.3 Decision Module

While Gunetti and Picardi [6] compared the score to the references of all users in the system, Ferreira and Santos [10] proposed a far less expensive method with regards to computational performance. We have adopted this method for our Decision Module, which bases its decisions on a *lockout threshold*. If the dissimilarity score is higher than said threshold, the user is locked out. The threshold is personally adjusted for every genuine user. Similarly to the threshold for penalty or reward in our CA system, the PA system's lockout threshold is calculated by using the user's validation set to find their average dissimilarity score against their own reference and adding a *tolerance level* to the average score. Adjusting the tolerance level allowed us to find a suitable balance between FMR and FNMR. This is further discussed in Section 4.2.

Our approach to finding the user's average score is slightly different to that of Ferreira and Santos [10], as they performed a *leave-one-out cross-validation* (LOOCV). Their structure for references consisted of several 750-length block samples, which easily allowed them to take out a single sample and test its score against the rest of the user's reference. Since our reference structure is not divided into such samples, utilizing the user's validation set for finding their average score was a natural choice.

3.4 Combined system

When merging the CA and PA systems, they can be regarded as *subsystems* of the combined system. As our CA subsystem evaluates every user's behavior and adjusts the trust level accordingly, the PA subsystem waits until enough keystrokes are recorded to fill a block of a specified size. If the trust level is brought below the CA subsystem's lockout threshold before the PA system's block sample is filled, the user is still locked out as if there is no PA subsystem involved. When the user logs back in, the process starts over and the PA subsystem begins waiting for a new block to be filled, starting from the first key pressed after log-in. When testing the performance of our systems, this was simulated by resetting the trust level to its maximum after having dipped below the lockout threshold. If the amount of recorded keystrokes evaluated reaches the PA subsystem's block size without being locked out by the CA subsystem, periodic verification is initiated. The PA subsystem then processes the block of keystrokes as described in Section 3.3. When having constructed a probe from the block sample and evaluated it, the result influences current trust level. This means that the PA system can not directly lock out the user on its own, but rather does so *indirectly* by lowering the trust level, which in turn can cause a lockout.

Two different designs were implemented for the combination of our subsystems. The difference between these two designs is the *fusion level*, or in other words, the way the PA subsystem influences the CA subsystem. Not to be confused with *multi-modal biometrics* and fusion levels associated with such systems, our CA/PA combination is either fused at the *decision level* or the *score level*. Sections 3.4.1 and 3.4.2 describe these fusion levels.

3.4.1 Decision level fusion

In multi-modal biometric systems, *decision level fusion* generally means that the boolean (Match or Non-Match) output of a comparison of a specific *mode* is combined with that of another mode using logical operators. A mode in this context is a combination of a biometric characteristic type, sensor type and processing method [5]. Our combined system is not multi-modal, as both systems are using the same characteristic type and are capturing samples using the keyboard, which can be interpreted as the system's sensor. Instead, our system can be regarded as *multi-algorithmic*.

A diagram of the decision level fusion of the subsystems can be found in Figure 4. When the PA subsystem processes a probe, it passes its decision on to the CA subsystem's Trust Model. De-

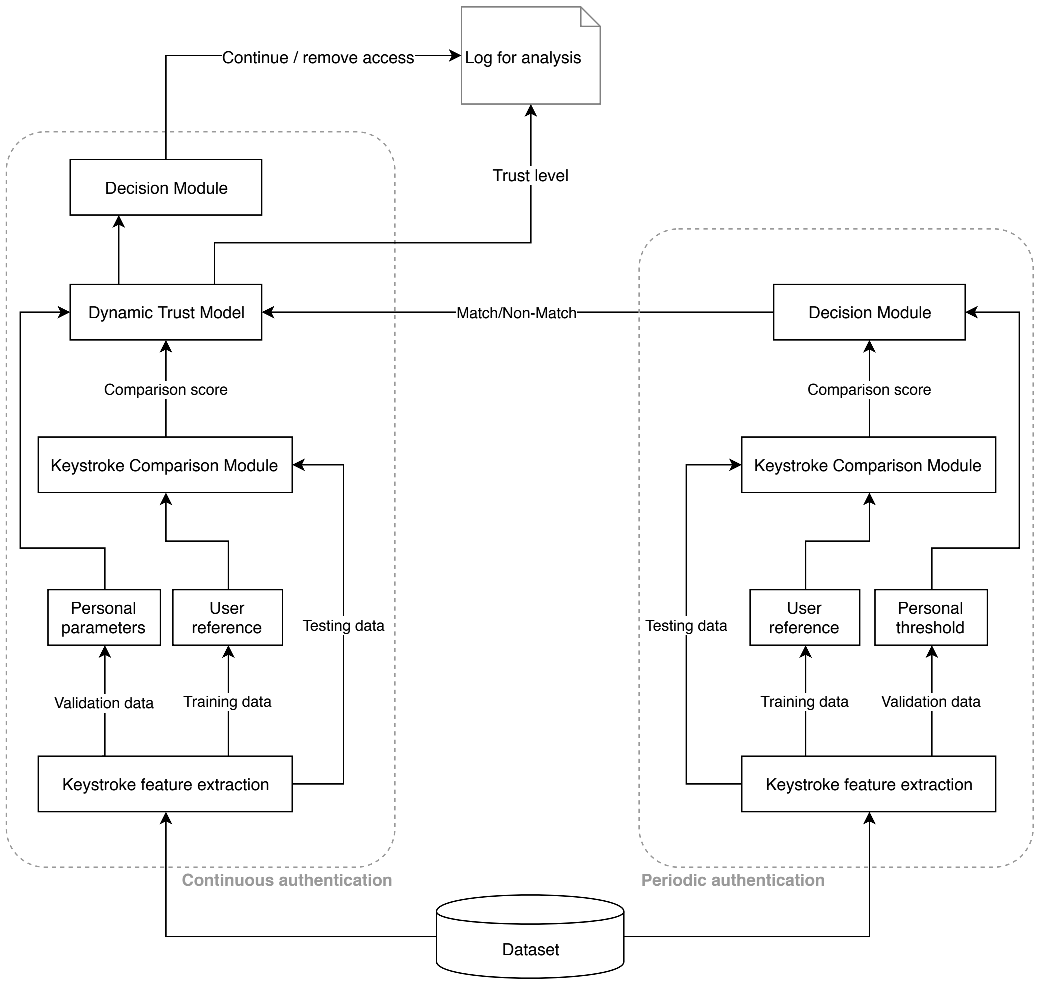


Figure 4: Decision level fusion of the CA and PA subsystems.

pending on whether the PA subsystem's decision is "Match" or "Non-Match", the trust level is raised or lowered by a certain amount. Different values for these changes to the trust level are tested in Chapter 4. The new trust level is fed to the CA subsystem's Decision Module which decides to either let the user continue or lock them out.

3.4.2 Score level fusion

Instead of using the decision of the PA subsystem to influence the trust level, another option is to directly use its comparison score, as seen in Figure 5. This gives the advantage of making smaller changes to the trust level when the classification score is close to the PA subsystem's lockout threshold. An alternative way to look at this is that the PA subsystem's influence on the trust level is *weakened* when it is less confident in the genuineness of the current user. This is achieved by creating a separate sigmoid function Sig_{PA} in the Dynamic Trust Model which takes the PA subsystem's comparison score as one of its inputs. If a probe gets a significantly high dissimilarity score, it may be desirable to let the PA subsystem bring the trust level below the CA subsystem's lockout threshold, even if the current trust is at the maximum level. This can be achieved by configuring the height of the sigmoid function to allow such large influences.

For example, let the CA system's lockout threshold $T_{\text{lockout}} = 40$. The maximum allowed trust level is 100, and so the range of allowed trust levels T_{range} is $100 - 40 = 60$. We can then set the height of the sigmoid to be slightly higher than T_{range}, for example 60.001, such as in the third image of Figure 2. The PA subsystem can then influence the CA system to lock the user out even if the current trust level is 100, as $100 - 60.001 = 39.999$. For that to happen, the PA classification score has be sufficiently higher than the PA subsystem's own lockout threshold, meaning that the recorded behavior largely deviates from that in the reference.

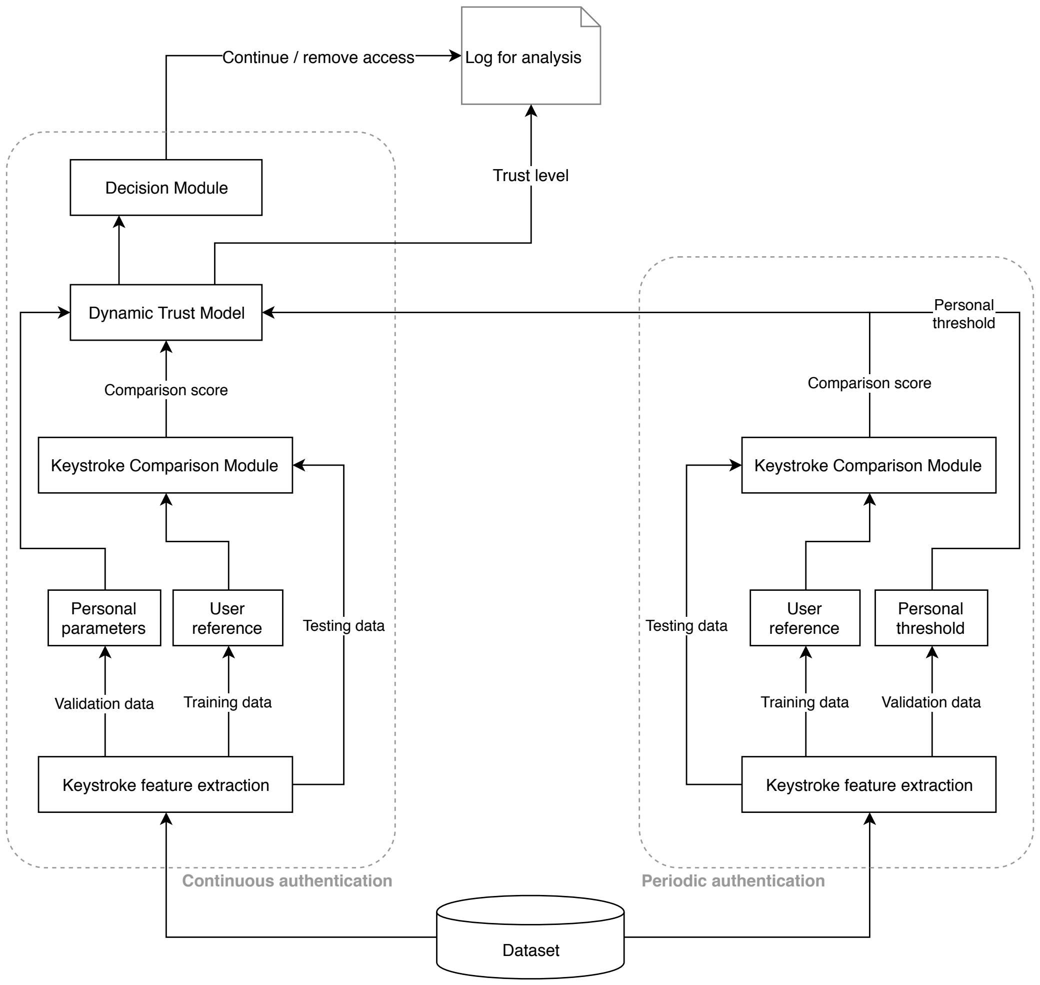


Figure 5: Score level fusion of the CA and PA subsystems.

