FREE-TEXT KEYSTROKE DYNAMICS DATA SET AND ALGORITHMS FOR CONTINUOUS AUTHENTICATION IN EDUCATIONAL PLATFORMS WITH MASSIVE OPEN ONLINE COURSES (MOOC)

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea Politehnica Timişoara
în domeniul Calculatoare și Tehnologia Informației
de către

Augustin-Cătălin IAPĂ

Conducător ştiinţific: prof.univ. emerit dr.ing Vladimir-Ioan CREȚU
Referenţi ştiinţifici: prof.univ.dr.ing. Dorian GORGAN
prof.univ.dr.ing. Viorel NEGRU
prof.univ.dr.ing. Nicolae ROBU

Ziua susţinerii tezei: 9 aprilie 2021

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică și Telecomunicații
8. Inginerie Industrială
9. Inginerie Mecanică
10. Știința Calculatoarelor
11. Știința și Ingineria Materialelor
12. Ingineria sistemelor
13. Inginerie energetică
14. Calculatoare și tehnologia informației
15. Ingineria materialelor
16. Inginerie și Management

Universitatea Politehnica Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul Şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
Tel./fax 0256 403823
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Prezenta lucrare a fost elaborată în cadrul Departamentului de Calculatoare și Tehnologia Informației al Universității Politehnica Timişoara.

Îi mulțumesc în mod deosebit, în primul rând, mentorului meu, domnului profesor emerit dr. ing. Vladimir-Ioan Crețu, conducătorul meu de doctorat, pentru reușita prezentei lucrări, pentru răbdarea cu care mi-a coordonat pașii pe tot parcursul perioadei de studii doctorale în scopul finalizării cu succes a acestui drum. Sfaturile pertinente și observațiile domnului profesor au fost întotdeauna acordate la momentele potrivite.

De asemenea, mulțumesc membrilor echipei de îndrumare: doamnei dr. ing. Diana Andone pentru ideea de a studia modul de identificare al utilizatorului în funcție de tipologia tastării, fiind un domeniu de interes și intens cercetat științific, doamnei prof. dr. ing. Carmen Holotescu pentru sfaturile oferite ori de câte ori le-am solicitat dar și pentru îndrumarea și încurajarea publicării primelor lucrări științifice, precum și domnului profesor emerit dr. ing. Ionel Jian pentru îndrumarea oferită.

Pe parcursul studiilor doctorale am avut și oportunitatea de a preda în calitate de student doctorand, cadru didactic asociat, studenților de la Facultatea de Automoatică și Calculatoare. Pentru oferirea acestei oportunități le mulțumesc în mod deosebit domului profesor dr. ing. Horia Ciocârlie, domnului profesor dr. ing. Vladimir-Ioan Crețu și domnului S.l. dr. ing. Sebastian Fuicu.

Le mulțumesc, de asemenea, tuturor cadrelor didactice de pe parcursul perioadei de licență și masterat de la Facultatea de Automatică și Calculatoare din cadrul Universității Politehnica Timișoara și în mod special domnului prof. dr. ing. Mircea Popa pentru coordonarea lucrărilor de licență și masterat. Le mulțumesc tuturor profesorilor care m-au format pe parcursul întregului meu parcurs școlar, de la Liceul Grigore Moisil din Timișoara și de la Școala Generală nr. 4 din orașul Moldova-Nouă, parcurs care a construit, cărămidă cu cărămidă, pentru a putea duce la bun sfârșit acest demers. Le mulțumesc în mod deosebit doamnei învățătoare Natalia Guran, doamnei profesoare Elena Bojici, care nu mai este, din păcate, printre noi și doamnei profesoare Adriana Simulescu.

Le mulțumesc tuturor celor 80 de voluntari care au răspuns pozitiv rugăminții de a furniza date în scopul prezentei cercetări, pentru timpul dedicat completării formularului conceput în acest sens.

Le mulțumesc în mod deosebit părinților pentru formarea oferită, pentru încurajarea constantă și pentru tot sprijinul dăruit cu sacrificii, de cele mai multe ori. Îi mulțumesc în mod deosebit și prietenei mele, Claudia, care m-a înțeles, m-a sprijinit și mi-a oferit timpul și liniștea necesare lucrului la teză.

Le mulțumesc colegilor care mi-au împărtășit din experiența lor de studenți doctoranzi: Norbert Kazamer, Alexandru Topârceanu, Iulia Știrb, Alexandru Filipovici, Ovidiu Sicoe, Sergiu Nimara, Renata Boar. Îi mulțumesc Krisztinei Verneș pentru ajutorul dat pentru revizuire. De asemenea aș vrea să mulțumesc tuturor care au fost alături de mine în toți acești ani și pe care nu i-am menționat anterior.

Sunt recunoscător lui Dumnezeu pentru răbdarea, puterea și sănătatea care m-au dus la finalizarea cu succes a tezei de doctorat.

Timişoara, februarie 2021 Augustin-Cătălin Iapă

Iapă, Augustin-Cătălin

Algoritmi și un set de date care folosesc dinamica tastării tastelor în cazul textului scris liber pentru autentificarea continuă în platforme educaționale care cuprind cursuri online deschise masive (MOOC)

Teze de doctorat ale UPT, Seria X, Nr. YY, Editura Politehnica, 2021, 169 pagini, 74 figuri, 26 tabele.

Cuvinte cheie: dinamica apăsării tastelor, algoritm de autentificare, identificarea utilizatorului, modul de tastare, metrici pentru calculul distanțelor

Rezumat:
Prezenta lucrare se focusează pe autentificarea continuă a utilizatorului unui calculator pe baza modului de a tasta la tastatură. În cadrul cercetării s-a dezvoltat un algoritm de autentificare pe baza modului de tastare, s-a colectat un set de date referitoare la modul de tastare de la 80 de voluntari, s-au propus două metrici modificate pentru a se obține performanțe mai bune ale algoritmului de autentificare și s-a propus o structură de date pentru a stoca informațiile necesare ale utilizatorilor.
Această metodă de autentificare își justifică atenția mai ales în cadrul platformelor educaționale online, platforme care au cunoscut o creștere foarte mare în anul 2020, datorită mutării majorității cursurilor în mediul online, restricție generată de criza COVID-19.

Free-text keystroke dynamics data set and algorithms for continuous authentication in educational platforms with massive open online courses (MOOC)

Keywords: keystroke dynamics, authentication algorithm, user identification, typing pattern, distance metrics, di-graphs

Abstract:
This paper focuses on the continuous authentication of a computer user based on keystroke dynamics, the way to type on the keyboard. During the research, an authentication algorithm based on keystroke dynamics was developed, a data set regarding the typing mode was collected from 80 volunteers, two modified metrics were proposed to obtain better performances of the authentication algorithm and a data structure was proposed to store the necessary information of the users.
This method of authentication justifies its attention especially in online educational platforms, platforms that experienced a very large increase in 2020, due to the relocation of most courses in the online environment, a restriction generated by the COVID-19 crisis.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF CODES

ABBREVIATIONS

ACM	Association for Computing Machinery
CCK08	Connectivism and Connective Knowledge 2008
DD	Down-Down time
DU	Down-Up time
EER	Equal Error Rate
FAR	False Acceptance Rate
FN	False negatives
FP	False positives
FRR	False Rejection Rate
HCI	Human computer interaction
MOOC	Massive Open Online Course
ROC curve	Receiver Operating Characteristic curve
SVM	Support-vector machine
TAR	True Acceptance Rate
TN	True negatives
TP	True positives
TRR	True Rejection Rate
UD	Up-Down time
UU	Up-Up time
ZMFAR	Zero Miss False Acceptance Rate

1 INTRODUCTION

1.1 Thesis context

This thesis started from the need to develop additional ways to identify the identity of a user who uses a private account on a computer. This need is more pronounced in the case of courses or exams that take place online. The MOOC phenomenon (Massive Open Online Courses), courses attended by a large number of students from any corner of the world online, was born in 2008. This phenomenon reached a first maximum in 2012, and in 2020 there was an exponential increase in the number of students enrolled [IAP21b].

The year 2020 also led to radical changes in education systems as an outcome of the health crisis caused by the SARS-CoV-2 virus. This has resulted in an unprecedented push to online learning. Universities, primary schools or high schools have been pressed to adapt and move the entire classical education system from studying in the classroom, face to face, to distance platforms. In this context, it has become much more important to find methods to ensure that, for instance, during an exam, where both the teachers and students are in different locations, to ensure that the student, through easily accessible means, is the one who solves the subjects and receives a grade based on his knowledge and performance [IAP21b].

There are many ways and possibilities to identify and authenticate a user from an electronic account. The most common method is to retain a username and its password and based on these two, the user has access to the account. The use of physical cards, such as those used by banks, or fingerprints, retinal scanning or face recognition requires the existence of additional devices for retrieving data from users. For authentication during an exam, it is not enough to have an account and a password, in case the student wants to speculate by leaving someone else in his place to solve the subjects. In most cases, the camera and microphone must be turned on throughout the exam [IAP21b].

An effective method in solving the problem described above is continuous authentication using keystroke dynamics. Keystroke dynamics is the method by which a user can be identified or authenticated based on his or her particular way of typing text on the keyboard. This method does not require additional hardware, any computer or laptop that is equipped with a keyboard is accepted. Additionally, another advantage is represented by the fact that the identity verification can be done continuously, at any time when the user types on the keyboard. The password authentication cannot be done the same way presented before, being done usually only once when accessing the account, and along the way the user can change without the system to realize the change.

Another advantage of using identification or authentication using keystroke dynamics is that the user does not have to take additional steps. The participants just have to type and the system monitors the way of their typing. In this case, after an

authentication in a system, if the user changes, the system will realize that someone else is at the computer and can signal this change.

Thousands of students can participate in MOOC courses at the same time. In the case of an exam with thousands of students, it becomes impossible to supervise through the video camera and the microphone, this method being effective when the number of students is reduced. In the case of keystroke dynamics, any number of students can be continuously authenticated, there is no such limitation in this regard.

The disadvantage of a system with authentication or identification of users through the keystroke dynamics method is the accuracy of the algorithm with which the user can be identified. Currently, systems that use this method do not reach error rates of 0%. They have performance that identifies the user with an error rate of less than 10%, or in some cases with even higher accuracy, instead improving algorithms based on keystroke dynamics is still a challenge in this area. Along these, another challenge for scientific research in this field is the fact that in order to test the efficiency of the algorithms proposed in various researches, databases are needed that capture the typing mode, thus better simulating the real conditions.

Within the scientific research made about the keystroke dynamics they were identified two different branches. The first would be when a user types a default text on the keyboard, such as a user, a standard password or phrase. The second one would be the typing of a free text on the keyboard without certain conditions being imposed [UMP85][MES11]. The two methods are analyzed separately by different methods in the scientific literature on this subject. Both, however, involve a phase in which the system collects data about the user, the typing times, and the typing mode, thus, creating a profile of the user that he will use later in the continuous authentication phase. The first method has been more intensively explored and the results are more successful in this direction because it is the same text entered from the keyboard each time. The second method, when the user types a free text with the help of the keyboard, without conditions, has been researched especially in recent years, and the results are increasingly improved.

Only in the last 5 years over 10,000 scientific papers have been published about keystroke dynamics. Also, survey papers have been published as keystroke dynamics biometrics has drawn intense research interest the past couple of decades [ZHO15]. In Table 1.1 is the number of scientific papers in the field of "keystroke dynamics" and also in the field of "free text keystroke dynamics". The graphic represented in Figure 1.1 illustrates the growing interest in the field of "keystroke dynamics" and also in the field of "free text keystroke dynamics" [IAP21b].

Table 1.1 Number of scientific publication in the field [IAP21b]

Interval	„keystroke dynamics"	„free text keystroke dynamics"
1981-1985	224	108
1986-1990	643	277
1991-1995	1.080	566
1996-2000	1.630	863
2001-2005	2.950	1500
2006-2010	4.940	2520
2011-2015	7.890	4020
2016-2020	10.100	4880

The number of scientific publications in this field was counted by searching for the two text sequences on scholar.google.com, filtered on 5-year intervals, on the

following time intervals: 1981-1985, 1986-1990, 1991-1995, 1996 -2000, 2001-2005, 2006-2010, 2011-2015 and 2016-2020 [IAP21b].