4 Analysis

We have tested the performance of both individual systems as well as the combined system with several different parameters and settings. This chapter describes these tests, their results and how they affected further analysis. The results are discussed continuously throughout the sections. During testing, we set up every user as a genuine user and ran all other users' test sets against the genuine user's reference, simulating *zero-effort attacks*. In zero-effort attacks, imposters are not actively trying to spoof or imitate the behavior of the genuine user, but rather type in their own speed and rhythm. This describes a scenario where for instance the imposter is unaware of the CA/PA system running in the background of the genuine user's computer.

For every user in our dataset, there is one genuine user (them self) and 45 imposters. As our dataset contains 46 genuine users, this gives us $46 \times 45 = 2070$ imposter runs every time we test the system's performance. We have chosen to categorize and present certain test results similarly to Mondal's [4] result presentations. Using Table 7 as an example, users are then separated into the following four groups.

+/+ : The genuine user was never locked out, and all imposters were locked out at some point, which is the best case scenario.

+/- : The genuine user was never locked out, however at least one imposter was never locked out.

-/+ : The genuine user was locked out at least once, but so were all imposters.

-/- : This is the worst case scenario. The genuine user was locked out at least once, and at least one imposter was never locked out.

We chose to do full test runs for every system configuration, meaning that all 46 users were included, and the full test sets of all imposters were used. If the intention of this project was to propose biometric systems with certain performances using given sets of parameters, doing full test runs like this would lead to *overfitting*. However, our aim was to observe how different ways of combining CA and PA systems affected performance. To observe this, full test runs were needed per configuration. Replicating our systems with the exact same parameters for another dataset is therefore likely to give different results, though one can expect to see similar *effects* on base performances when adjusting parameters in the same way as we have done.

Section 4.1 describes how we tested the CA system and found the configuration to be used in the combined system. The testing of the PA system is described in Section 4.2. The results for the decision level fusion are discussed in Section 4.3, while score level fusion results are presented in Section 4.4. An overview of the best results is given in Section 4.5, before the chapter is concluded with a discussion on computational impact in Section 4.6.

4.1 CA system

The performance analysis of our stand-alone CA system is presented in this section. This includes certain edge-case issues we had to account for, as well as the general performance using various parameters.

4.1.1 Single and no occurrences

Adjusting the fixed score for n-graph features having only a single occurrence (SO) in the reference had minimal impact on detection performance. This was expected, as these are relatively rare. For that reason, it is natural that features from probe actions either have none or several occurrences in the reference more often than only a single occurrence. We observed that adjusting the fixed score for n-graphs missing in the reference had significant impact on the ANIA and ANGA ratings.

For example, when using the sigmoid function seen in Figure 6, we tested adjusting the fixed score for single and no occurrences (NO). The results of these tests can be found in Table 7. The CA system becomes more strict as the SO and NO scores are increased, as this means that the trust levels are affected more negatively. However, it has a larger impact on ANIA than ANGA. This was expected, as the genuine user does not type these n-graphs regularly, but an imposter still migh. Furthermore, we also tested the impact of lowering the SO score while keeping the NO score near maximum. The result of this test can be seen in the last section of Table 7. Compared to the section above, it seems that the NO score has a much larger impact. We can also see that the performance was negatively affected by using a low SO score, as 4 more imposters went undetected, and one user was moved from the -/+ category to the -/- category. Therefore, it seems that having harsh punishments for n-graphs not present or with only a single occurrence in the reference leads to better overall detection performance. We continued testing the system using SO $= 3.0$ and NO $= 3.3$.

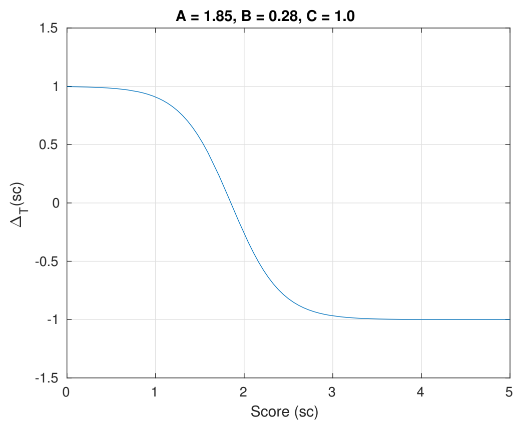


Figure 6: Plot of the sigmoid function used to achieve the results in Table 7.

	Category	#Users	ANGA	ANIA	#Imp. ND
	+/+	6	139010	529	0
SO 2.0	+/-	2	759945	4749	12
NO 2.3	-/+	24	4408	932	0
	-/-	14	5110	3107	33
	Summary	46	8973	1707	45
	+/+	5	13475	283	0
SO 2.3	+/-	1	139525	5252	7
NO 2.6	-/+	35	3022	712	0
	-/-	5	4071	2090	7
	Summary	46	7240	914	14
	+/+	3	9372	251	0
SO 2.6	+/-	1	139525	4335	5
NO 2.9	-/+	38	3041	596	0
	-/-	4	3400	1499	6
	Summary	46	6452	734	11
	+/+	3	9372	234	0
SO 3.0	+/-	1	139525	4014	4
NO 3.3	-/+	38	2671	548	0
	-/-	4	3396	1389	6
	Summary	46	6146	676	10
	+/+	3	9372	249	0
SO 2.3	+/-	1	139525	4747	7
NO 3.3	-/+	37	3064	556	0
	-/-	5	3577	1646	7
	Summary	46	6497	746	14

Table 7: CA results achieved by adjusting Single Occurrence (SO) and No Occurrences (NO) parameters. DTM parameters were $A = 1.85$, $B = 0.28$, $C = 1$ and $T_{\text{lockout}} = 90$.

4.1.2 Outlier removal

In Section 3.2.1, we mentioned that we tested the CA system with and without outlier values in the reference. When testing the system with these outliers removed, we observed a severe drop in both ANGA and ANIA ratings. This essentially means that the system became more *strict*, rapidly locking out both imposters and genuine users. The reason for this lies in how our classifier (SMD) is scaled using standard deviation, which accounts for the dispersion of timing values. The dispersion decreases when removing outliers, and a consequence is that the system expects keystroke features with less distance to the reference than before outlier removal. Therefore, we had to make adjustments to the DTM's system level parameters to account for this change in expected user behavior.

The results in Table 7 were achieved *without* outlier removal. The parameters which gave an ANGA of 6146 and ANIA of 676 gave an ANGA of 1255 and ANIA of 218 when outliers were removed during reference construction. We attempted to lower T_{lockout} from 90 to 80 to make the system less strict, though this resulted in the ANIA rating increasing rapidly compared to the ANGA rating. Specifically, it resulted in 4429 ANGA and 787 ANIA. In other words, imposters gained too much of an advantage compared to the genuine user to justify this method for making the system more liberal.

Lowering the DTM's "A" parameter to 1.3 gave better results, which can be found in Table 8a. These are comparable to the best result from Table 7, being the 6146 ANGA and 676 ANIA. Specifically the fourth row in Table 8a shows a result with 694 ANIA, only 18 actions more than the 676 ANIA without outlier removal. Despite the small difference in ANIA ratings, the result *with* outlier removal gave a significantly higher ANGA rating, as the genuine user could on average perform $8436 - 6146 = 2290$ more keypresses before being locked out. With such positive results, we kept the outlier removal mechanism when combining the CA system with the PA system.

T_{lockout}	ANGA	ANIA	#Imp. ND
80	1615	174	2
70	3437	335	6
60	5385	504	12
50	8436	694	26
40	10846	882	31
30	12208	1103	50
20	12981	1325	62
10	13183	1503	69

(a) Without reference cutoff.

T_{lockout}	ANGA	ANIA	#Imp. ND
80	1569	159	2
70	3283	299	5
60	5297	450	10
50	8362	625	21
40	10622	798	25
30	11871	1009	44
20	12422	1180	52
10	12727	1350	61

(b) With reference cutoff.

Table 8: CA results achieved with outlier removal using the following DTM parameters: $A = 1.3, B = 0.28$ and $C = 1$.

4.1.3 Reference cutoff

When the individual CA and PA systems were implemented, a decision had to be made regarding the amount of keystroke data to be used in reference building. As mentioned in Section 3.1, we use 35% of the user's keystrokes recorded during data collection. We tested the impact of limiting this amount to a maximum of 20000 keystrokes for users with exceptionally large datasets, so that the remaining training data could be used for testing instead. Another motivating factor is that it leads to slightly less variance in reference quality between users.

The results in Table 8 shows the impact of applying the reference cutoff. We observed that both the ANGA and ANIA ratings as well as the number of undetected imposters generally were lowered as a consequence. The difference was still minimal, and we concluded that it was reasonable to continue further analysis *with* the reference cutoff.

4.1.4 Personal and system level parameters

As our PA system has an element of *personalization* in its lockout threshold, we faced the issue of personalizing the CA system as well before combining the systems. The incentive to do so was that incorporating a personalized PA system into a CA system which only uses global parameters might be viewed as an unfair method for improving performance. Therefore, we also built a version of our CA system which uses customized thresholds for rewards/penalties for each user. Section 3.2.3 describes how the thresholds are calculated by using mean scores and tolerance levels. The SO and NO parameters were then locked at 5, to ensure that users were punished for typing rarely or never before seen n-graphs, even with these customized thresholds.

Adjusting the tolerance level led to the results presented in Table 9, where using a tolerance level of 0.5 led to the most reasonable performance. It was achieved with $T_{\text{lockout}} = 50$, and can be compared to the result in Table 8b where also $T_{\text{lockout}} = 50$. Whereas the ANIA dropped by an insignificant amount, the ANGA dropped from 8362 to 8087. Though this was a negative change, the number of undetected imposters also dropped from 21 to 18, which slightly weighs up for the loss of ANGA rating. Overall, these settings gave a satisfying performance and were used as the base when incorporating the PA system.

Toler.	ANGA	ANIA	#Imp. ND
0	268	69	0
0.1	770	93	0
0.2	2107	143	3
0.3	3951	218	3
0.4	5671	381	5
0.5	8087	623	18
0.6	10461	986	36

Table 9: CA results achieved with personal thresholds for reward/penalty. DTM parameters: $A =$ personal, $B = 0.28, C = 1$, and $T_{\text{lockout}} = 50$.

4.2 PA system

4.2.1 Reference cutoff

The PA system's performance was tested with and without the reference cutoff, similarly to the CA tests discussed in Section 4.1.3. Table 10 shows the cutoff's impact on performance. As in the case of the CA system, the impact is negligible. Adjusting the tolerance level allowed us to balance the relation between FNMR and FMR, as mentioned in Section 3.3.3. In other words, this parameter decided the system's strictness.

The users' datasets had to be separated into references, validation sets and tests sets in an identical manner for the CA and PA systems. This was needed due to the combined system using the exact same keystroke data for both the CA and PA subsystems during testing. If the reference cutoff was used in only one of the subsystems, the test sets would contain different keystrokes, as the cutoff causes data from the reference portion of the dataset to be moved into the test portion. The advantages given by the cutoff described in Section 4.1.3 also hold for the PA system. Seeing as the impact of the cutoff was small for both systems, the cutoff was used in further analysis. This included testing of the combined system.

Toler.	FNMR	FMR	#Imp. ND
0.02	47.51	4.68	2
0.06	37.75	5.93	4
0.10	30.25	7.47	4
0.14	22.97	9.23	7
0.18	17.77	11.48	9
0.22	13.40	14.07	13
0.26	9.44	17.11	17
0.30	6.91	20.52	22
0.35	4.70	25.37	37
0.40	2.85	30.80	54
0.45	1.80	36.67	87
0.50	1.29	42.61	121

(a) Without reference cutoff.

Toler.	FNMR	FMR	#Imp. ND
0.02	48.08	4.68	2
0.06	38.54	5.94	4
0.10	30.04	7.50	4
0.14	22.59	9.37	7
0.18	17.36	11.70	9
0.22	12.89	14.28	12
0.26	9.29	17.42	18
0.30	6.44	20.91	21
0.35	3.85	25.88	36
0.40	2.26	31.47	53
0.45	1.51	37.45	88
0.50	1.13	43.55	126

(b) With reference cutoff.

Table 10: Excerpt of PA results showing the performance impact of using a reference cutoff. A block size of 500 keystrokes was used. A complete version on this table is found in Appendix A.1.

4.2.2 R- and A-distance weights

Pinto et al. [15] studied how adjusting the weights of the R-and A-distances when combining them affected detection performance. They found that 20%-80% weights for R- and A-distances was the best configuration for increasing the gap between average genuine and imposter scores while still considering the R-measure. We tested our PA system using the same configuration to see how it affected its performance. The result is illustrated as Detection error tradeoff (DET) curves in Figure 7. The curves show that summing the R-and A-distances with equal weights outperforms the 80-20 weighting scheme for our PA system, as all FNMRs give lower tradeoffs for FMR with the equal weights scheme.

While we did not analyze the underlying reasons, it is worth pointing out that Pinto et al. used five different timing features where our system only uses three. We also used a smaller block size of 500 keystrokes compared to their 750 keystrokes, and as mentioned in Section 3.3.2, their dataset was limited. All of these factors could play a role in why we had less success with their weighting scheme. As we were not looking for the optimal PA solution, we did not test to see if other weights were better. Equal weights were chosen for further analysis due to the superior performance.

4.2.3 Block size

Different block sizes were tested for the PA system. These were 500, 250 and 100 keystrokes. As a dissimilarity score is produced per block, the system has more information to base every dissimilarity score on when the block size is large. This generally leads to more accurate decisions per block, and is reflected in the DET curves in Figure 8. There is a clear difference in detection accuracy between the block lengths that were tested, with 500 outperforming both of the other block sizes, and size 100 achieving the worst performance in terms of detection accuracy.

This does not necessarily imply that using a block size of 500 gives a *better* PA system. When

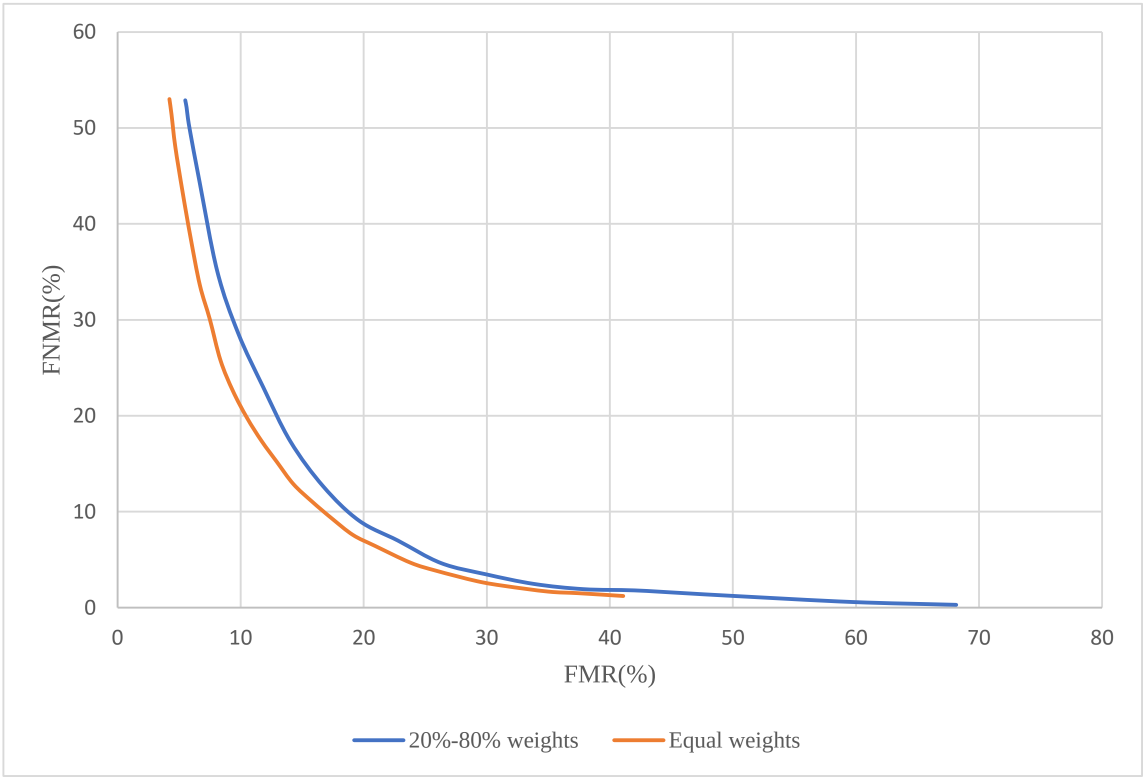


Figure 7: Detection error tradeoff curves showing the detection performance of weighing R- and A-distances by 20-80% respectively as well as using equal weights.

the PA system waits for 500 keystrokes to be collected before processing probes, the imposter has a large window of access to the computer before they are locked out. An example of an unfortunate scenario could be an imposter taking control over an unattended computer after the genuine user had either logged in or been successfully authenticated by the PA system without pressing any keys thereafter. In both of these cases, the imposter would have free reign over the computer until they had typed 500 keystrokes, or until the genuine user returned to the workstation.

At first glance, this may seem like the worst-case scenario. However, the imposter could also take control over the computer in the middle of a block, for example after 300 genuine keystrokes. Ideally, they would be detected and locked out when the block was filled up, i.e. after only $500 - 300 = 200$ imposter keystrokes. However, their window of opportunity could also become larger than 500. The reason for this is that the genuine keystrokes making up the first 300 keystrokes of the block may outweigh the 200 imposter keystrokes, causing the probe to be accepted as *genuine*, i.e. a match. The PA system would then continue collecting keystrokes for the next block without locking the imposter out. Their effective window of opportunity would then be $200 + 500 = 700$,

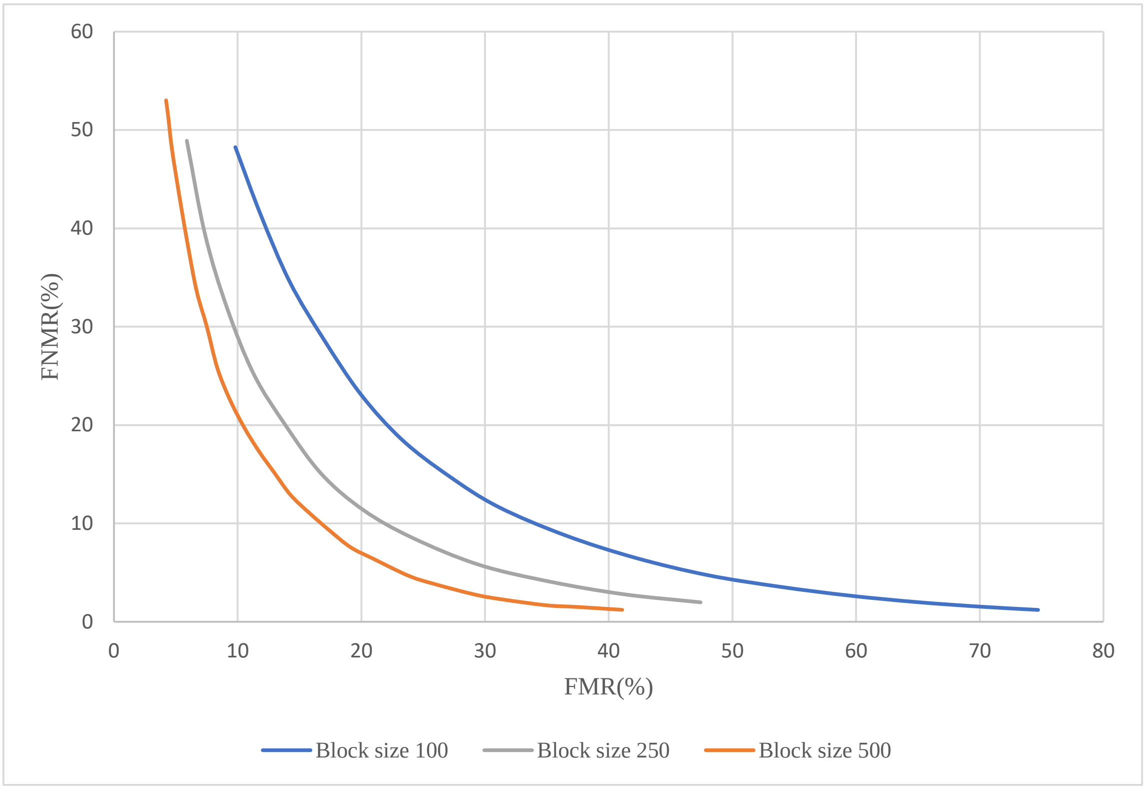


Figure 8: DET curves showing the performance for different block sizes.

giving them increased time to perform potentially harmful actions.

Situations like these highlight fundamental issues with PA systems and why such large block sizes can be problematic. There is a tradeoff between higher detection performance and lower block sizes, which is why we tested the system with several block sizes. All of these block sizes could then be used for testing the combined system later on.

The ANGA and ANIA rates of the PA system are calculated based on FNMR and FMR rates and are presented in Table 11, where the performance of the block sizes can be further compared. An immediate observation is that small blocks sizes give lower ANIA rates than larger block sizes when similar ANGA rates are compared. For example, when comparing results with approximately 2000 ANGA, the respective ANIA rates for block sizes 500, 250 and 100 are around 552, 313 and 192. If we were to judge the performance solely based on ANGA and ANIA rates, it would seem that small block sizes are better. However, the smaller block sizes tend to have significantly more undetected imposters. Also, with higher ANGA rates, the smaller block sizes cause the system to often need more than one block to catch imposters. While this is not necessarily a large issue in a stand-alone PA system with such small block sizes, problems can arise when combined with the CA system. If

Toler.	ANGA	ANIA	#Imp. ND
0.35	12989	675	36
0.33	10670	656	31
0.3	7760	632	21
0.26	5383	605	18
0.22	3880	583	12
0.18	2880	566	9
0.14	2213	552	7
0.1	1664	541	4
0.06	1298	532	4

(a) Block size 500.

Toler.	ANGA	ANIA	#Imp. ND
0.5	12634	476	52
0.45	8957	426	31
0.43	7845	409	27
0.4	6124	386	19
0.35	4302	354	16
0.3	2920	329	11
0.26	2198	313	8
0.22	1665	300	6
0.18	1252	290	2

(b) Block size 250.

Toler.	ANGA	ANIA	#Imp. ND
0.7	8288	395	98
0.65	6531	335	66
0.6	5028	286	38
0.55	3779	247	30
0.45	2112	192	12
0.35	1136	157	3
0.26	672	137	1
0.18	433	125	0
0.1	289	117	0

(c) Block size 100.

Table 11: Excerpt of PA results achieved with different block sizes and tolerance levels.

the PA system can influence the trust level to be *increased*, then the blocks that produce a false match before the imposter is detected will boost the trust level. This can negate the CA system's own progress in detecting the imposter. For instance, if the imposter's current trust level is 60, and $T_{\text{lockout}} = 50$, a falsely matched block could increase the current trust level back to for example 90. This would give the imposter a larger window of opportunity than if the PA system was not involved in the first place, as the CA system probably would have brought the trust level below T_{lockout} a few moments later. Smaller block sizes are therefore not necessarily the better option for the combined system.