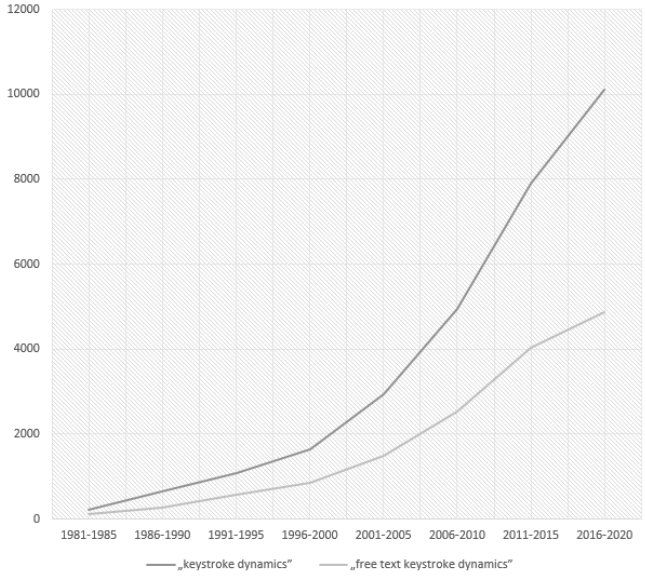


Figure 1.1 Evolution of publication about "keystroke dynamics" and "free text keystroke dynamics" from 1981 till 2020

It is observed that in the last 5 years over 10,000 scientific papers have been published with the topic "keystroke dynamics", and scientific papers that have addressed the branch "free text keystroke dynamics" represent about half of these, reaching about 5,000 papers published in the last 5 years [IAP21b]. In the Figure 1.2 is also a hierarchy chart with the volume of publication about "keystroke dynamics" on the following time intervals: 1981-1985, 1986-1990, 1991-1995, 1996 -2000, 2001-2005, 2006-2010, 2011-2015 and 2016-2020.

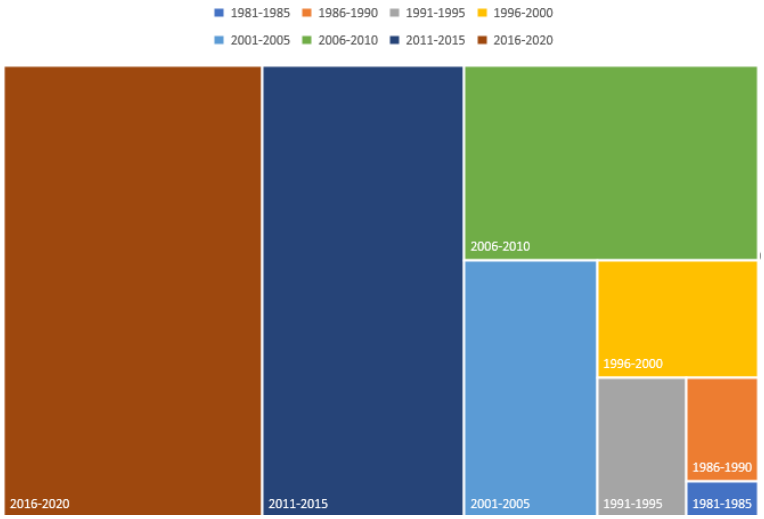


Figure 1.2 Hyerarchy chart with the volume of publication about "keystroke dynamics"

According to ACM Computing Classification System [ACM12], the Figure 1.3 and the Table 1.2 list categories and concepts of the computing discipline used in this work.

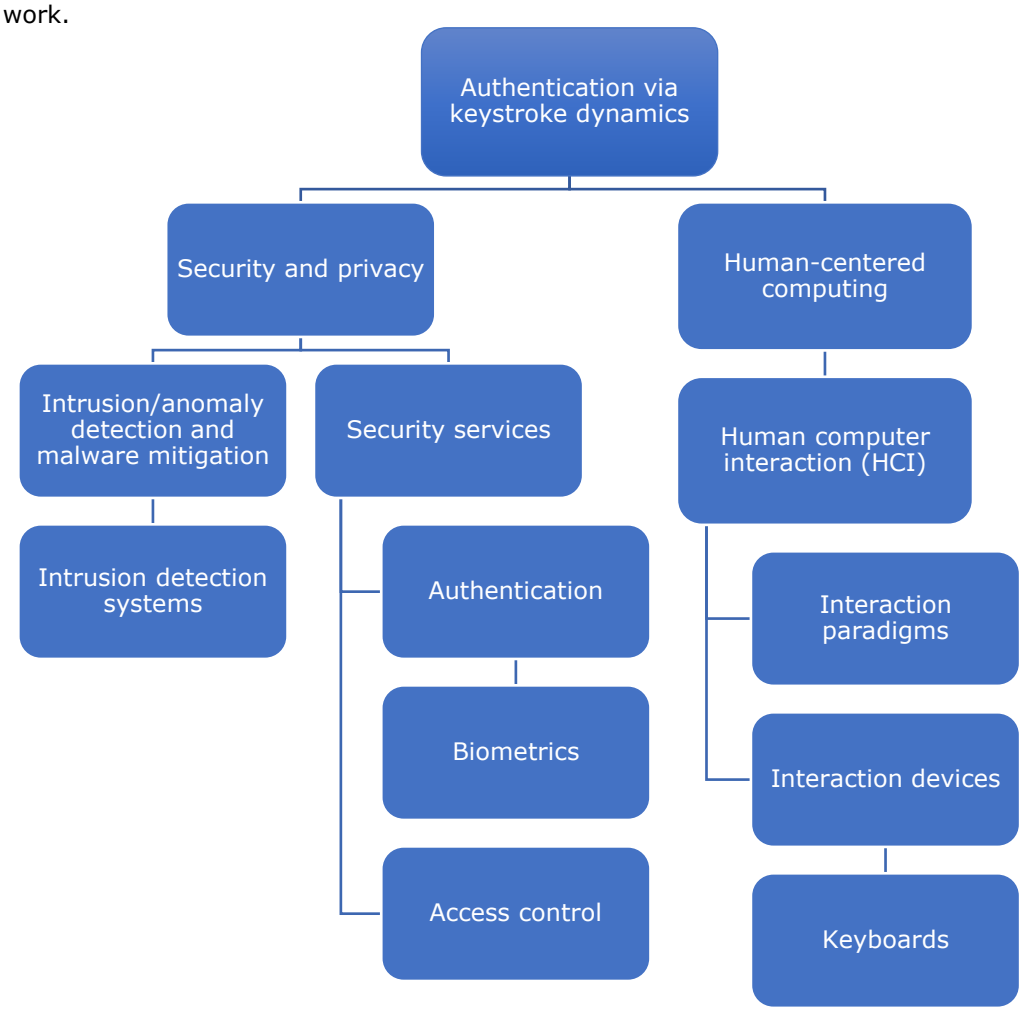


Figure 1.3 Categories and concepts of the computing discipline used in this work according ACM Computing Classification System [ACM12]

Table 1.2 Categories and concepts of the computing discipline used in this work generated whit ACM tool [ACM12]

Security and privacy ~ Security services ~ **Authentication ~ Biometrics;**
Security and privacy ~ Security services ~ **Access control;**
Security and privacy ~ Intrusion/anomaly detection and malware mitigation ~ **Intrusion detection systems;**
Human-centered computing ~ Human computer interaction (HCI) ~ **Interaction paradigms;**
Human-centered computing ~ Human computer interaction (HCI) ~ **Interaction devices ~ Keyboards;**

1.2 Thesis objectives

In this research project, the author set the following four objectives:

Objective 1, O1, The first objective of this thesis is to collect a database with the test pattern from at least 80 users, in order to test the authentication algorithm for this research, but also to make it available to other interested researchers.

Objective 2, O2, The second objective of this thesis is to implement an algorithm for authenticating the users of a computer based on the keystroke dynamics, the keyboard typing mode.

Objective 3, O3, The third objective of this thesis is to propose at least two new metrics for calculating the distances between two vectors that generate better performance compared to the Equal Error Rate (EER) performance indicator than the classical methods.

Objective 4, O4, The fourth objective of this thesis is to propose a data structure as efficient as possible, which should contain the most relevant information about the typing of a user.

1.3 Thesis structure

The thesis is organized as follows:

• Chapter 1 presents the thesis context, the thesis objectives and the thesis structure. Subchapter 1.1 Thesis context briefly presents the starting point of the research, the evolution of research in the field of keystroke dynamics and categories and concepts of the computing discipline used in this work. Subchapter 1.2 Thesis objectives presents the four objectives proposed in this paper and subchapter 1.3 Thesis structure briefly presents each chapter of the thesis.

• Chapter 2 presents the state-of-the-art of the field to which this work is addressed. The first part, subchapter 2.1 Evolution of educational systems, analyses the evolution of educational systems and the most important platforms worldwide with MOOC (Massive Open Online Courses) and their evolution from its inception until now (2020), the year in which online education has grown exponentially. This analysis shows the importance and dimension of the field of e-Learning has reached and justifies the scientific research of this paper. Subchapter 2.2.1, A biometric feature: keystroke dynamics, gives an overview of Biometrics and how a person can be identified based on it, thus, the purpose being the investigation of continuous authentication based on free-text keystroke dynamics. Keystroke dynamics is a biometric. After going through the evolution of e-Learning and Biometrics systems, in the subchapter 2.2 Keystroke dynamics – literature review, a detailed research is conducted on keystroke dynamics. It analyses the types of keystroke dynamics, in the subchapters 2.2.3 and 2.2.4: fixed text and free-text, also the methodologies applied are analyzed in other researches to collect data and to evaluate the collected data, in the subchapters 2.2.9, 2.2.10 and 2.2.11. It covers, in the subchapters 2.2.5, 2.2.6, 2.2.7 and 2.2.8, the technical characteristics that are considered within the existing algorithms, but also, in the subchapter 2.2.13 Evaluating the performance of authentication algorithms based on keystroke dynamics, the characteristics that measure the performances of the algorithms in this field.

• Chapter 3 presents the research methodology applied in this research project. The steps performed in the present scientific research are presented below: A. Development of the platform for the acquisition of input data, B. Acquisition and

initial processing of input data from 80 volunteers (how typing on their keyboard), C. Processing the input data so as to generate a user pattern for each user, D. Development of an algorithm in the C programming language for calculating distances used in keystroke dynamics authentication, E. Simulation of system authentication by genuine users or impostors to measure the performance of the developed algorithm. Each of these five steps are detailed in this chapter, in five subchapters 3.1, 3.2, 3.3, 3.4 and 3.5.

• Chapter 4 is about the data set collected for the present research. The first subchapter 4.1, Platform for collecting data about keyboard typing from 80 volunteers, presents the platform for collecting data about keyboard typing and how data was collected from 80 users in order to later develop an algorithm. Moreover, after presenting the platform with the help of which data were collected from users, it will proceed to the analysis of these collected data and to the presentation of the particular characteristics, in subchapters 4.2 Analysis of time and key events collected from users, 4.3 Acquisition and initial processing of input data from 80 volunteers (how typing on the keyboard) and 4.4 Analysis of keys collected from users. It shows how they were processed using an algorithm written in the C programming language. The subchapter 4.5, Processing the input data so as to generate a user pattern for each user, presents the structures that store user typing data. The subchapter 4.6, Keys distribution analysis, presents the analysis of the collected keys. The subchapter 4.7, Differences between users, graphically displays the typing pattern for different users. This chapter addresses the validation of O1 from the first chapter of this thesis.

• Chapter 5 In this chapter it is presented the authentication algorithm based on free-text keystroke dynamics. First of all, the algorithm developed for processing the data obtained from the users is presented. The algorithm simulates user authentication based on keystroke dynamics and measures the obtained performances. In the chapter it is presented the architecture of the algorithm in the subchapter 5.1 The architecture of the authentication algorithm and the structure of the algorithm in the subchapter 5.2 The structure of the authentication algorithm. The development of this algorithm is established by O2 from the first chapter of this thesis.