4.3 Decision level fusion

This section presents results achieved by combining continuous and periodic authentication at the decision level. The fusion scheme is based on the allowed range of trust T_{range} as described in Section 3.4.2. As the CA configuration chosen for the combined system had $T_{\text{lockout}} = 50$, the allowed range of trust was $T_{\text{range}} = 100 - 50 = 50$. When testing the decision level fusion, the binary decision of the PA subsystem would therefore increase or decrease the trust level by a specific amount between 0-50.

In order to control how much to increase or decrease the trust level, two parameters were used.

They were *'UP'* and *'DOWN'*, each representing how large a portion of T_{range} to increase or decrease the current trust level by, respectively. For instance, with *UP* $= 0.3$ and *DOWN* $= 0.6$, probe blocks producing a match would increase the trust level by $0.3 \times 50 = 15$. Probes producing a non-match would decrease the trust level by $0.6 \times 50 = 30$. Such a configuration would punish imposters more than it would reward the genuine user.

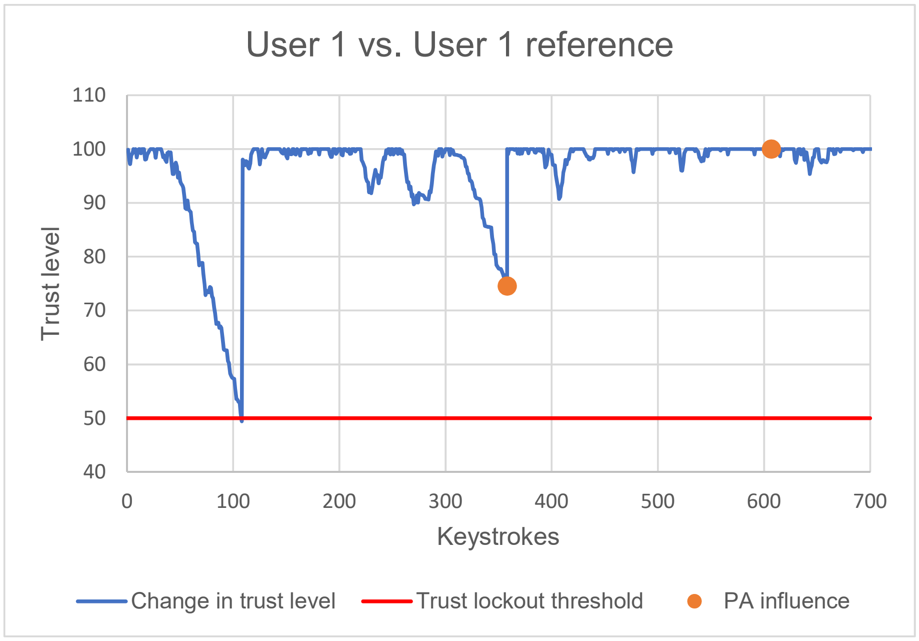
The *UP* and *DOWN* values were set as system level parameters, i.e. they were the same for all users per test. Optimizing these values for each user using genetic algorithms or other optimization techniques is another option. Such algorithms are however computationally expensive and time consuming, and were not used for this project.

An example of a setting that we tested was *UP* $= 1$ and *DOWN* $= 0.6$. Figure 9a shows an excerpt of testing this setting with User 1 as a genuine user, meaning that their test set was used against their own reference. In the beginning, the user seemed to be typing in an unusual manner compared to their reference causing the CA subsystem to rapidly decrease the trust level. After 108 keystrokes, the trust level went below T_{lockout}, marking the undesired event of locking out the genuine user. The block size was 250 in this case, and since the CA subsystem locked out the user before the block was filled up, the PA subsystem did not get a chance to prevent the lockout.

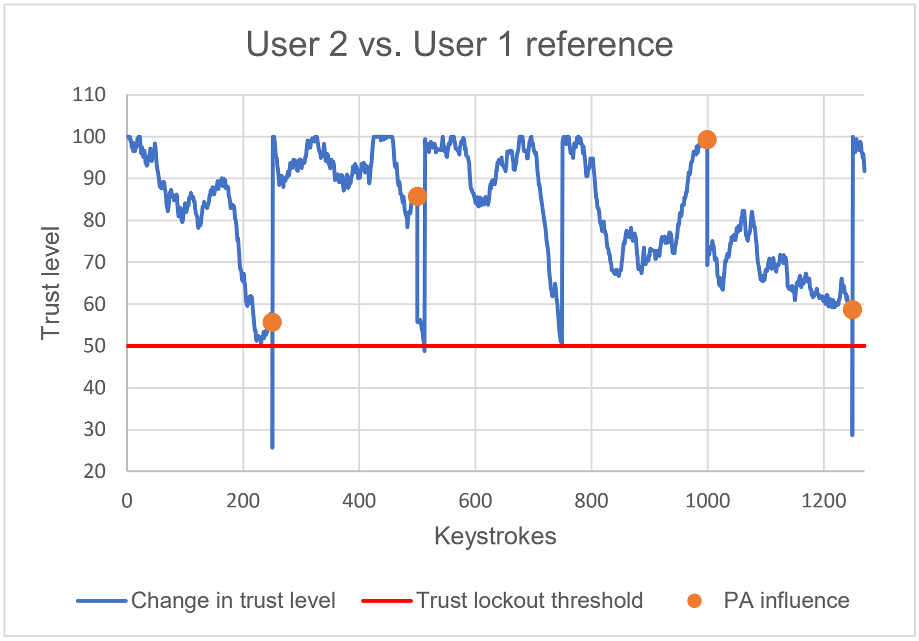
The trust level was brought back up to 100 after the lockout, which simulated the user logging back in and continuing typing. The PA subsystem started collecting keystroke data for a new block from that point onward. After keystroke number 300, the trust level started sinking again, down to 75 at keystroke number 358. At that point, a block of 250 keystrokes had been filled up since the lockout at keystroke number 108. This triggered the PA system to process the block probe, which resulted in a match. Here we can observe the advantage of combining the systems, as the PA match caused the trust level to be increased by $1 \times 50 = 50$, capped at the maximum value of 100. Such positive adjustments have the potential to give the genuine more time before potentially being wrongfully locked out. In this specific example, we can see that another block was filled up, processed and matched after the following 250 keystrokes, however the trust level was at that point already at 100.

Figure 9b shows a portion of a test run where an imposter was locked out four times over the course of around 1250 keystrokes. At keystroke number 231, the trust level dipped to 50.2 before increasing slightly again, barely keeping the imposter logged in. However, at keystroke number 250, the PA subsystem kicked in and reduced the trust level by $0.6 \times 50 = 30$, bringing it far below T_{lockout}. As this meant that the imposter was locked out, the trust level was then brought back to 100, and the test run continued.

Keystroke number 500 shows another example of the cooperation between the two subsystems. The PA subsystem brought the trust level down by 30, which was not quite enough to bring it below the threshold. The CA subsystem was however able lock the user out a few keystrokes later due to the assistance it received just before. Later on, the CA subsystem locked the imposter out at keystroke number 749, just one keystroke before the PA subsystem would have kicked in. After that, the CA subsystem was unable to cause another lockout on its own. However, at keystroke number 999, the PA subsystem brought the trust level down to 70, and later on brought it below



(a) User 1 as genuine user.



(b) User 2 as imposter vs User 1's reference.

Figure 9: Examples of decision level fusion with block size $= 250$. $UP = 1, DOWN = 0.6$.

the threshold at keystroke 1249. Especially the last two PA influences show how using the statistical information available in block probes can be beneficial to utilize in a combined CA/PA system.

4.3.1 Results

Before looking at how the incorporation of the PA system had an influence on the original CA system's performance, we will discuss some observations regarding the impact of the decision level fusion's parameters. The first observation to mention is that adjusting the *DOWN* parameter generally had a larger impact on detection performance than the *UP* parameter. An example of this can be seen in Tables 12a and 12b, as the performance difference between them was negligible even with a considerable difference in *UP* values. However, adjusting the *DOWN* parameter from 0 to 1.001 caused a difference of over 2500 ANGA and over 340 ANIA. The reason for testing *DOWN* = 1.001 was the same as explained in Section 3.4.2; it allowed the PA subsystem's influence to cause a direct lockout even when the current trust level was at its maximum. As seen in the tables, the extra 0.001 in *DOWN* value showed its effect by causing a drop in both ANGA and ANIA. However, the ANGA showed a drop of $6366 - 6018 = 348$ in Table 12b, while the ANIA only dropped by $306 - 298 = 8$. This indicates that letting the PA subsystem lock out users who are currently at trust level 100 has large consequences for genuine users compared to the benefit of locking out imposters faster. This makes sense, as a user at trust level 100 is more likely to be the genuine user than an imposter. When these genuine users are wrongfully brought from trust level 100 to below T_{lockout}, the CA subsystem is given no chance to correct the PA subsystem's mistake. Since imposters spend more time below trust level 100, reducing their trust level by $1 \times T_{\text{range}}$ will often bring them far below T_{lockout} which could be considered an "overkill". Therefore, the extra 0.001 punishment is less likely to make a difference for them than for genuine users. This ties into our *research sub-question 2*, as it seems that having the PA subsystem adjust the trust level is generally better than guaranteeing that a non-match results in an instant lockout. In spite of this, we continued testing *DOWN* = 1.001 to see if similar behavior was shown for other settings as well.

The reason for testing *UP* = 1.001 instead of 1 is simply that we originally tested the system using equal values for *UP* and *DOWN*, and *DOWN* = 1.001 was needed for the reason mentioned above. The extra 0.001 for the *UP* parameter makes no difference in practice, and was only used for convenience. What does however make a difference for the *UP* parameter is giving it a low value, usually below 0.4. At such low values, the tests showed an exception to the general behavior being that adjustment of *UP* makes little to no difference. This can be seen in Table 12c, where *UP = 0.2*. When comparing these results to the other two tables, we can see that it gave lower ANGA ratings for similar ANIA ratings. In other words, using such a low *UP* gave a worse performance across the board.

A likely explanation is that some genuine users have benefited largely by being 'saved' by the PA system at certain times, when their trust level was low. With smaller *UP* values, the PA subsystem had less power in preventing the CA subsystem from wrongfully locking them out, and so the small boost in trust level may not have been enough to keep them logged in for much longer. Therefore, most of the following results presented in this section were achieved using a high *UP* value.

DOWN	ANGA	ANIA	#Imp. ND
0	8547	644	19(0.9%)
0.2	8545	605	18(0.9%)
0.4	8263	564	15(0.7%)
0.6	8089	508	11(0.5%)
0.8	7676	445	8(0.4%)
1	6278	307	3(0.1%)
1.001	6019	299	3(0.1%)

(a) *UP* $= 1.001$

DOWN	ANGA	ANIA	#Imp. ND
0	8543	642	19(0.9%)
0.2	8541	603	18(0.9%)
0.4	8261	563	15(0.7%)
0.6	8088	507	11(0.5%)
0.8	7676	444	8(0.4%)
1	6366	306	3(0.1%)
1.001	6018	298	3(0.1%)

(b) *UP* $= 0.4$

DOWN	ANGA	ANIA	#Imp. ND
0	8134	638	19(0.9%)
0.2	8132	599	18(0.9%)
0.4	7850	558	15(0.7%)
0.6	7676	503	11(0.5%)
0.8	7264	441	8(0.4%)
1	5869	304	3(0.1%)
1.001	5612	296	3(0.1%)

(c) *UP* $= 0.2$

Table 12: Differences in performance when adjusting *DOWN* parameter for different *UP* values. Block size was 500 and PA tolerance was 0.33.

In this analysis, we will mainly focus on two types of performance improvement introduced by the combination, both regarding ANGA and ANIA ratings. Recall that the specific settings used for the CA subsystem resulted in 8087 ANGA and 623 ANIA. Our focus areas are then as follows:

ANGA improvements: For settings giving around 623 ANIA, we are interested in ANGA ratings above 8087.

ANIA improvements: Where the ANGA ratings are around 8087, we are looking for ANIA ratings below 623.

Some other results of interest will also be discussed, such as how far up the combination was able to boost the ANGA beyond the original system while accepting the compromise of a higher ANIA.

Block size 500

After having described the general effects of the *UP* and *DOWN* parameters, we can discuss how the original CA system's performance was affected by incorporating the PA system, which was the main goal of the analysis phase. The first block size tested was 500 keystrokes. From the PA results in Table 11a, we chose to use a tolerance level of 0.33, as it gave an ANGA of 10670 and ANIA of 656. Having such a high ANGA while still having an ANIA comparable to the CA system's ANIA of 623 seemed beneficial for the combined system, and was the reason for choosing this PA setting. The underlying FNMR and FMR rates which were used for calculating the ANGA and ANIA were 4.69% and 23.8%, respectively.

Looking back at Table 12a, we can see that the mentioned PA configuration was indeed able to positively affect the CA system's performance. With $DOWN = 0.2$, the results were a higher ANGA for a slightly lower ANIA. Specifically, the genuine users were able to type $8545 - 8087 = 458$ more keystrokes on average before being locked out, which is an increase of 5.66%. The ANIA was impacted more heavily, as seen where DOWN = 0.6. With an almost identical ANGA to that of the original CA system, the ANIA was lowered by $623 - 508 = 115$ which is an improvement of 18.46%. This means that imposters were caught significantly quicker on average with the combined system, supporting our main research question. Also, the number of imposters that were never detected was reduced from 18 to 11. This setting gave the best result for decreasing ANIA that we were able to find in this project.

Lastly, we can point out that for this block size, the ANGA saturated around 8545. Increasing it any further beyond that point only resulted in a worse ANIA. However, by sacrificing $8087 - 6278 = 1809$ ANGA, the ANIA dropped by 50.72% down to 307, as seen where $DOWN = 1$. With that setting, only 3 out of 2070 imposters were undetected.

Block size 250

We were interested in seeing if smaller block sizes also could improve performance, as it would allow the PA system to engage more often, possibly detecting imposters more effectively. Table 13 shows test results using 250 keystrokes as block size. The PA tolerance used was 0.4, and as seen in Table 11b, it gave 6124 ANGA and 386 ANIA, calculated from an FNMR of 4.08% and FMR of 35.24%. As mentioned in Section 4.2.3, using a PA configuration with an ANIA rating too far from the block size could end up falsely matching imposter blocks too regularly, boosting their trust level and essentially preventing them from being detected by the CA system. This was the reason for not choosing a more liberal setting for this block size.

DOWN	ANGA	ANIA	#Imp. ND
0	8944	706	25(1.2%)
0.1	8944	663	22(1.1%)
0.2	8602	630	20(1%)
0.4	8025	569	16(0.8%)
0.6	7380	499	12(0.6%)
0.8	7062	432	12(0.6%)
1	5604	284	3(0.1%)
1.001	5239	273	2(0.1%)

Table 13: Performance results for block size = 250, PA tolerance = 0.4 and $UP = 1.001$.

Block size 250 seems to be less successful in improving the CA system's ANIA rating compared to block size 500. In the fourth row of Table 13, we see that even though the ANGA was lower than the 8089 ANGA in Table 12a, it had a higher ANIA. Still, an interesting observation is that the lower block size was able to boost the ANGA to 8602 with only $630 - 623 = 7$ higher ANIA than the original CA system, which is also slightly higher than what was achieved with block size

500. Furthermore, it was able to bring the ANGA up to 8944 when accepting an ANIA increase of $663 - 623 = 40$.

Block size 100

For the smallest block size, we used a PA tolerance of 0.26. This had quite low performance ratings, namely 672 ANGA and 137 ANIA, as seen in Table 11c. Again, we saw this as necessary in in order to avoid boosting imposter trust levels too often. The results can be found in Table 14, where we see some similar effects as with block size 250. For reducing ANIA without sacrificing ANGA, this block length was the worst performer of the three block sizes. This is evident by looking at the result where $DOWN = 25$. By comparing this result to block size 250 in Table 13 where $\text{DOWN} = 0.4$, we see a lower ANGA together with a higher ANIA with block size 100.

DOWN	ANGA	ANIA	#Imp. ND
0.1	8992	748	31(1.5%)
0.2	8737	636	23(1.1%)
0.25	7919	581	18(0.9%)
0.4	6815	438	9(0.4%)
0.6	5135	301	5(0.2%)
0.8	3686	224	3(0.1%)
1	973	134	0(0%)
1.001	769	126	0(0%)

Table 14: Performance results for block size = 100, PA tolerance = 0.26 and $UP = 1.001$.

Similarly to block size 250, we also see an ability to boost ANGA for the compromise of higher ANIA. We also see that it was able to bring both ANGA and ANIA ratings far down. This is however likely due to the low PA tolerance of 0.26, which is a fairly strict setting causing probe blocks from both imposters and genuine users to be relatively likely to result in a non-match. Specifically, the underlying FNMR was 14.87% and FMR was 27.02%. The result was therefore a highly sensitive *DOWN* parameter, giving a large range between the most liberal and most strict setting.

Other tolerance levels

Lastly, it should be pointed out that different tolerance levels for each block size was tested to see what effect adjusting it up or down had. In general, this resulted in worse performance, and the previously discussed tolerance levels produced the best results. The results from using other tolerance levels can be found in Appendix B.1.

4.4 Score level fusion

As with the decision level fusion, the score level fusion is also based on T_{range}. During testing, we gave the DTM's sigmoid function for PA influence, Sig_{PA}, a height that allowed a change of trust between 0 and 50.001. Figure 10 shows how the score level fusion allows the PA influence to have varying degrees of impact depending on the dissimilarity scores calculated from probe blocks. The first of the three times the PA system kicked in, it caused a lockout by lowering the trust by 39 levels. It kicked in again at keystroke number 333, lowering the trust by 35, meaning that block was slightly more similar to the reference than the first. While it did not cause a direct lockout, it assisted the CA system enough for it to lock the user out at at keystroke number 535. Lastly, we see the PA system being less decisive in its dissimilarity score, causing a drop of only 7 levels. Because the trust level was already close to T_{lockout}, it still managed to nudge the trust level down just enough to cause a lockout slightly before the CA subsystem would have done so itself.

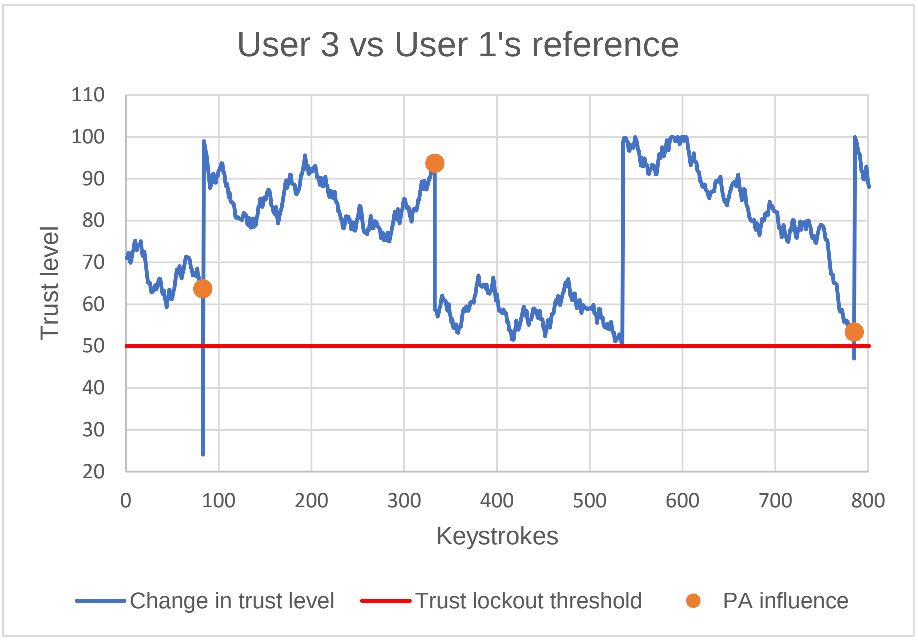


Figure 10: Example of score level fusion with block size $= 250$, where User 3 was tested as an imposter vs User 1's reference. DTM parameters for *Sig*$_{\text{PA}}$ were as follows: $A =$ personal$+0.5$ tolerance, $B = 0.1$, $C = 50.001$.

4.4.1 Results

The performance was tested by adjusting the width of *Sig*$_{\text{PA}}$, as well as the PA tolerance. The tolerance was added to the user's mean score from the validation set, similarly to the decision level fusion, however in this case it controlled the threshold for reward and penalty. Increasing the tolerance essentially shifted *Sig*$_{\text{PA}}$ towards the right. Following are the results achieved by adjusting these parameters for different block sizes.

Block size 500

We were able to find settings improving the ANIA rating of the original CA system also with this fusion scheme. Table 15 shows the performance using different *Sig*$_{PA}$ widths with blocks size 500. There are more results available Appendix C.1, however Table 15 is focused on results where either the ANGA or ANIA is comparable to the original CA system. Similarly to decision level fusion, the ANGA was boosted to around 8500 when the ANIA was around the same as the CA system, being 623. The best widths for boosting the ANGA seemed to be 0.25 and 0.3 for our specific combination and dataset, giving 8520 and 8540 ANGA while having a lower ANIA than the CA system, as well as one less undetected imposter.

Width	**Tolerance**	**ANGA**	**ANIA**	**#Imp. ND**
0.05	0.55	8097	539	13(0.6%)
	0.7	8550	631	18(0.9%)
0.1	0.5	8094	560	13(0.6%)
	0.6	8497	611	15(0.7%)
0.25	0.3	7892	527	11(0.5%)
	0.5	8540	614	17(0.8%)
0.3	0.262	8158	533	11(0.5%)
	0.5	8520	613	17(0.8%)
0.45	0.2	7914	552	14(0.7%)
	0.5	8500	621	18(0.9%)
0.65	0.3	8085	587	16(0.8%)
	0.5	8432	629	18(0.9%)

Table 15: Selected portions of score level fusion results with block size 500, sorted by the width of Sig_{PA}.

Width 0.3 also seems to be the best performer for decreasing ANIA, as it was lowered to 533 while having a higher ANGA than the CA system's 8087. The number of undetected imposters was also seven less. In order to be able to compare the result to the decision level fusion with block size 500, we attempted to lower the tolerance in hopes of bringing the ANGA down to a rating closer to 8087. This resulted in a 7943 ANGA and 530 ANIA, which was worse than the decision level fusion.

Block size 250

We observed that a larger width for Sig_{PA} was needed with block size 250. This was clear, as widths 0.05 and 0.3 decreased the original CA system's performance apart from having one less undetected imposter, as seen in Table 16. Both of these widths had lower ANGAs and higher ANIAs, meaning genuine users were locked out faster, while imposters had larger windows of opportunity. At larger widths, we saw more reasonable results. A possible cause could be the fact that smaller block sizes result in less accurate PA comparisons, as was shown in Figure 8. With larger sigmoid widths, comparison scores will on average cause smaller changes to the trust level. In other words, comparison scores are given more "wiggle room" before the influence on the trust level becomes very large, which prevents inaccurate scores to cause too much damage. While less accurate, the smaller block sizes also cause PA comparison scores to be produced more rapidly. Therefore, reducing the impact that each of these scores have on the trust level by increasing the width seems logical, and we believe this to be a likely explanation.