• Chapter 6 In this chapter it is presented a series of experiments performed to measure the performance of the written algorithm for the purpose of this research and to analyze the results obtained. Gradually, experiments with the keystroke time of a single key, in the subchapter 6.1, and experiments with di-graphs, in the subchapter 6.2, are presented. Both in the analysis of the characteristics with a single key and with a di-graph, the degree of Equal Error Rate (EER) is calculated in order to appreciate the performances of the algorithms. The results are presented in the case of experiments using Euclidean distance (in the subchapters 6.1.1 and 6.2.3), Manhattan distance (in the subchapters 6.1.2 and 6.2.4), R distance (in the subchapter 6.1.3) and A distance (in the subchapters 6.1.4 and 6.2.5). The chapter also investigates, in the subchapter 6.1.5 The sample size, the differences in performance if the pattern is built for each user with various sample sizes, starting from 200 key events / pattern and up to 3000 key events / pattern. At the end of the chapter, following all the experiments performed and presented, the author proposes, in the subchapter 6.4, Proposing new metrics for calculating distances between users, the modification of two metrics obtaining new metrics for calculating the distances between two vectors that have higher performances than the classical calculation methods. For the two new metrics, the performances obtained in terms of Equal Error Rate (EER) are presented. By proposing these metrics, O3 is validated. It also proposes, in the subchapter 6.5, Proposed user pattern, a structure for retaining a user's pattern, a structure that takes up small memory and requires little time to

perform all the necessary calculations in the algorithms. By proposing the user pattern, O4 is validated. In the end of the Chapter, in the subchapter 6.6 Comparison of the related works, the performances obtained in the present research are compared with those obtained by other authors in their researches.

- Chapter 7 summarizes the conclusions drawn from the previous chapters and future research directions in this field, starting from the results presented in this paper. The author's own contributions to the field of keystroke dynamics are presented in the subchapter 7.1.1 The personal contribution: the proposal of two new metrics for calculating the distance between two vectors in order to allow the approximation of the degree of similarity between two patterns from two different users or from the same user. Also, the data collected from the 80 users about how to type on the keyboard is a contribution to the advantage of future researches because they will be available to all researchers interested in conducting investigation in the field. Another own contribution is the proposal of a pattern in order to retain the minimum necessary data about a user so to obtain performances in the continuous authentication. The last part of this chapter, the subchapter 7.2, Future works, presents the future research directions. The field still needs to be exploited, and future research directions may bring higher performance than those currently obtained.

In the present thesis are taken elements (conclusions, experiments, results, passages, formulas, images, graphics, phrases, etc.) from works written before the final writing of the thesis by the author. Papers are published, presented or submitted for publication. The list of papers of the author is at the end of the thesis, in the Chapter SCIENTIFIC ACTIVITY.

2 STATE-OF-THE-ART

The thesis context, its' objectives and its' structure were presented in the previous chapter, Chapter 1 Introduction.

Subsequently, this Chapter presents the state-of-the-art of the field to which this work is addressed. The first part, subchapter 2.1 Evolution of educational systems, analyses the evolution of educational systems and the most important platforms worldwide with MOOC (Massive Open Online Courses) and their evolution from its inception until now (2020), the year in which online education has grown exponentially. This analysis shows the importance and dimension of the field of e-Learning has reached and justifies the scientific research of this paper. Subchapter 2.2.1, A biometric feature: keystroke dynamics, gives an overview of Biometrics and how a person can be identified based on it, thus, the purpose being the investigation of continuous authentication based on free-text keystroke dynamics. Keystroke dynamics is a biometric. After going through the evolution of e-Learning and Biometrics systems, in the subchapter 2.2 Keystroke dynamics – literature review, a detailed research is conducted on keystroke dynamics. It analyses the types of keystroke dynamics, in the subchapters 2.2.3 and 2.2.4: fixed text and free-text, also the methodologies applied are analyzed in other researches to collect data and to evaluate the collected data, in the subchapters 2.2.9, 2.2.10 and 2.2.11. It covers, in the subchapters 2.2.5, 2.2.6, 2.2.7 and 2.2.8, the technical characteristics that are considered within the existing algorithms, but also, in the subchapter 2.2.13 Evaluating the performance of authentication algorithms based on keystroke dynamics, the characteristics that measure the performances of the algorithms in this field.

2.1 Evolution of educational systems

In this subchapter the author presents the evolution of MOOC (Massive Open Online Courses) platforms. In 2020, in the Coursera Platform are involved nearly 69 million learners [VAN20]. The number of Massive Open Online Courses increased in the last years.

Debates about future and evolution of eLearning and MOOC (Massive Open Online Courses) were in the last few years. In this chapter the author makes an introspection in evolution of Massive Open Online Courses with a comparison of the most important platforms of MOOC. Also, in the last years, researchers have paid attention to Learning Analytics field [IVA16]. We have more and more data from Learning Management Systems. There were noticeable additional challenges regarding the field of education in 2020. With the COVID-19 pandemic the authorities have not only introduced restrictions on the movement of citizens, but have also tightened the preventive measures implementing new regulations with reference to education. A decisive number of universities have had to adapt to the unfamiliar circumstances, moving all their activities to the online environment. These limitations

have led to an unprecedented leap in online education. Suddenly, both teachers and students or pupils, were forced by the newly implemented conditions to move their entire activity to online educational platforms and thus continue their courses in this manner. This process has led to a development of the e-learning section, helping the growth of companies that are being active in this field and has forced those who have not used these systems so far to learn them in a very quick way [IAP14a].

The educational system has continually evolved due to technological innovations. In [DAN12] the author made an enumeration of innovations: In 1841 the blackboard, in 1940 the motion picture, in 1957 the television. Programmed learning and computers were another invention which contributed on education evolution. Internet and communication technologies could develop the format of education [IAP14a].

The MOOC evolution starts with the "Connectivism and Connective Knowledge" – CCK08 course in 2008 which had a large number of online participants. The course was facilitated by Downes and Siemens [DOW14] [IAP14a].

The MOOC starts in the 2008 but the year 2012 was declared the MOOC year. The next years after 2012 was good years for MOOC, with millions of learners and hundreds of partners involved to develop courses [IAP14a].

A record number of users turned to online learning in 2020. Since March, there were more than 69 million enrollments only on Coursera. About 430% increase compared to the same period last year [VAN20] [IAP21b].

2.1.1 Platforms of MOOC

The most important platforms with massive open online courses and with the largest number of users and partners are Coursera and edX. The number of MOOC platforms is increasing. In December 2020 Coursera platform had 69 million users and edX platform had 24 million learners. Coursera is a for profit platform. Coursera platform can be accessed online at www.coursera.org and it has started on April 2012 [COU20]. EdX platform can be accessed at www.edX.org. EdX has started on December 2011 and is a non-profit platform [EDX20][MAT20][IAP21b].

In the Figure 2.1 is represented the evolution of the number of courses on the Coursera and edX platforms in 2014 - 2020. Coursera had 622 courses in February 2014, 761 courses in September 2014, 1557 courses in December 2015, 2000 courses in February 2017 and 3900 courses in December 2020. EdX had 151 courses in February 2014, 287 courses in September 2014, 814 courses in December 2015, 1283 courses in February 2017 and 3000 in December 2020 [IAP14b][IAP21b].

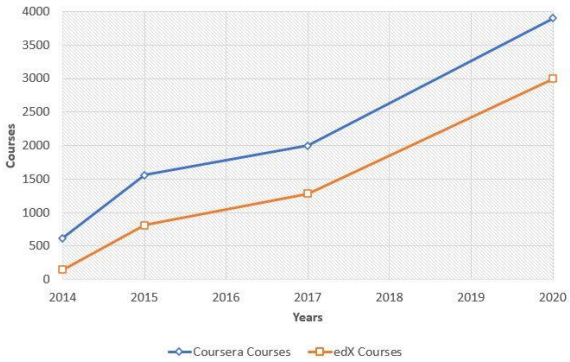


Figure 2.1 Evolution of courses on Coursera and edX platforms [IAP21b]

In the Figure 2.2 is represented the evolution of the number of partners on the Coursera and edX platforms. Coursera had 108 partners in February 2014, 114 partners in September 2014, 140 partners in December 2015, 149 partners in February 2017 and 227 partners in December 2020. EdX had 32 partners in February 2014, 53 partners in September 2014, 90 partners in December 2015, 94 partners in February 2017 and 145 partners in December 2020 [IAP14a][IAP21b].

Figure 2.2 Evolution of the number of partners on Coursera and edX platforms [IAP21b]

2.1.2 Coursera Platform

Coursera is the largest MOOCs platform in the world to this day. At the end of 2020, Coursera reached over 3,900 courses and specialization. Users can choose from this varied offer of courses in different fields. More than 20 Degrees and MasterTrack Certificates and over 13 Professional Certificates. This information can be checked periodically on their own website. The Figure 2.3 shows these numbers that define the platform at the end of 2020. Coursera started offering courses to users in 2012. „Coursera was founded by Daphne Koller and Andrew Ng in 2012 with a vision of providing life-transforming learning experiences to learners around the world. Today, Coursera is a global online learning platform that offers anyone, anywhere, access to online courses and degrees from leading universities and companies.” [COU20] [IAP21b]

Figure 2.3 Number of courses, specializations, degrees, MasterTrack and profesionals certificates on Coursera in 2020 [COU20]

On Coursera, the users can choose from 6 learning programs [COU20], detailed in Figure 2.4 : Guided project, Course, Specialization, Professional certificate, Mastertrack™ certificate, Degree [IAP21b].

LEARNING PROGRAM	DESCRIPTION
GUIDED PROJECT	Learn a job-relevant skill that you can use today in under 2 hours through an interactive experience guided by a subject matter expert. Access everything you need right in your browser and complete your project confidently with step-by-step instructions.
COURSE	Take courses from the world's best instructors and universities. Courses include recorded auto-graded and peer-reviewed assignments, video lectures, and community discussion forums. When you complete a course, you'll be eligible to receive a shareable electronic Course Certificate for a small fee.
SPECIALIZATION	Enroll in a Specialization to master a specific career skill. You'll complete a series of rigorous courses, tackle hands-on projects, and earn a Specialization Certificate to share with your professional network and potential employers.
PROFESSIONAL CERTIFICATE	Whether you're looking to start a new career or change your current one, Professional Certificates on Coursera help you become job ready. Learn at your own pace from top companies and universities, apply your new skills to hands-on projects that showcase your expertise to potential employers, and earn a career credential to kickstart your new career.
MASTERTRACK™ CERTIFICATE	With MasterTrack™ Certificates, portions of Master's programs have been split into online modules, so you can earn a high quality university-issued career credential at a breakthrough price in a flexible, interactive format. Benefit from a deeply engaging learning experience with real-world projects and live, expert instruction. If you are accepted to the full Master's program, your MasterTrack coursework counts towards your degree.
DEGREE	Transform your resume with a degree from a top university for a breakthrough price. Our modular degree learning experience gives you the ability to study online anytime and earn credit as you complete your course assignments. You'll receive the same credential as students who attend class on campus. Coursera degrees cost much less than comparable on-campus programs.