Table 16 shows that the best result for decreasing ANIA using this block size was achieved with

Width	Tolerance	ANGA	ANIA	#Imp. ND
0.05	0.7	8047	652	17(0.8%)
0.3	0.7	7912	629	17(0.8%)
0.5	0.35	7906	572	15(0.7%)
	0.45	8745	623	21(1%)
0.7	0.2	7946	538	14(0.7%)
	0.5	8705	640	22(1.1%)

Table 16: Selected excepts of score level fusion results with block size 250, sorted by the width of Sig_{PA}. Extended table is found in Appendix C.2.

width = 0.7, and tolerance = 0.2. It gave an ANGA of 7946 and ANIA of 538, which was not as effective as the best ANIA decreasing result with block size 500. On the other hand, we saw positive results for increasing ANGA using block size 250. With width = 0.5, and tolerance = 0.45, it gave an ANGA of 8745 while having the same ANIA as the original system. This was the best ANGA improvement observed in the project's analysis phase, being an 8.14% increase.

Just as with decision level fusion, we see that smaller block sizes are better at improving ANGA, while larger block sizes are better at decreasing ANIA. Before the analysis, we expected otherwise. Specifically, we expected large block sizes to be better for reducing ANIA and shorter block lengths to be better for increasing ANGA. Because the CA subsystem has the ability to rapidly drop imposters' trust levels and locking them out after a small amount of keystrokes. Therefore, we expected large block sizes to cause the PA subsystem to kick in too seldom to significantly increase imposter detection rates, since blocks are reset every time the CA subsystem causes a lockout on its own. More research is needed to conclude on why larger block sizes seem better for lowering ANIA ratings.

Block size 100

We performed limited testing of block size 100 with score level fusion. However, we did test a number of different sigmoid widths, and the results in Table 17 show that for ANGA values close to that of the CA system, it showed worse ANIA ratings for all widths. We were unable to find a setting with this block size which improved the CA system's performance. It is possible that increasing the width of Sig_{PA} is not enough to reduce the impact of inaccurate comparisons. This issue could be interesting to investigate further. Reducing the height of the sigmoid function to forcibly restrict the PA system from causing too large changes to the trust level could yield different results.

4.5 Overview of the best results

We conclude the analysis of detection performance by presenting an overview the best results for increasing ANGA and decreasing ANIA, both of which have been discussed in Sections 4.3.1 and 4.4.1. Here, we can see how the users fell into different categories before and after combining the systems. Table 18 shows the detailed results of the original CA system. When comparing it to the best result for increasing ANGA, in the upper half of Table 19, we see that the distribution of users in different

Width	**Tolerance**	**ANGA**	**ANIA**	**#Imp. ND**
0.05	0.8	8043	935	35(1.7%)
0.15	0.6	8010	771	30(1.4%)
0.25	0.4	6990	576	22(1.1%)
0.3	0.5	7927	715	29(1.4%)
0.5	0.4	7866	649	25(1.2%)
0.7	0.4	8015	652	23(1.1%)
0.9	0.4	8063	656	21(1%)
1.2	0.4	7981	656	22(1.1%)

Table 17: Selected excepts of score level fusion results with block size 100, sorted by the width of Sig_{PA}.

categories is very similar. One user was moved from -/+ to +/+, which was a small but positive change. This means that the combined system cause one more user to never be locked out. While the total number of imposters not detected went up by 0.1%, the genuine user was on average able to type 658 characters more, an increase of 8.14%.

In the lower half of Table 19, we see that the best result for decreasing ANIA showed some more changes on user categories. This setting also had one more user in the best category compared to the CA system. There were also six less users who had at least one undetected imposter, as seen in the decreases of users in the +/- and -/- categories. The total number of imposters who were not detected was therefore decreased, down by 7. This, in addition to imposters on average being able to type $623 - 508 = 115$ less characters before being locked out, was overall a positive result.

Category	#Users	ANGA	ANIA	#Imp. ND
+/+	6	17409	269	0
+/-	1	12256	485	1
-/+	30	6682	390	0
-/-	9	6091	1650	17
Summary	46	8087	623	18(0.9%)

Table 18: Detailed performance results of the CA configuration used in the combined system.

4.6 Computational impact

Regarding *research sub-question 1*, we realize that computation speeds are highly dependent on a number of factors such as processing power, programming language, choice of classifier and number of considered features. Still, we can mention that our CA system performs a comparison of a single keystroke in 0.2 milliseconds on average. For our PA system, the average computing time for processing blocks of 500, 250 and 100 keystrokes was 75.2, 46 and 25.7 milliseconds, respectively. These times were collected by finding the user with the largest reference in terms of unique mono- and digraphs, and running their validation set against their own reference. The systems were developed in MATLAB 2017, and were run on an Intel Core i7, on a single thread

	Category	#Users	ANGA	ANIA	#Imp. ND
	+/+	7	21735	336	
Score level	+/-	1	12256	483	1
Block size 250	-/+	29	6130	328	
	-/-	9	6678	1815	20
	Summary	46	8745	623	21(1%)
	+/+	7	21979	308	
Decision level	+/-	0			
Block size 500	-/+	35	5328	353	
	-/-	4	7945	2214	11
	Summary	46	8089	508	11(0.5%)

Table 19: CA results achieved by adjusting Single Occurrence (SO) and No Occurrences (NO) parameters. DTM parameters were $A = 1.85$, $B = 0.28$, $C = 1$ and $T_{\text{lockout}} = 90$.

at 2.4 GHz frequency. While the speeds were reasonable, they could be further improved by for example using more a low-level programming language, not to mention multi-threading.

What is most interesting to discuss is the computational impact of combining the systems. The most important factor here is the PA subsystem's block size. In an example where the combined system has to process 100,000 keystrokes and the block size is 100, a crude way to calculate the number of times the PA system could kick in is simply $100,000/100 = 1000$, which is the maximum. For block size 250 and 500, it could kick in at most 400 and 200 times, respectively. However, what must also be considered is the fact that the likelihood of the CA subsystem locking out the user before a block is filled up increases with larger block sizes. When that happens, the PA subsystem resets and waits until enough keystrokes are recorded to fill a new block. This further reduces computational costs when using large block sizes.

It is also important to discuss the difference between the tests we have performed and a real-time system. During testing, we ran both the CA and PA subsystems on the same thread. This was to avoid having the CA subsystem continuing to adjust the trust level while the PA subsystem was still processing a block probe. If that were to happen, our results could have been less accurate. An example could be the CA subsystem increasing an imposter's trust level before the PA subsystem had finished processing a block, and that increase of trust preventing the PA subsystem's influence from locking the user out. Situations like these could therefore alter the performance readings. This would not matter in a deployed real-time system as the objective would simply be to lock out imposters, and not to measure performance. Therefore, the PA subsystem could process a block parallel to the CA system processing keystrokes with multithreading, further reducing computational impact.

5 Conclusion

Throughout the work of this thesis, we have investigated the possibility of combining continuous and periodic authentication in biometric systems using keystroke dynamics. Our approach was to develop a CA system and a PA system, both with foundations based on designs from literature, and having them cooperate to detect imposters. The CA system, which evaluated the user's typing behavior after every keystroke, had the ability to lock out imposters after only a few keystrokes and worked as the base system. The PA system would wait until enough keystrokes were recorded to fill a block of a certain length, so that it could utilize statistical data for evaluating the typing behavior. It would then feed information to the CA system, influencing its trust in the current user's genuineness.

Two system architectures were proposed, both showing promising results. The main difference between them lied in the information that was passed from the PA system to the CA system. The first architecture was a decision level fusion scheme, and had the PA system make independent decisions, producing a Match or Non-Match result per probe block. This decision was then used to change the trust level by a fixed amount. The second architecture was a score level fusion scheme, where the PA system did not attempt to make any decision on whether or not the current user was genuine, but rather just sent a comparison score to the CA system. That score was then used to influence the trust level in a more dynamic manner than the first architecture.

Using an existing dataset of 46 users where keystroke data was recorded in the background in an uncontrolled environment, testing was performed using three different block lengths for the PA subsystem. Results showed that incorporating the PA system into the CA system allowed for quite large adjustments to the general strictness of the system. For settings giving the same ANIA rating as the original CA system, the best result for increasing ANGA was found with the score level fusion scheme at 250 keystroke block length. It gave an 8.14% improvement in the amount of keystrokes a user could type on average before being wrongfully locked out. When configuring the combined system to have a similar ANGA rating to the CA system, the decision level fusion scheme with block length 500 gave the best result for decreasing ANIA. It was able to cut down the average number of keystrokes that imposters could type by 18.46%. We find that these results support our main research question regarding whether the performance of a CA system can be improved by incorporating a PA system.

5.1 Future Work

There are a number of areas we would like to see further researched, both to resolve limitations of our own work, as well as taking this topic into new directions. Firstly, since we wanted to see how different parameters to our systems impacted performance, we had to manually test a large number of configurations, using the full dataset for each test. We prioritized seeing these effects over attempting to find the optimal settings. As there is an unfathomable amount of possible combinations of parameters, especially for the score level fusion, we were unable to test them all. The tests we performed indicated that the two fusion schemes were able to give quite similar improvements to performance. Therefore, we were unable to conclude which of the two architectures were the best overall. It would therefore be interesting to test the systems where they are customized automatically for each user by utilizing an optimization algorithm. We would also like to see if combining CA and PA systems with state-of-the-art detection performances would differ largely from our results.

Another issue is that we have only tested zero-effort attacks, which is a common limitation throughout most of the literature on CA/PA using keystroke dynamics. It is unclear how our results would be affected if imposters were actively trying to mimic the typing behavior of genuine users. It would be interesting to test our individual systems as well as the combined system using a dataset where such attacks are available. Also, running tests where genuine user and imposter data is mixed could also give a more realistic attack scenario. We could then simulate the genuine user leaving their workstation in the middle of a block, and an imposter taking control from that point. This would make it harder for the PA subsystem to detect the imposter.

There are a few variations to the PA subsystem we would like to explore. One of them is using a *sliding window* when analyzing blocks, where it would not wait until an entire new block had been filled since the last time it kicked in. For example, with a block size of 500, after having influenced the trust level once, it could kick in again after 150 keystrokes. Then, the new block would be the last 350 keystrokes from the previous block plus the new 150 keystrokes. The other variation could be to have the PA system kick in whenever the CA subsystem brings the trust level below the threshold, to see if it agrees before a final decision is made. These two variations could also be combined. It is however worth mentioning that these solutions would increase computational impact as they would cause the PA system to kick in at higher rates.

Lastly, we would like to test our proposed architectures with systems using other biometric characteristics where continuous and periodic authentication is applicable. An example could be extending an existing CA system for mouse dynamics [41] with a PA system, or even a combination of mouse and keystroke dynamics [42]. Gait or voice recognition could also be viable options to explore.

Bibliography

[1] Nilsen, P.-K. Dec 2017. A survey of periodic authentication systems based on keyboard dynamics. *Unpublished work*. IMT4215 Specialization Project. NTNU in Gjøvik.

[2] Dowland, P. S., Furnell, S. M., & Papadaki, M. *Keystroke Analysis as a Method of Advanced User Authentication and Response*, 215–226. Springer US, Boston, MA, 2002. URL: https://doi.org/10.1007/978-0-387-35586-3_17, doi:10.1007/978-0-387-35586-3_17.

[3] Bours, P. 2012. Continuous keystroke dynamics: A different perspective towards biometric evaluation. *Information Security Technical Report*, 17(1), 36 – 43. Human Factors and Bio-metrics. URL: http://www.sciencedirect.com/science/article/pii/S1363412712000027, doi:https://doi.org/10.1016/j.istr.2012.02.001.

[4] Mondal, S. *Continuous User Authentication and Identification: Combination of Security & Forensics*. PhD thesis, NTNU, Feb 2016. Pages 67-73. Available online (BIBSYS): http://hdl.handle.net/11250/2388087.

[5] Information technology — vocabulary — part 37: Biometrics. Standard, International Organization for Standardization, Geneva, CH, February 2017.

[6] Gunetti, D. & Picardi, C. August 2005. Keystroke analysis of free text. *ACM Trans. Inf. Syst. Secur.*, 8(3), 312–347. URL: http://doi.acm.org/10.1145/1085126.1085129, doi:10.1145/1085126.1085129.

[7] Bours, P. & Mondal, S. 2015. Performance evaluation of continuous authentication systems. *IET Biometrics*, 4(4), 220–226. doi:10.1049/iet-bmt.2014.0070.

[8] Davoudi, H. & Kabir, E. Oct 2009. A new distance measure for free text keystroke authentication. In *2009 14th International CSI Computer Conference*, 570–575. doi:10.1109/CSICC.2009.5349640.

[9] Davoudi, H. & Kabir, E. Dec 2010. Modification of the relative distance for free text keystroke authentication. In *2010 5th International Symposium on Telecommunications*, 547–551. doi:10.1109/ISTEL.2010.5734085.

[10] Ferreira, J. & Santos, H. Oct 2012. Keystroke dynamics for continuous access control enforcement. In *2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 216–223. doi:10.1109/CyberC.2012.43.

[11] Hu, J., Gingrich, D., & Sentosa, A. May 2008. A k-nearest neighbor approach for user authentication through biometric keystroke dynamics. In *2008 IEEE International Conference on Communications*, 1556–1560. doi:10.1109/ICC.2008.301.

[12] Huang, J., Hou, D., & Schuckers, S. Feb 2017. A practical evaluation of free-text keystroke dynamics. In *2017 IEEE International Conference on Identity, Security and Behavior Analysis (ISBA)*, 1–8. doi:10.1109/ISBA.2017.7947695.

[13] Kolakowska, A. *User Authentication Based on Keystroke Dynamics Analysis*, 667–675. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. URL: https://doi.org/10.1007/978-3-642-20320-6_68, doi:10.1007/978-3-642-20320-6_68.

[14] Messerman, A., Mustafić, T., Camtepe, S. A., & Albayrak, S. Oct 2011. Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics. In *2011 International Joint Conference on Biometrics (IJCB)*, 1–8. doi:10.1109/IJCB.2011.6117552.

[15] Pinto, P., Patrão, B., & Santos, H. *Free Typed Text Using Keystroke Dynamics for Continuous Authentication*, 33–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. URL: https://doi.org/10.1007/978-3-662-44885-4_3, doi:10.1007/978-3-662-44885-4_3.

[16] 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 *CVPR 2011 WORKSHOPS*, 31–38. doi:10.1109/CVPRW.2011.5981729.

[17] Kang, P. & Cho, S. 2015. Keystroke dynamics-based user authentication using long and free text strings from various input devices. *Information Sciences*, 308(Supplement C), 72 – 93. URL: http://www.sciencedirect.com/science/article/pii/S0020025514009062, doi:https://doi.org/10.1016/j.ins.2014.08.070.

[18] Kaneko, Y., Kinpara, Y., & Shiomi, Y. July 2011. A hamming distance-like filtering in keystroke dynamics. In *2011 Ninth Annual International Conference on Privacy, Security and Trust*, 93–95. doi:10.1109/PST.2011.5971969.

[19] Monrose, F. & Rubin, A. 1997. Authentication via keystroke dynamics. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, CCS '97, 48–56, New York, NY, USA. ACM. URL: http://doi.acm.org/10.1145/266420.266434, doi:10.1145/266420.266434.

[20] Harun, N., Woo, W. L., & Dlay, S. S. May 2010. Performance of keystroke biometrics authentication system using artificial neural network (ann) and distance classifier method. In *Computer and Communication Engineering (ICCCE), 2010 International Conference on*, 1–6. doi:10.1109/ICCCE.2010.5556852.

[21] Tappert, C. C., Cha, S.-H., Villani, M., & Zack, R. S. January 2010. A keystroke biometric systemfor long-text input. *Int. J. Inf. Sec. Priv.*, 4(1), 32–60. URL: http://dx.doi.org/10.4018/jisp.2010010103, doi:10.4018/jisp.2010010103.

[22] Park, S., Park, J., & Cho, S. June 2010. User authentication based on keystroke analysis of long free texts with a reduced number of features. In *2010 Second International Conference on Communication Systems, Networks and Applications*, volume 1, 433–435. doi:10.1109/ICCSNA.2010.5588979.

[23] Ganzhorn, D., Lu, D., Ordal, P., Norwood, J., & Fong, W. Sep 2005. Continuous identity verification through keyboard biometrics. *Journal of Undergraduate Research*, 1. URL: http://hdl.handle.net/1802/4667.

[24] Locklear, H., Govindarajan, S., Sitová, Z., Goodkind, A., Brizan, D. G., Rosenberg, A., Phoha, V. V., Gasti, P., & Balagani, K. S. Sept 2014. Continuous authentication with cognition-centric text production and revision features. In *IEEE International Joint Conference on Biometrics*, 1–8. doi:10.1109/BTAS.2014.6996227.

[25] Ahmed, A. A. & Traore, I. April 2014. Biometric recognition based on free-text keystroke dynamics. *IEEE Transactions on Cybernetics*, 44(4), 458–472. doi:10.1109/TCYB.2013.2257745.

[26] Kim, J., Kim, H., & Kang, P. 2017. Keystroke dynamics-based user authentication using freely typed text based on user-adaptive feature extraction and novelty detection. *Applied Soft Computing*. URL: http://www.sciencedirect.com/science/article/pii/S1568494617305847, doi:https://doi.org/10.1016/j.asoc.2017.09.045.

[27] Solami, E. A., Boyd, C., Clark, A., & Ahmed, I. Sept 2011. User-representative feature selection for keystroke dynamics. In *2011 5th International Conference on Network and System Security*, 229–233. doi:10.1109/ICNSS.2011.6060005.

[28] Wu, P. Y., Fang, C. C., Chang, J. M., & Kung, S. Y. Nov 2017. Cost-effective kernel ridge regression implementation for keystroke-based active authentication system. *IEEE Transactions on Cybernetics*, 47(11), 3916–3927. doi:10.1109/TCYB.2016.2590472.

[29] Alsultan, A., Warwick, K., & Wei, H. 2017. Non-conventional keystroke dynamics for user authentication. *Pattern Recognition Letters*, 89, 53 – 59. URL: http://www.sciencedirect.com/science/article/pii/S0167865517300429, doi:https://doi.org/10.1016/j.patrec.2017.02.010.

[30] Shimshon, T., Moskovitch, R., Rokach, L., & Elovici, Y. Dec 2010. Continuous verification using keystroke dynamics. In *2010 International Conference on Computational Intelligence and Security*, 411–415. doi:10.1109/CIS.2010.95.

[31] Monaco, J. V., Bakelman, N., Cha, S. H., & Tappert, C. C. Aug 2013. Recent advances in the development of a long-text-input keystroke biometric authentication system for arbitrary text input. In *2013 European Intelligence and Security Informatics Conference*, 60–66. doi:10.1109/EISIC.2013.16.

[32] Stewart, J. C., Monaco, J. V., Cha, S. H., & Tappert, C. C. Oct 2011. An investigation of keystroke and stylometry traits for authenticating online test takers. In *2011 International Joint Conference on Biometrics (IJCB)*, 1–7. doi:10.1109/IJCB.2011.6117480.

[33] Janakiraman, R. & Sim, T. *Keystroke Dynamics in a General Setting*, 584–593. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. URL: https://doi.org/10.1007/978-3-540-74549-5_62, doi:10.1007/978-3-540-74549-5_62.

[34] Dowland, P. S. & Furnell, S. M. 2004. A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In *Security and Protection in Information Processing Systems*, Deswarte, Y., Cuppens, F., Jajodia, S., & Wang, L., eds, 275–289, Boston, MA. Springer US.

[35] Hempstalk, K. *Continuous typist verification using machine learning*. PhD thesis, The University of Waikato, Hamilton, New Zealand, 2009. Available online (BIBSYS): https://hdl.handle.net/10289/3282.

[36] Filho, J. R. M. & Freire, E. O. 2006. On the equalization of keystroke timing histograms. *Pattern Recognition Letters*, 27(13), 1440 – 1446. URL: http://www.sciencedirect.com/science/article/pii/S0167865506000602, doi:https://doi.org/10.1016/j.patrec.2006.01.010.

[37] Monaco, J. V. & Tappert, C. C. 2018. The partially observable hidden markov model and its application to keystroke dynamics. *Pattern Recognition*, 76, 449 – 462. URL: http://www.sciencedirect.com/science/article/pii/S0031320317304752, doi:https://doi.org/10.1016/j.patcog.2017.11.021.

[38] Chang, J. M., Fang, C. C., Ho, K. H., Kelly, N., Wu, P. Y., Ding, Y., Chu, C., Gilbert, S., Kamal, A. E., & Kung, S. Y. July 2013. Capturing cognitive fingerprints from keystroke dynamics. *IT Professional*, 15(4), 24–28. doi:10.1109/MITP.2013.52.

[39] Villani, M., Tappert, C., Ngo, G., Simone, J., Fort, H. S., & Cha, S.-H. June 2006. Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, 39–39. doi:10.1109/CVPRW.2006.115.

[40] Killourhy, K. S. & Maxion, R. A. June 2009. Comparing anomaly-detection algorithms for keystroke dynamics. In *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, 125–134. doi:10.1109/DSN.2009.5270346.

[41] Mondal, S. & Bours, P. Sept 2013. Continuous authentication using mouse dynamics. In *2013 International Conference of the BIOSIG Special Interest Group (BIOSIG)*, 1–12.

[42] Mondal, S. & Bours, P. 2017. A study on continuous authentication using a combination of keystroke and mouse biometrics. *Neurocomputing*, 230, 1 – 22. URL: http://www.sciencedirect.com/science/article/pii/S0925231216314321, doi:https://doi.org/10.1016/j.neucom.2016.11.031.

A PA testing data

Following are results from testing the PA system which were not represented in the main thesis.

A.1 Reference cutoff impact

In Table 10, we only showed a excerpt of the results from testing the impact of using a reference cutoff with block size 500. Here we present the complete version of said table.