Figure 2.4 What Coursera has to offer [COU20]

In 2020, most popular courses worldwide on Coursera Platform was [VAN20]:

”1. The Science of Well-Being from the Yale University

2. COVID-19 Contact Tracing from the Johns Hopkins University

3. Programming for Everybody (Getting Started with Python) from the University of Michigan

4. Machine Learning from the Stanford University

5. Learning How to Learn: Powerful mental tools to help you master tough subjects from the McMaster University UC San Diego

6. English for Career Development from the University of Pennsylvania

7. Financial Markets from the Yale University

8. First Step Korean from the Yonsei University

9. Introduction to Psychology from the Yale University

10. Write Professional Emails in English from the Georgia Institute of Technology” [IAP21b]

2.1.3 edX Platform

In 2012, Harvard and MIT came together with the idea to create edX, a nonprofit online learning platform to reimagine education as we knew it. In 2012, MIT offered its first massive open online course (MOOC), Circuits and Electronics. [IMP20] [IAP21b]

EdX mission is focused on three pillars [IMP20] :

”1. Expanding access to high quality education to everyone, everywhere

2. Reimagining education both on-campus and online

3. Improving teaching and learning outcomes through research”

For the first time ever, in 2015, learners earned college credit for MOOCs on edX [IMP20]. In 2020, the edX platform has reached over 24 million unique users. edX had over 5700 instructors for more than 3000 courses from 145 partners. The number of countries from which the users came is 196. More than 1.6 million certificates have been issued [IMP20] [IAP21b]. The Figure 2.5 shows these numbers graphically.

Figure 2.5 Numbers from edX platform 2020 [IMP20]

Some of the top Universities on edX platform are showed in the Figure 2.6

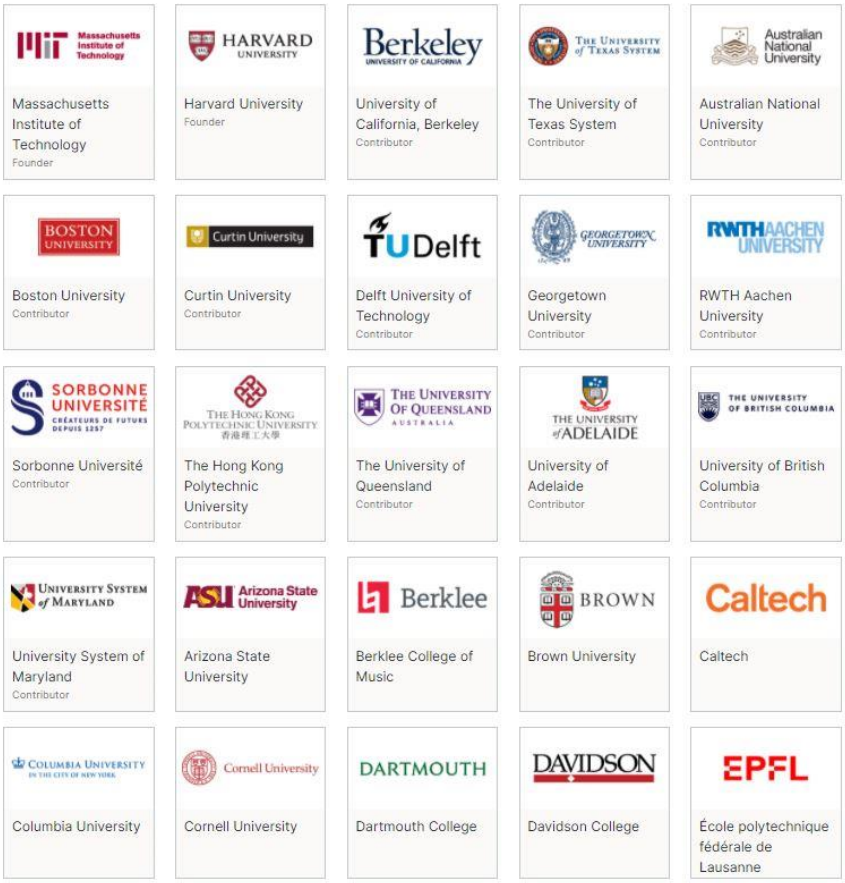


Figure 2.6 Some of the top Universities on edX Platform [EDX20]

2.2 Keystroke dynamics – literature review

There were noticeable additional challenges regarding the field of education in 2020. With the COVID-19 pandemic the authorities have not only introduced restrictions on the movement of citizens, but have also tightened the preventive measures implementing new regulations with reference to education. A decisive number of universities have had to adapt to the unfamiliar circumstances, moving all their activities to the online environment [IAP21b].

These limitations have led to an unprecedented leap in online education. Suddenly, both teachers and students or pupils, were forced by the newly implemented conditions to move their entire activity to online educational platforms and thus continue their courses in this manner. This process has led to a development of the e-learning section, helping the growth of companies that are being active in this field and has forced those who have not used these systems so far to learn them in a very quick way [IAP21b].

This unforeseen change brought a series of challenges such as: how to adapt the classic courses to the new context that imposes restrictions and teach them online, how long a course should be, how to pay attention to the students, how to interact with each of the participants in this structured system of organizational and didactic measures, how to evaluate them, what criteria will be used for grading, or how to supervise each one during an exam session, considering that each one is physically in a different place [IAP21b].

The challenges and advantages or disadvantages of conducting exams during the crisis was a major issue worthy of study. To be even more specific, on an issue that most teachers have lifted, namely the way you check whether a student is cheating or no during the exam, if he will write the exam and will not let anyone else to do it for him [IAP21b].

Most evaluation systems in this regard have been done through grid-type exams, with limited time, or through oral assessments through online platforms, or even through tests in which students are constrained to sit with the camera and microphone turned on so that the teacher can monitor them remotely [IAP21b].

To emphasize one prominent method that partially solves the problem of identifying the students that are attending the exams, the paper focused on the method of continuous authentication with the help of the unique way of pressing the keys while writing the subject of the exam (keystroke dynamics). Coursera, one of the biggest platforms of Massive Open Online Courses, used keystroke dynamics to verify online users from courses [MAA14] [IAP21b].

The method of continuous authentication using keystroke dynamics has been exhaustively researched lately. This practice has several fields in which it can be successfully applied, for example, as an additional security method when a user accesses his bank account on the internet or when making a payment in a similar way [BAN12] [IAP21a]. It can be applied for e-mail accounts, or any other online platform that requires a lot of typing. The authentication process can be categorized by the number of incorporated factors: something you know like a username and a password, something you have, like card, token or something you are, like biometrics. [BUR06] A combination of these processes is a strong authentication [BAN12] [IAP21a].

Two-factor authentication is a large scale used approach, in some systems even mandatory, for online services [KAN14]. The traditional password is the first factor and the second factor can be a SMS access code or a PIN generated randomly at the time of authentication [DAS16]. The keystroke dynamics can also be the second factor authentication [IAP21a].

Up to 28 muscles are used during a keystroke [KOC19]. The keystroke dynamics technique consists in capturing and analyzing the typing mode of a user. More precisely, the pressing time on one key, but also the time between the pressing of two consecutive keys. The rhythm along the pressure of keys plays an important role when it comes to study the cases [TSA14]. These features are unique, as are other methods of identifying individuals such as fingerprint, facial recognition, account password, or the use of a physical card or other physical identification device [IAP21b].

Keystroke dynamics has been pointed out as a practical behavioral biometric feature that does not require any additional device for scale up user identification or authentication [PIL15]. In the next subchapter a short presentation about biometrics, classification and keystroke dynamics like a biometric are made [IAP21b].

2.2.1 A biometric feature: keystroke dynamics

With technology used in online educational settings, cheating is easier than in traditional settings [JAY19] [FAB97] [IAP21b].

The physiological features include fingerprints, face, eye – iris patterns or retina patterns, palm topology, hand geometry, wrist veins and thermal images. The behavioral features include handwritten signatures, voiceprints and keystroke dynamics [POL00][BER02]. Keystroke dynamics, the behavioral biometric, is a method that can secure the cyberspace [HUA17][BAN12] [IAP21b].

Biometric features are unique to each user and they cannot be lost or stolen [JOY90]. The same physiological factors that give uniqueness to a signature made on a sheet are found in the case of keystroke dynamics [CAL19] [IAP21b].

Hard biometric requires additional hardware that costs and decreases the availability of users to use it. This barrier does not exist in the case of keystroke dynamics. [CHA20] [IAP21b]

To implement authentication systems, physiological characteristics are more successful than behavioral characteristics. Physiological characteristics do not vary over time, while behavioral characteristics can change quite a bit over time. Keyboard analysis can be done without the help of special tools, the classic computer keyboard is enough [BER02][IAP21a].

Behavioral biometrics measuring human actions. Behavioral traits such as handwriting, signatures, keystroke dynamics, and mouse dynamics can be used to identify users. They are less costly, less accurate than physiological characteristics, as they often change slightly depending on circumstances [ALI17][JAY19].

For institutions of higher education, "typing signature" is the most cost-effective and reasonable approach to improve online assessment security [JAY19][FAB97][IAP21a].

2.2.2 The route of scientific research and branches of the field

Keystroke dynamics is an research field with more and more importance in network access control and cyber security [ZHO12] [IAP21b]. For now, only a few studies are about free-text keystroke dynamics, the way that the users type what text the user wants. Most of them are analyzed only fixed text, static text [ZHO12][SAL10][ZAC10]. Fixed content and fixed length data are usernames or passwords [MON02]. Free text requires two phases: the user enrollment phase in the system and the user verification phase [MON02].

First, the use for users identification was researched in the 1970`s [ZHO12]. Spillane wrote his conclusions about the first investigation in 1975 [FOR77] and Forsen, Nelson and Staron in 1977 [SPI75]. 'Fist of the Sender' was a methodology in World War II that was used to identify, by using the rhythm, the sender of the telegraph. [BAN12] [VAC07] [DUN08][IAP21a].

Keystroke dynamics have been studied mostly in connection to authentication, but some studies, such as [MES11], have also studied the detection of emotional states of the user who uses the keyboard. Other studies focus on predict users age and gender from unintentional traces, that left behind by use of keyboard and mouse [AVA17]. In [SAL18], the authors explored the relevance of individual and general keyboard and mouse interaction patterns and they had modeled user`s keystroke dynamics and mouse movements with data mining techniques to detect the emotion of users in real-world learning scenarios [IAP21a].

In [LIM14], the authors indicates that automatic analysis of human stress from mouse input and keyboard input is potentially useful for providing adaptation in e-learning systems [IAP21a].

Typing behavior for continuous authentication is a biometric modality proposed in [ROT14]. The authors collected a video database from 63 users with static text and free text typing and developed computer vision algorithms to extract hand movement from the video stream.

If most studies use only data retrieved from the keyboard, there are studies that use a mixed method of user identification, based on data retrieved from the keyboard, but also on data retrieved from the mouse [LOZ17]. Additional features, like pressure, are used in addition to time-based features, but to capture this data you need touch screens or other special devices [TEH13]. The stages that a research in this field goes through are: extracting the keyboard features, creating user profiles and updating them and identifying the efficiency criteria [KOC19][IAP21a].

Most studies analyze data collected in English. There are studies that research the field for texts in other languages, such as French [BOU17], Italian [SOL11], Japanese [SAM09], Russian [KOC19], Arabic [ALS16], Korean [JUN20] or others.

Commercial keystroke dynamic products exist. In 2003, the paper [ILO03] presents the company BioNet Systems which patented the BioPassword authentication system [ZIL98]. In Romania, Typing DNA is a company, a start-up, that received funds of 6.2 million euros in 2020 to create a typing identity for security [STE20].

Other studies, like [ARW17], incorporates the use of nonconventional typing features using free text typing dynamics. Semi-timing features along with the editing features were extracted from the users' typing flow and decision trees were used to classify each of the user data.

Algorithms of dynamic authentication can be divided into three major groups: estimation of metric distances, statistical methods and machine learning. Methods of keyboard recognition used in the literature are: distance, neural networks, statistical, probabilistic, machine learning, clustering, decision tree, evolutionary computing, fuzzy logic or other [KOC19] [IAP21a].

Some limitations of keystroke dynamics previous research were: it took a long time to train the model, data were manual preprocessed by human or large database was required [YUE04]. The authors from [YUE04] conclude that use of keystroke dynamics can make a more secure system.

The following sections of this chapter present technical aspects from the literature in the field of keystroke dynamics.

2.2.3 Fixed text keystroke dynamics

Fixed text keystroke dynamics is applied to the exactly same text typing, both in the user data retrieval phase and in the user identification or verification phase. Being the same text, with the same sequences it is much easier to analyze how it is typed. For example, when a user enters their username and password it is always the same text sequence. In this case you can analyze the similarities or the differences with greater accuracy, remaining at the same typing mode. Difference occurs if every time there can be another text typed from the keyboard, as is the case with free text keystroke dynamics [IAP21a].