Toler.	FNMR	FMR	#Imp. ND
0.001	51.98	4.20	2(0.1%)
0.003	51.43	4.24	2(0.1%)
0.005	51.15	4.30	2(0.1%)
0.008	50.32	4.36	2(0.1%)
0.02	47.51	4.68	2(0.1%)
0.04	43.05	5.29	3(0.1%)
0.06	37.75	5.93	4(0.2%)
0.08	33.47	6.66	4(0.2%)
0.1	30.25	7.47	4(0.2%)
0.12	26.47	8.30	5(0.2%)
0.14	22.97	9.23	7(0.3%)
0.16	19.66	10.31	8(0.4%)
0.18	17.77	11.48	9(0.4%)
0.2	15.65	12.71	11(0.5%)
0.22	13.40	14.07	13(0.6%)
0.24	11.46	15.53	14(0.7%)
0.26	9.44	17.11	17(0.8%)
0.28	8.06	18.77	20(1%)
0.3	6.91	20.52	22(1.1%)
0.33	5.48	23.34	32(1.5%)
0.35	4.70	25.37	37(1.8%)
0.38	3.64	28.60	45(2.2%)
0.4	2.85	30.80	54(2.6%)
0.43	2.07	34.24	76(3.7%)
0.45	1.80	36.67	87(4.2%)
0.48	1.34	40.23	108(5.2%)
0.5	1.29	42.61	121(5.8%)

(a) Without reference cutoff.

Toler.	FNMR	FMR	#Imp. ND
0.001	53.01	4.21	2(0.1%)
0.003	52.38	4.26	2(0.1%)
0.005	52.18	4.31	2(0.1%)
0.008	51.38	4.38	2(0.1%)
0.02	48.08	4.68	2(0.1%)
0.04	43.35	5.28	3(0.1%)
0.06	38.54	5.94	4(0.2%)
0.08	33.64	6.68	4(0.2%)
0.1	30.04	7.50	4(0.2%)
0.12	25.77	8.36	5(0.2%)
0.14	22.59	9.37	7(0.3%)
0.16	19.79	10.51	8(0.4%)
0.18	17.36	11.70	9(0.4%)
0.2	15.19	12.94	10(0.5%)
0.22	12.89	14.28	12(0.6%)
0.24	11.05	15.81	14(0.7%)
0.26	9.29	17.42	18(0.9%)
0.28	7.57	19.13	20(1%)
0.3	6.44	20.91	21(1%)
0.33	4.69	23.80	31(1.5%)
0.35	3.85	25.88	36(1.7%)
0.38	2.76	29.20	45(2.2%)
0.4	2.26	31.47	53(2.6%)
0.43	1.67	34.98	73(3.5%)
0.45	1.51	37.45	88(4.3%)
0.48	1.21	41.09	111(5.4%)
0.5	1.13	43.55	126(6.1%)

(b) With reference cutoff.

Table 20: Complete version of Table 10.

B Testing data from decision level fusion

B.1 Other PA tolerance levels

Different PA tolerance levels were tested for the different block lengths when testing the decision level fusion. Below are the resulting performance ratings when using tolerance levels other than those covered in Section 4.3.1.

DOWN	ANGA	ANIA	#Imp. ND
0.1	8550	648	20(1%)
0.2	8550	628	20(1%)
0.4	8436	590	17(0.8%)
0.6	8325	535	13(0.6%)
0.7	8057	510	10(0.5%)
0.8	7976	481	10(0.5%)
1	7690	354	3(0.1%)
1.001	7543	345	3(0.1%)

(a) PA tolerance $= 0.4$, *UP* $= 1.001$. PA subsystem's ANGA was 22130 and ANIA was 730.

DOWN	ANGA	ANIA	#Imp. ND
0.1	8460	611	18(0.9%)
0.2	8458	591	18(0.9%)
0.4	7932	543	14(0.7%)
0.6	7432	474	8(0.4%)
0.8	7009	392	5(0.2%)
1	4211	265	2(0.1%)
1.001	2936	256	2(0.1%)

(b) PA tolerance $= 0.22$, *UP* $= 1.001$. PA subsystem's ANGA was 3880 and ANIA was 583.

Table 21: Other PA tolerance levels with block size 500

DOWN	ANGA	ANIA	#Imp. ND
0.2	8812	655	22(1.1%)
0.4	8293	606	18(0.9%)
0.6	7544	555	15(0.7%)
0.8	7263	486	13(0.6%)
1.001	6560	348	4(0.2%)

(a) PA tolerance $= 0.5$, *UP* $= 0.4$. PA subsystem's ANGA was 12634 and ANIA was 476.

DOWN	ANGA	ANIA	#Imp. ND
0.1	8388	594	18(0.9%)
0.2	7963	548	15(0.7%)
0.4	6859	459	8(0.4%)
0.6	6385	369	5(0.2%)
0.8	4933	291	4(0.2%)
1	1152	173	1(0%)
1.001	872	167	1(0%)

(b) PA tolerance $= 0.14$, *UP* $= 1.001$. PA subsystem's ANGA was 993 and ANIA was 282.

Table 22: Other PA tolerance levels with block size 250

DOWN	ANGA	ANIA	#Imp. ND
0.1	9554	886	41(2%)
0.2	9128	782	33(1.6%)
0.4	7568	609	23(1.1%)
0.6	7174	447	13(0.6%)
0.8	5335	320	6(0.3%)
1	3030	197	1(0%)
1.001	2151	189	0(0%)

Table 23: PA tolerance level 0.4 with block size 100. PA subsystem's ANGA was 1539 and ANIA was 173.

C Testing data from score level fusion

Here, we show all results from testing the score level fusion. The results are categorized by block sizes, and the following tables are extensions of the tables displayed in Section 4.4.1.

C.1 Block size 500

Table 24: Complete version of Table 15.

Width	Toler.	ANGA	ANIA	#Imp. ND
0.05	0.4	7936	449	9(0.4%)
	0.5	8026	510	12(0.6%)
	0.55	8097	539	13(0.6%)
	0.6	8212	572	13(0.6%)
	0.7	8550	631	18(0.9%)
	0.8	8550	670	22(1.1%)
0.1	0	5424	285	3(0.1%)
	0.1	6186	333	4(0.2%)
	0.2	7446	394	7(0.3%)
	0.3	7746	440	9(0.4%)
	0.4	7996	494	11(0.5%)
	0.5	8094	560	13(0.6%)
	0.6	8497	611	15(0.7%)
	0.7	8550	660	22(1.1%)
	0.8	8550	678	22(1.1%)
0.15	0	6188	340	4(0.2%)
	0.1	6805	385	5(0.2%)
	0.2	7558	434	8(0.4%)
	0.3	7949	478	10(0.5%)
	0.4	8014	527	11(0.5%)
	0.5	8087	590	15(0.7%)
	0.6	8548	637	20(1%)
	0.7	8550	667	22(1.1%)
	0.8	8550	683	22(1.1%)
0.2	0	6376	380	5(0.2%)
	0.1	6963	421	6(0.3%)
	0.2	7665	468	10(0.5%)
	0.3	7919	509	11(0.5%)
	0.4	8067	550	11(0.5%)
	0.5	8487	614	19(0.9%)

Continued on next page

Table 24 – continued from previous page

Width	Toler.	ANGA	ANIA	#Imp. ND
	0.6	8544	645	20(1%)
	0.7	8550	670	22(1.1%)
	0.8	8550	684	23(1.1%)
0.25	0	6667	419	6(0.3%)
	0.1	7258	449	8(0.4%)
	0.2	7355	496	11(0.5%)
	0.3	7892	527	11(0.5%)
	0.4	8352	575	14(0.7%)
	0.5	8540	614	17(0.8%)
	0.6	8544	652	20(1%)
	0.7	8544	673	22(1.1%)
	0.8	8554	682	23(1.1%)
0.3	0	6763	443	8(0.4%)
	0.1	7178	472	9(0.4%)
	0.2	7410	512	11(0.5%)
	0.25	7943	529	11(0.5%)
	0.265	8158	534	11(0.5%)
	0.28	8160	540	11(0.5%)
	0.3	8160	546	12(0.6%)
	0.4	8447	583	16(0.8%)
	0.5	8520	613	17(0.8%)
	0.6	8544	655	21(1%)
	0.7	8544	670	21(1%)
	0.8	8548	682	23(1.1%)
0.35	0	7005	461	8(0.4%)
	0.1	7279	498	10(0.5%)
	0.2	7860	528	11(0.5%)
	0.3	8214	558	14(0.7%)
	0.4	8497	588	16(0.8%)
	0.5	8520	618	17(0.8%)
	0.6	8540	656	21(1%)
	0.7	8544	672	22(1.1%)
	0.8	8548	681	23(1.1%)
0.4	0	7268	482	9(0.4%)
	0.1	7582	512	10(0.5%)
	0.2	7882	544	13(0.6%)
	0.3	8214	566	15(0.7%)
	0.4	8429	596	16(0.8%)
	0.5	8500	620	17(0.8%)
	0.6	8520	655	21(1%)
	0.7	8547	673	22(1.1%)
	0.8	8547	681	23(1.1%)
0.45	0	7273	500	10(0.5%)

Continued on next page

Table 24 – continued from previous page

Width	Toler.	ANGA	ANIA	#Imp. ND
	0.1	7584	522	11(0.5%)
	0.2	7914	552	14(0.7%)
	0.3	8382	575	16(0.8%)
	0.4	8428	597	16(0.8%)
	0.5	8500	621	18(0.9%)
	0.6	8520	646	19(0.9%)
	0.7	8547	672	22(1.1%)
	0.8	8547	680	23(1.1%)
0.65	0.3	8085	587	16(0.8%)
	0.4	8399	609	17(0.8%)
	0.5	8432	629	18(0.9%)
	0.7	8504	654	20(1%)
	0.9	8547	683	23(1.1%)

C.2 Block size 250

Table 25: Complete version of Table 16.

Width	Toler.	ANGA	ANIA	#Imp. ND
0.05	0.6	7973	568	14(0.7%)
	0.7	8047	652	17(0.8%)
0.1	0	4040	212	2(0.1%)
	0.2	6351	309	4(0.2%)
	0.4	7195	458	10(0.5%)
	0.5	7654	547	15(0.7%)
	0.6	8042	635	17(0.8%)
0.3	0.4	7618	565	14(0.7%)
	0.5	7912	629	19(0.9%)
	0.6	8268	675	22(1.1%)
0.5	0.1	6864	472	10(0.5%)
	0.2	7536	510	12(0.6%)
	0.3	7633	544	13(0.6%)
	0.35	7906	572	15(0.7%)
	0.4	8693	601	19(0.9%)
	0.45	8745	623	21(1%)
	0.5	8892	642	22(1.1%)
0.7	0.2	7946	538	14(0.7%)
	0.4	8402	605	19(0.9%)
	0.5	8705	640	22(1.1%)
	0.6	8851	672	24(1.2%)

Continued on next page

Table 25 – continued from previous page

Width	Toler.	ANGA	ANIA	#Imp. ND
0.9	0.2	7978	558	15(0.7%)
	0.3	8377	579	16(0.8%)
	0.4	8392	604	17(0.8%)
	0.5	8402	641	22(1.1%)

C.3 Block size 100

Table 26: Complete version of Table 17.

Width	Toler.	ANGA	ANIA	#Imp. ND
0.05	0	560	104	0(0%)
	0.4	5160	299	4(0.2%)
	0.6	6755	582	16(0.8%)
	0.8	8043	935	35(1.7%)
0.15	0	2138	155	2(0.1%)
	0.1	3458	194	2(0.1%)
	0.2	5071	254	4(0.2%)
	0.3	5854	350	10(0.5%)
	0.4	6770	470	16(0.8%)
	0.5	7290	625	23(1.1%)
	0.6	8010	771	30(1.4%)
0.25	0.4	6990	576	22(1.1%)
	0.6	8323	830	31(1.5%)
0.3	0.5	7927	715	29(1.4%)
	0.7	8607	976	43(2.1%)
0.5	0.4	7866	649	25(1.2%)
	0.5	8212	741	30(1.4%)
0.7	0.4	8015	652	23(1.1%)
0.9	0.4	8063	656	21(1%)
1.2	0.4	7981	656	22(1.1%)