Coursera, one of the biggest platforms of Massive Open Online Courses, used keystroke dynamics to verify online users from courses [MAA14].

2.2.4 Free text keystroke dynamics

„While static text keystroke dynamics biometrics are often used during the logon process to provide a onetime authentication, free text keystroke biometric systems enable continuously authentication of a user during the entire session for increased security.” [ZHO15] [IAP21a]

2.2.5 Di-graph

A di-graph is a sequence of two consecutive keys. The time for which each keystroke was pressed is named as key hold time or dwell time. [BAN12] The dwell time is the Down-Up time for one single key. In the Figure 2.7, the graph shows the distribution of dwell times (DU) for one of the users from data set. [IAP21a]

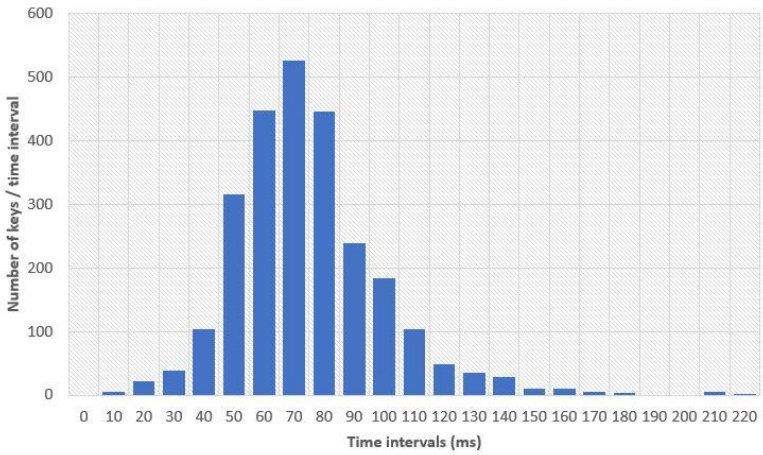


Figure 2.7 Distribution of dwell times (DU) for one of the users from data set [IAP21a]

The Release - Press time or Up-Down time between two consecutive keys was called Flight Time [STE10] [IAP21a].

A di-graph is a sequence of two consecutive keys pressed by the user. When pressing two consecutive keys we will take 4 time periods, noted on the image with t1, t2, t3 and t4. These time periods are captured when the K key (t1) is pressed, when the K key (t2) is raised, when the D key (t3) is pressed and when the D key (t4) is raised [IAP21a]. The time the K key is pressed is dwell time and is calculated as the difference between t2 and t1:

$$DU\ (K) = t2 - t1 \tag{2.1}$$

Flight Time represents the time period between the 2 keys, or more precisely the time from which the first key is left until the second key is pressed [IAP21a]. It is calculated as the difference between t3 and t2 in the image:

$$UD\ (K\text{-}D) = t3\text{-}t2 \tag{2.2}$$

In the same way as the calculation method for (4.1) Dwell Time is calculated for the second key (in our example, the D key) [IAP21a]. The time the second key was pressed is calculated as the difference between t4 and t3:

$$DU\ (D) = t4\text{-}t3 \tag{2.3}$$

In the Figure 2.8 the graph shows the distribution of flight times (UD) for one of the users from data set.

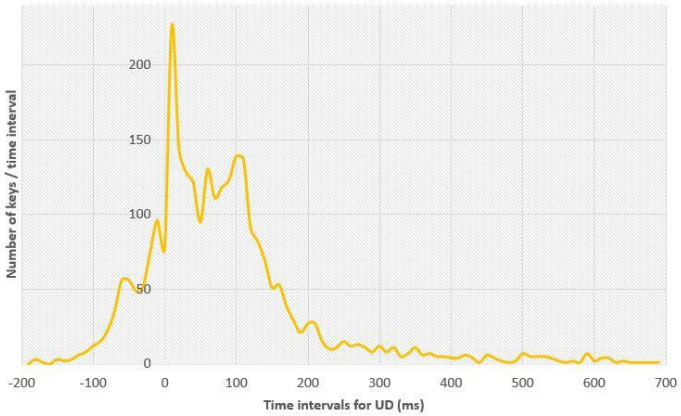


Figure 2.8 Distribution of flight times (UD) for one of the users from data set [IAP21a]

Other time periods that can be calculated and used in algorithms are [IAP21a]:

• the time period between pressing the two DD (K-D) keys that we calculate, according to the notations in the figure as the difference between t3 and t1:

$$DD\ (K\text{-}D) = t3\text{-}t1 \tag{2.4}$$

the time between raising the first key and raising the second key, in our drawing it is about the difference between times t4 and t2:

$$UU\ (K\text{-}D) = t4\text{-}t2 \tag{2.5}$$

the total time required to press the 2 keys, in our example, is calculated as the difference between t4 and t1:

$$D1U2\ (K\text{-}D) = t4\text{-}t1 \tag{2.6}$$

Furthermore, to dwell and flight times, additional ways allow the user to extract data from keystrokes dynamics [JAY19] [FLI10] [RYB08]:

- Typing speed
- Overlap of specific keys combinations
- Number or percentage of errors
- Method of error correction
- Persistent use of navigation-specific keys
- Seek time (time required to press a subsequent key in a common digraph base)

- Characteristic errors

Another feature to evaluate extracted from key press and key release events is also typing speed. [SHU13][LIM14]

Figure 2.9 shows an event required to retrieve the data for a di-graph, a sequence of two consecutive keys pressed by the user.

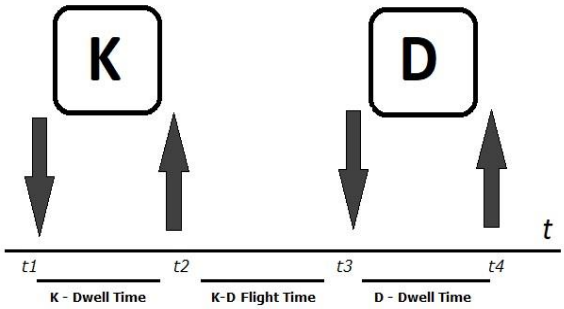


Figure 2.9 Key events and time intervals for a di-graph [IAP21a]

2.2.6 N-graph

Two keys typed one after the other with their typing time are called a di-graph. Similarly, three consecutively typed keys with their typing time are called a tri-graph. And in general, N keys typed one after the other with their typing time are called a n-graph.

In their survey, in [TEH13], the authors noticed that 80% used di-graphs, 7% tri-graphs and only 4% used n-graphs. The n-graphs can be used with success in the experiments that have a big amount of input text.

2.2.7 Metric distances

Given that typing times result in time vectors, and these must be compared to see the similarities between them to identify or validate the user, the convenient method that is also used frequently is to calculate the distance of two vectors. In this way we can say that whether some vectors are similar or not similar. To calculate the distance, several types of distances between two vectors are used in the literature. Each distance can be effective in given cases, in certain circumstances [IAP21a].

Given two typing samples of the same letters is necessary to approximate their similarity or their difference. Is necessary to choose a measure of the distance of the two samples. [BER02] [IAP21a]

2.2.7.1 Euclidian Distance

Euclidian distance is the most used distance between two points. For points given by Cartesian coordinates in n-dimensional Euclidean space, the distance is [TAB14]:

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2 + \cdots + (x_n - y_n)^2} \quad (2.7)$$

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (2.8)$$

In [ZHO15], the authors conclude that "despite its intuitiveness and simplicity, Euclidean distance has two limitations:

- It is highly sensitive to scale variations in the feature variables
- It has no means to deal with the correlation between feature variables." [IAP21a]

2.2.7.2 Manhattan Distance

For points given by Cartesian coordinates in n-dimensional space, the Manhattan distance is:

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i| \quad (2.9)$$

The Manhattan distance has the advantages of easy de-composition into contributions made by each variable and simple computation [ZHO15] [IAP21a].

2.2.7.3 R Distance

The R distance was introduced in [BER02] in 2002. The authors described and tested a new biometric measure of the typing characteristics of individuals. In the case of fixed text R distance provided good results [IAP21a].

An example of calculating the distance R is given by the authors in [GUN05]. Figure 2.10 and formula show the method of calculating this distance.

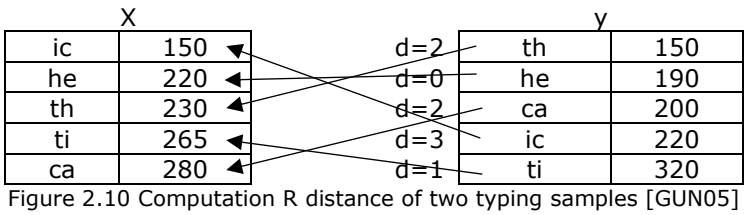


Figure 2.10 Computation R distance of two typing samples [GUN05]

$$d(x,y) = \frac{2+0+2+3+1}{12} = \frac{8}{12} = 0.66 \quad [GUN05] \ (2.10)$$

2.2.7.4 A Distance

A measure only considers the absolute value of the typing speed of each pair of identical n-graphs in the two samples under comparison. [BER02]

An example of calculating the distance A is given by the authors in [GUN05]. Figure 2.11 and formula show the method of calculating this distance, at t=1.25.

x		Y	
280	Ca	200	280/200=1.4
220	He	190	220/190=1.157 (similar pair) (1)
150	Ic	220	220/150=1.466
230	Th	150	230/150=1.533
265	Ti	320	320/265=1.207 (similar pair) (2)

Figure 2.11 Computation A distance of two typing samples [GUN05]

$$d(x,y) = 1 - \frac{2}{5} = 1 - 0.4 = 0.6$$

[GUN05] (2.11)

2.2.7.5 Bhattacharyya Distance

The Bhattacharyya distance between two vectors is defined as [IAP21a]:

$$d(x,y) = -\ln\left(BC(x,y)\right) \tag{2.12}$$

where

$$BC(x,y) = \sum_{i=1}^{n} \sqrt{x_i y_i} \tag{2.13}$$

where n is the dimensions of the vectors x and y.

2.2.7.6 Mahalanobis Distance

Mahalanobis Distance has been popularly used to match keystroke features because it handles the correlated data well [ZHO15]. The squared Mahalanobis distance is defined as:

$$(x-y)^2 = (x-y)^T S^{-1} (x-y) \tag{2.14}$$

where S is the covariance matrix of the data. [IAP21a]

"Mahalanobis distance is related to the logarithmic likelihood under the assumption that the data follows a multivariate Gaussian distribution, which is a reasonable approximation for most practical data." [ZHO15]

2.2.7.7 Distance Metric Fusion

The results of the same researches show that with regards of this field it is often proposed to combine (merge) two or more metric mutes in order to obtain better

performance. For example, in [AYO19] compare the performances of 3 distance calculation methods, but combine two by two and then combine all 3.

2.2.8 Keystroke dymanics authentication algorithms from the literature

This subchapter lists the methods used in various continuous authentication algorithms using keystroke dynamics presented in another scientific research in this field [IAP21b].

Leggett's zone of acceptance algorithm "assumes that the latencies for all situations in which it occurs of a digraph in the reference profile follow a normal distribution. If the average latency for a digraph in the test sample is between the acceptance area, the digraph is then considered accepted, otherwise, rejected." [LEG88] [IAP21b]

Gunetti and Picardi's algorithm is based on both the A measure and R measure for measuring similarity [GUN05]. In free text keystroke dynamics field, the Gunetti and Picardi's algorithm is considered the state-of the-art [AHM14] [IAP21b].

In [MON00] [JOY90] [BLE91], [AHM14], [ROB98] and [COL99], was applied a statistical classifier, using techniques like k-means or Bayes. In [ARA03] and [RUE97], was applied fuzzy logic using a user's categorization as output. In [LIN97], [OBA97] and [WON01], was applied neural networks, but in [MON00], it was concluded that these algorithms are time consuming. In [HAI00], a neural network, a fuzzy classifier and a statistical were combined [ARA04].

2.2.9 Normalization tehniques

2.2.9.1 Min-max normalization

The simplest normalization technique is the Min–max normalization. Min–max normalization is best technique for the case where the maximum and minimum are known. The minimum and maximum scores can easily shift to 0 and 1, or to -1 and 1. [JAI05]

The formula for min-max normalization is (2.15). The x value is the x' is the normalized value:

$$x` = \frac{x_i - x_{min}}{x_{max} - x_{min}}(new_{max} - new_{min}) + new_{min} \tag{2.15}$$

If the new range is between 0 and 1, then the formula is simplified as follows:

$$x' = \frac{x_i - x_{min}}{x\,max^{-} x_{min}} \tag{2.16}$$

2.2.9.2 Z-score normalization

The z-score normalization technique uses the mean and standard deviation for each feature from a set of data [PAT15]. The z-score is the most commonly used normalization technique. It is optimal for Gaussian data.

The normalized scores are given by:

$$x' = \frac{(x_i - \mu_i)}{\sigma_i} \tag{2.17}$$

where μ is the arithmetic mean and σ is the standard deviation of the given data:

$$\mu_i = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \tag{2.18}$$

$$\sigma_i = \sqrt{\frac{1}{(n-1)} \sum_{i=1}^{n} (x_i - \mu_i)} \tag{2.19}$$

"Z-score normalization does not guarantee a common numerical range for the normalized scores of the different matchers. If the input scores are not Gaussian distributed, z-score normalization does not retain the input distribution at the output. This is due to the fact that mean andstandarddeviation are the optimal location andscale parameters only for a Gaussian distribution." [JAI05]

2.2.9.3 Decimal scaling

Decimal scaling is the reduction of the value of some variables by dividing by 10^n. In this way the number can become subunit.

$$x' = \frac{x_i}{10^n} \tag{2.20}$$

where

$$n = \log_{10} \max(x_i) \tag{2.21}$$

2.2.9.4 Median and median absolute deviation (MAD)

The median absolute deviation – MAD - normalization is insensitive to outliers and does not guarantee the common numerical range [NAN05]. The normalization formula is [LAT11]:

$$x` = \frac{x - \text{median}}{\text{const}(\text{median}|x - \text{median}|)} \tag{2.22}$$

2.2.9.5 Double sigmoid

Double sigmoid normalization provides a linear and non-linear transformation of the scores. For the scores in the region of overlap is linear and for the scores outside the region are non-linearl [NAN05]. The normalization formula is [LAT11]:

$$x` = \frac{1}{1 + \exp\left(-2\frac{x - t}{r1}\right)}, \text{if } x < t$$

[LAT11](2.23)

$$x` = \frac{1}{1 + \exp\left(-2\frac{x - t}{r2}\right)}, \text{if } x \geq t$$

[LAT11](2.24)

t = point of reference
r1 = left edge of the region in which the function is linear
r2 = right edge of the region in which the function is linear

2.2.9.6 tanh-estimator

Tanh-estimator is one of the most efficient and powerful normalization techniques. It is introduced by Hample [BHA18]. The normalization formula is:

$$x` = 0.5\left(\tanh\left(\frac{0.01(x - \mu)}{\delta} + 1\right)\right)$$

[LAT11](2.25)

μ = mean value
δ = standard deviation

In [JAI05] the author made a summary of normalization techniques based on robustness and efficiency presented in the Table 2.1:

Table 2.1 Summary of normalization techniques [JAI05]

Normalization techniques	Robustness	Efficiency
Min–max	No	N/A
z-score	No	High
Decimal scaling	No	N/A
Median and MAD	Yes	Moderate
Double sigmoid	Yes	High
tanh-estimators	Yes	High
Biweight estimators	Yes	High

2.2.9.7 Gaussian mixture model

The Gaussian mixture model was used in statistical modeling tasks. It is a parametric model (it is parameterized by mean vectors and covariance matrixes of

the Gaussian distributions and weights of all of the Gaussian components). The real distribution of the data can be unknown [DEN13]. A Gaussian Mixture Model is a weighed sum of M multivariate Gaussian functions [JAI99].

2.2.10 Length of input

The length of the text that is analyzed when verifying the user's identity is very relevant to the performance of the algorithm. The longer the analyzed text, the better the accuracy by which the identity is verified. At text with shorter lengths the performance of the algorithms decreases. In [AYO19] the authors show that with a small number of di-graphs, like 100 di-graphs, the EER is 35,3%. At 200 di-graphs the ERR drops to 15,3%. With more di-graphs, the performance continues to improve. At one analysis with 1000 di-graphs the authors obtain an ERR of 3,6%. The authors from [KAI11] needs minimal 700 keys typed on the keyboard for their algorithm because they need a minimum number of common di-graphs for the authentication or validation process [IAP21b].

2.2.11 Improvement of initial data

In [HUA16] the authors improved the quality of data by eliminating text considered as being noises. They concluded that the data set obtained from a user must first be filtered. By filtering, they ended up eliminating up to 23.3% of the initial text. The atypical behavior of a user has been eliminated and the performance has improved in relation to false rejection rate.

The type of gibberish proposed in [HUA16] are:

- ”Repetition: Repeating the same characters at least 3 times
- Gaming: gaming patterns, any combinations of the four keys used for movement in games (‘a’,’s’,’d’,’w’) and space
- Distinct: Strings with too few distinct characters (moving window of 15 characters with no more than 5 distinct characters). Parameters are chosen manually. Exemple: reeewereeweerewe
- Length: Long strings (greater than 20) of alphabetical letters. Exemple: idhuduisidjcdcvdscvois
- The unstable keystrokes may generate from such activities as when the user is playing a computer game. Addition of gibberish keystrokes has no impact on false accept rate but increases false reject rate significantly. Filtering implies that a larger test sample is needed before an authentication can be attempted.” [HUA16]

Prescreening the data is essential to maximize the performance of the classifiers from the data being analyzed. The performance of the algorithm is maximized by prescreening and removing non-essential elements. [JAY19] [FAW06]

In the [MON06] was proposed the time interval equalization, a non-linear mapping of time intervals for improve the performance of algorithm.

2.2.12 Updating dynamic datasets

Biometric systems commonly provide good performances but the recognition solutions tend to be affected over time due to aging of biometric data [ANI16] and

changing conditions [FAB08]. Adaptive systems, which adapt the reference over time, have been proposed to deal with such intra-class variability. The authors from [PAU19] provides discussion on adaptive biometrics systems, including formalization, terminology, sources or variations that motivates the use of adaptation, adaptation strategies, evaluation methodology and open challenges and concludes that an important advance in the field would be the standardization of the evaluation protocol of adaptive biometric systems.

The authors from [MON99] used an adaptation mechanism. Every time a successful authentication is performed, the algorithm creates a new updated pattern, saving the new sample and deleting the oldest one [ARA04].

2.2.13 Evaluating the performance of authentication algorithms based on keystroke dynamics

2.2.13.1 Confusion matrix

The confusion matrix has four categories: True positives (TP), False positives (FP), True negatives (TN) and False negatives (FN). The confusion matrix can be used to build points in ROC space [JES06] [IAP21b]. In Table 2.2 is shown a confusion matrix.

Table 2.2 The confusion matrix [JES06]

	Actual positive	Actual negative
Predicted positive	True positives (TP)	False positives (FP)
Predicted negative	False negatives (FN)	True negatives (TN)

2.2.13.2 False Rejection Rate (FRR)

False Rejection Rate (FRR) is ”the probability that a system incorrectly classifies a genuine user as an imposter. FRR is the precent of genuine users that are rejected as imposters.” [ZHO15] [IAP21b]

FRR= number of genuine user incorrectly classifies as an imposter / total number of genuine match attempts

(2.26)

$$\mathrm{FRR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}} \tag{2.27}$$

2.2.13.3 False Acceptance Rate (FAR)

False Acceptance Rate (FAR) is ”the probability that a system incorrectly classifies an imposter as a genuine user. FAR is the percent of imposters that are incorrectly accepted as genuine users, how often an intruder is granted access” [ZHO15] [IAP21b].

The formula by which FAR is calculated is given below:

FAR = number of imposters that are incorrectly accepted / total number of impostor match attempts (2.28)

$$\mathrm{FAR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TN}} \tag{2.29}$$

In the Figure 2.12 are graphically represented the FAR (False Acceptance Rate) and FRR (False Rejection Rate).

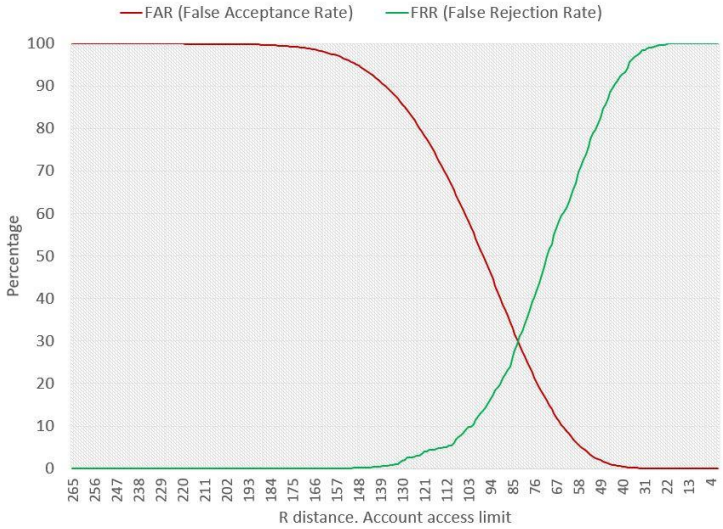


Figure 2.12 A graph with FAR and FRR [IAP21b]

2.2.13.4 Equal Error Rate (ERR)

The equal error rate (EER) is used as a performance metric and it is the point where the FAR equals FRR. The system with the lowest EER is the most accurate. [ZHO15] [IAP21b]

$$\mathrm{ERR} = \mathrm{FAR} = \mathrm{FRR} \tag{2.30}$$

The low accuracy is the main issue of keystroke dynamics [HAB17]. But an EER of 5% is suitable for educational systems that do not require high security [BAR06] [IAP21b].

2.2.13.5 Zero Miss False Acceptance Rate (ZMFAR)

Authentication accuracy is assessed with Equal Error Rate (EER), the percentage at which False Acceptance Rate (FAR) and False Rejection Rate (FRR) have equal value. Another indicator of algorithm performance, in addition to EER, is, according to [ZHO12] [KIL09] Zero Miss False Acceptance Rate (ZMFAR). ZMFAR is represented by the minimum percentage of FRR (False Rejection Rate) when FAR (False Alarm Rate) has the value equal to 0. In Figure 2.13 are graphically represented

the two performance indicators of a user authentication algorithm in the system [IAP21b].

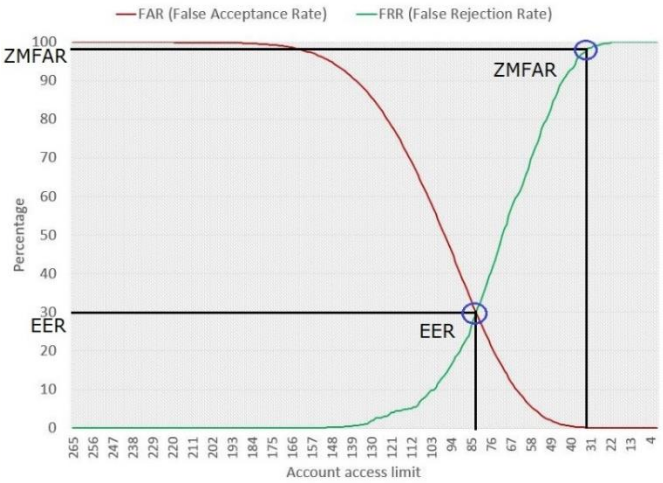


Figure 2.13 Graphical reprezentation of Equal Error Rate (EER) and ZMFAR (Zero Miss False Acceptance Rate) [IAP21b]

2.2.13.6 Receiver operating characteristic (ROC) curve

Receiver Operating Characteristic (ROC) curves describe an entire range of achievable performance characteristics relative to FAR and FRR's. [BAR06] In machine learning, Receiver Operating Characteristic (ROC) curves are used to present results for binary decision problems. ROC curves have many properties when the class distribution is close to being uniform. The confusion matrix can be used to build points in ROC space [JES06] [IAP21b]. In the Figure 2.14 are represented a Receiver operating characteristic (ROC) curve.

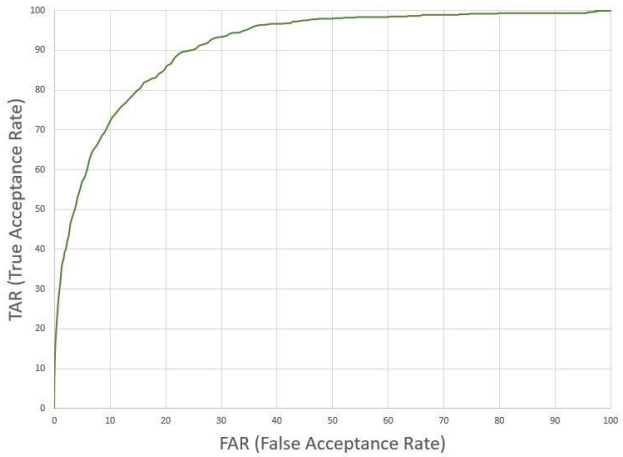


Figure 2.14 Receiver operating characteristic (ROC) curve [IAP21b]

2.3 Conclusions

In this chapter, the current state-of-the-art in the field of doctoral thesis was presented. The scientific papers cited in this chapter, as well as the presentations of concepts, methods, metrics, algorithms, are starting points for present research.

The first part of the chapter, at subchapter 2.1 Evolution of educational systems, describes the evolution of online education systems, an evolution that has met significant increases in recent years and especially in 2020, in the context of moving the classical education system to the online space due to certain constraints such as maintaining the physical distance between people because of the health crisis caused by the SARS-CoV-2 virus. The keystroke dynamics authentication method can be successfully applied as a second mandatory authentication method in case of authentication within online education platforms and especially during exams, when the user's identity must be confirmed throughout the session, not only once at the beginning. This additional verification would be required for two more reasons: 1.the educational systems with Massive Open Online Courses (MOOC) have seen a great growth from its appearance until today, reaching tens of millions of users, there are exams that are given on these platforms with thousands of students at the same time and 2.the medical crisis generated by the SARS-CoV-2 virus in 2020 provoked unprecedented travel restrictions around the globe, and the educational system was moved to an online one.

The second part of this chapter, subchapter 2.2 Keystroke dynamics - literature review, makes a generic presentation of the research stage in the field of keystroke dynamics, presents, in the subchapter 2.2.1, its classification as human biometrics that helps to identify the individual, as well as other biometrics: fingerprint, iris, facial recognition, how to shake hands, etc. It also presents, from the subchapter 2.2.2, the route of scientific research and branches of the field. The categories of keystroke dynamics analysis researched are: Fixed text keystroke dynamics, presented in subchapter 2.2.3, and Free text keystroke dynamics, presented in subchapter 2.2.4. The methods to group characters typed on a keyboard for further analysis in an algorithm are the following: analysis of consecutive key pairs (di-graphs), presented in subchapter 2.2.5, or groups of n consecutive keys (n-graphs), presented in subchapter 2.2.6. The authentication algorithm based on keystroke dynamics takes over, for each key pressed, key code, the time when it was pressed and the time at which it was picked up. In this way, for each user there will be a long series of keys and times. By processing this input data, the user can be identified. With the help of the pressing times, respectively of leaving the key, it can easily calculate the total time when a certain key has been pressed or the total time elapsed between two consecutive keys. Regardless of the analysis of the user's typing mode (one key, di-graph, tri-graph or n-graph analysis) the input data for the authentication algorithm are time vectors (intervals when a key has been pressed, or how many took until the press of the next key). The algorithm will process this input data to decide if the user who is now at the computer is the one who claims to be and can log in to the system. Time vectors are vectors of real numbers. In order to analyze the vectors of real numbers (time vectors) and to decide their similarity, different methods can be approached: (1) distance based classifier, (2) statistical classifier -generic, (3) probability classifier, (4) clustering, (5) machine learning methods - generic, (6) neural networks, (7) fuzzy logic, (8) decision tree, (9) evolutional computing, (10) SVM - support vector machines etc. In the subchapter 2.2.7, Metric distances, the main metrics for calculating the distances between two vectors were presented. In the subchapter 2.2.8 was presented the Keystroke dynamics authentication

algorithms from the literature. It analyses the conception and functioning of the algorithms submitted in scientific research in this field in order to be a starting point for achieving the objective.

In the subchapter 2.2.9 was presented Normalization techniques, that helps to format the initial data which are input data for an authentication algorithm based on keystroke dynamics in order to obtain similar data from all users, for better performances. The techniques presented to standardize are used in algorithms that are presented in scientific papers in the field. The last three subchapters, 2.2.10 Length of input, 2.2.11 Improvement of initial data and 2.2.12 Updating dynamic datasets analyzes particular cases in the development of authentication algorithms using keystroke dynamics.

The following chapter, Chapter 3, will present the research methodology applied by the author to the present scientific research.

3 RESEARCH METHODOLOGY

The previous chapter presented state-of-the-art in the field of keystroke dynamics, throughout which the evolution of the domain, the development steps of an algorithm were analyzed and presents the evolution of educational systems, the field in which authentication based on keystroke dynamics can be applied, especially in the context in which education on online platforms experienced a major increase in 2020.

In the following, this chapter will present the research methodology applied in this research project. The steps performed in the present scientific research are described below:

A. Development of the platform for the acquisition of input data

B. Acquisition and initial processing of input data from 80 volunteers (how typing on their keyboard)

C. Processing the input data so as to generate a user pattern for each user

D. Development of an algorithm in the C programming language for calculating distances used in keystroke dynamics authentication

E. Simulation of system authentication by genuine users or impostors to measure the performance of the developed algorithm

The first two steps of the research methodology, A. Development of the platform for the acquisition of input data and B. Acquisition and initial processing of input data from 80 volunteers (how typing on their keyboard), have the role of approaching O1, as described in the first chapter of the thesis: to collect a database with the test pattern from at least 80 users, in order to test the authentication algorithm for this research, but also to make it available to other interested researchers.

The third step of the research methodology, C. Processing the input data so as to generate a user pattern for each user, and the last step of the research methodology, E. Simulation of system authentication by real users or imposters to measure the performance of the developed algorithm, have the role of approaching O4, as described in the first chapter of the thesis: to propose a data structure as efficient as possible, which should contain the most relevant information about the typing of a user.

The third step of the research methodology, C. Processing the input data so as to generate a user pattern for each user, and the forth step of the research methodology, D. Development of an algorithm in the C programming language for calculating distances used in keystroke dynamics authentication, have the role of approaching O2, as described in the first chapter of the thesis: to implement an algorithm for authenticating the users of a computer based on the keystroke dynamics, the keyboard typing mode.

The last step of the research methodology, E. Simulation of system authentication by genuine users or impostors to measure the performance of the

developed algorithm, has the role of approaching O3, as described in the first chapter of the thesis: to propose at least two new metrics for calculating the distances between two vectors that generate better performance compared to the Equal Error Rate (EER) performance indicator than the classical methods.

Figure 3.1 shows the steps of research methodology, the steps performed in the present scientific research.

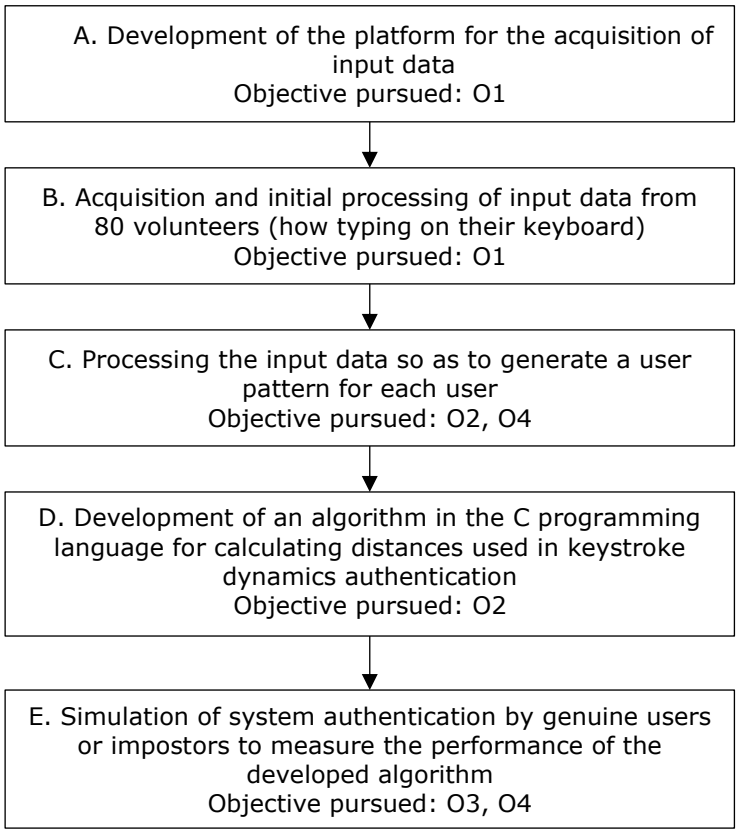


Figure 3.1 Summary of the research methodology applied in the thesis

Next, each step of the research methodology listed above will be described in detail, in a separate subchapter.

3.1 A. Development of the platform for the acquisition of input data

The first step in this research was to create a web platform for the acquisition of input data necessary for research. For this, the website from https://sites.google.com/view/cataliniapa was created, a form was created that would take over, besides the text typed by the users, the way of typing on the keyboard. A program in JavaScript language was written to take over the keystroke times. In order

to be able to download the necessary information, a Google Sheet file was configured, and the information collected using the web form was transmitted using the platform https://api.apispreadsheets.com/. The platform for acquiring input data has been completed and functional by integrating the script written in JavaScript with the data transfer application in the Google Sheet file. The steps described can be followed in the graph in Figure 3.2.

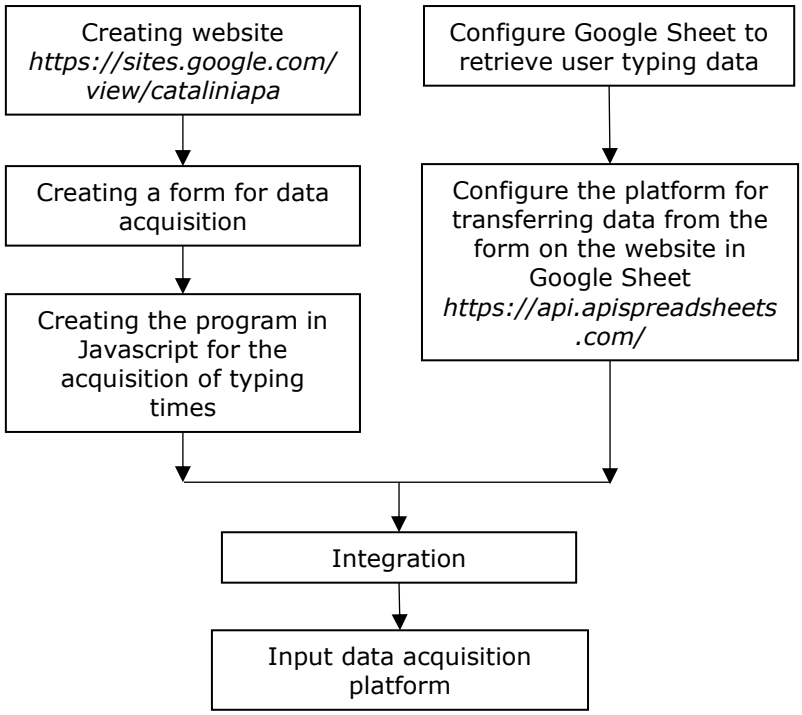


Figure 3.2 Steps taken to create the platform for retrieving data on how users type

This step of the research methodology has the role of approaching O1, as described in the first chapter of the thesis: to collect a database with the test pattern from at least 80 users, in order to test the authentication algorithm for this research, but also to make it available to other interested researchers. This step is detailed in the next chapter, Chapter 4, at subchapter 4.1 Platform for collecting data about keyboard typing from 80 volunteers.

3.2 B. Acquisition and initial processing of input data

The acquisition and initial processing of the input data went through the following steps: Data were collected from 80 users using a web program written in JavaScript. It was collected from the 80 volunteers, through a form, the keys typed on the keyboard but also the times at which they were typed. The collected data was initially stored in a Google Sheet file via the https://api.apispreadsheets.com/ platform. With a program written in the C programming language, the data collected

in the Google Sheet file was processed and transformed into key events in the following form:

68 0 123444
68 1 123555
59 0 123720
71 0 123800
59 1 123830
71 1 123992
...

where on the first column is the key code of the pressed key, on the second column is 0 or 1, 0 represents the pressed key, and 1 represents the raised key, and the third column represents the timestamps at which the key event occurred. The file with the form presented above is the input file for the continuous authentication algorithm developed in this thesis using the keystroke dynamics method. The steps described above are summarized in the graph in Figure 3.3.

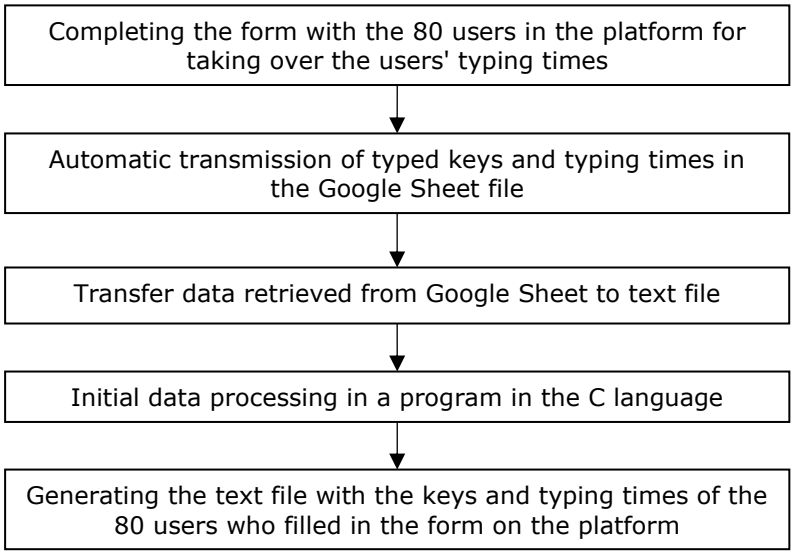


Figure 3.3 Steps taken for the acquisition and initial processing of key data and typing times of the 80 volunteers

This step of the research methodology has the role of approaching O1, as described in the first chapter of the thesis: to collect a database with the test pattern from at least 80 users, in order to test the authentication algorithm for this research, but also to make it available to other interested researchers. This step is detailed in the next chapter, Chapter 4, at subchapter 4.3 Acquisition and initial processing of input data from 80 volunteers (how typing on the keyboard).

3.3 C. Processing the input data so as to generate a user pattern for each user

In order to accomplish this step described in the research methodology, it was necessary to define the data structures for one key, di-graph, user pattern as well as to collect the input data from the text file in order to popularize the data structures within the program. The steps described in this subchapter are shown in the graph in Figure 3.4.

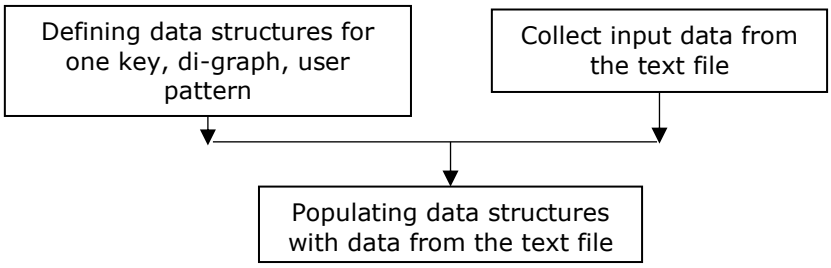


Figure 3.4 Generate user pattern for each user

This step of the research methodology has the role of approaching O4, as described in the first chapter of the thesis: to propose a data structure as efficient as possible, which should contain the most relevant information about the typing of a user and O2, as described in the first chapter of the thesis: to implement an algorithm for authenticating the users of a computer based on the keystroke dynamics, the keyboard typing mode. This step is detailed in the next chapter, Chapter 4, at subchapter 4.5 Processing the input data so as to generate a user pattern for each user.

3.4 D. Development of an algorithm in the C programming language for calculating distances used in keystroke dynamics authentication

Figure 3.5 shows the steps performed to calculate the distances between each two generated vectors, so as to approximate the similarity between two users. For this, the functions were written to calculate the distances between two vectors and were applied to the vectors resulting from the data present in each user pattern.

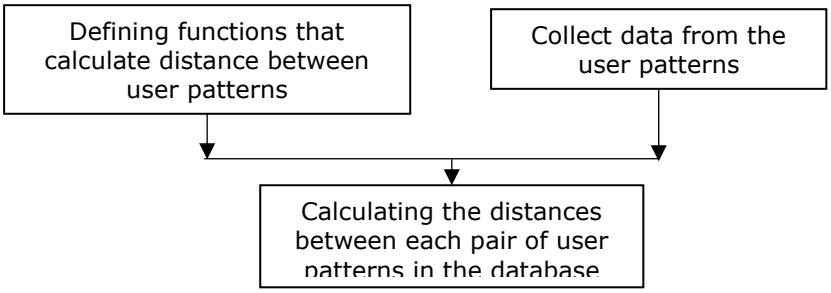


Figure 3.5 Calculating the distance between each pair of user patterns

This step of the research methodology has the role of approaching O2, as described in the first chapter of the thesis: to implement an algorithm for authenticating the users of a computer based on the keystroke dynamics, the keyboard typing mode. This step is detailed in the Chapter 5 - Algorithm development for keystroke dynamics authentication.

3.5 E. Simulation of system authentication by genuine users or impostors to measure the performance of the developed algorithm

In the previous step, the distances between user-generated time vectors were calculated to approximate the similarity between users. Based on them, the authentication was simulated and False Rejection Rate (FRR), False Acceptance Rate (FAR), True Acceptance Rate (TAR) and True Rejection Rate (TRR) were generated. Based on FRR and FAR, Equal Error Rate (EER) was calculated and the FAR-FRR graph was generated. Based on FAR and TAR, ROC curves were generated. These indicate the performance of the authentication algorithm. The steps described can be found in the graph in Figure 3.6.

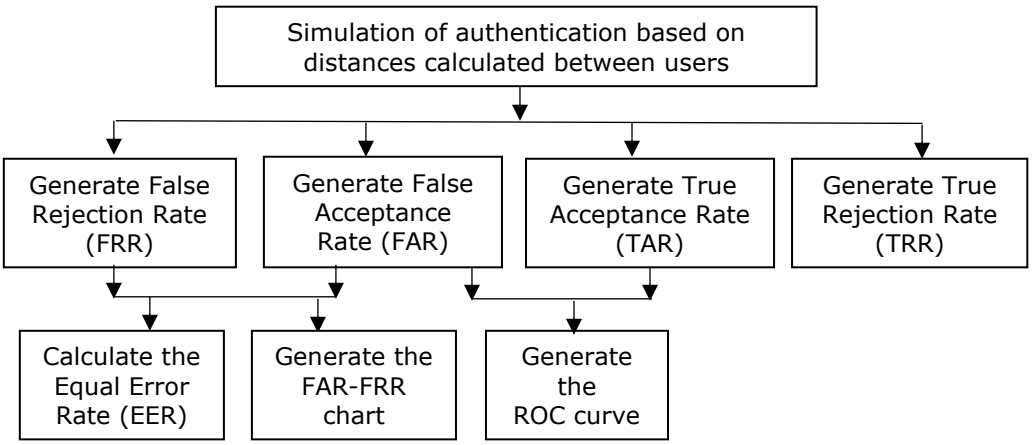


Figure 3.6 Simulation of authentication and calculation of algorithm performance

This step of the research methodology has the role of approaching O3, as described in the first chapter of the thesis: to propose at least two new metrics for calculating the distances between two vectors that generate better performance compared to the Equal Error Rate (EER) performance indicator than the classical methods. This step is detailed in the Chapter 6 – Experiments and results - Simulation of system authentication by genuine users or impostors.

3.6 Conclusions

This chapter described the research methodology. The five steps of the research methodology that have been presented in this chapter are: A. Development

of the platform for the acquisition of input data, B. Acquisition and initial processing of input data from 80 volunteers (how typing on their keyboard), C. Processing the input data so as to generate a user pattern for each user, D. Development of an algorithm in the C programming language for calculating distances used in keystroke dynamics authentication and E. Simulation of system authentication by genuine users or impostors to measure the performance of the developed algorithm.

The first step of the research methodology, A. Development of the platform for the acquisition of input data, has the role of approaching O1, as described in the first chapter of the thesis: to collect a database with the test pattern from at least 80 users, in order to test the authentication algorithm for this research, but also to make it available to other interested researchers. This step will be detailed in the next chapter, Chapter 4, at subchapter 4.1 Platform for collecting data about keyboard typing from 80 volunteers.

The second step of the research methodology, B. Acquisition and initial processing of input data from 80 volunteers (how typing on their keyboard), has the role of approaching O1, as described in the first chapter of the thesis: to collect a database with the test pattern from at least 80 users, in order to test the authentication algorithm for this research, but also to make it available to other interested researchers. This step will be detailed in the next chapter, Chapter 4, at subchapter 4.3 Acquisition and initial processing of input data from 80 volunteers (how typing on the keyboard).

The third step of the research methodology, C. Processing the input data so as to generate a user pattern for each user has the role of approaching O4, as described in the first chapter of the thesis: to propose a data structure as efficient as possible, which should contain the most relevant information about the typing of a user and the role of approaching O2, as described in the first chapter of the thesis: to implement an algorithm for authenticating the users of a computer based on the keystroke dynamics, the keyboard typing mode. This step will be detailed in the next chapter, Chapter 4, at subchapter 4.5 Processing the input data so as to generate a user pattern for each user.

The fourth step of the research methodology, D. Development of an algorithm in the C programming language for calculating distances used in keystroke dynamics authentication, has the role of approaching O2, as described in the first chapter of the thesis: to implement an algorithm for authenticating the users of a computer based on the keystroke dynamics, the keyboard typing mode. This step will be detailed in the Chapter 5 - Algorithm development for keystroke dynamics authentication.

The fifth step of the research methodology, E. Simulation of system authentication by genuine users or impostors to measure the performance of the developed algorithm, has the role of approaching O3, as described in the first chapter of the thesis: to propose at least two new metrics for calculating the distances between two vectors that generate better performance compared to the Equal Error Rate (EER) performance indicator than the classical methods. This step will be detailed in the Chapter 6 – Experiments and results - Simulation of system authentication by genuine users or impostors.

Based on the methodology presented in this chapter, the following chapters will present the data set collected in this research, the algorithm developed to identify users using keystroke dynamics as well as improvements to increase its performance and the experiments and results. The next chapter presents the platform with which the data were collected, the way of collecting data from 80 users and also an analysis of the data.

4 FREE-TEXT KEYSTROKE DYNAMICS DATA SET FOR CONTINUOUS AUTHENTICATION

In the previous chapter was presented the research methodology. The research methodology assumes, first of all, the existence of data from users.

This chapter is about the data set collected for the present research. The first subchapter 4.1, Platform for collecting data about keyboard typing from 80 volunteers, presents the platform for collecting data about keyboard typing and how data was collected from 80 users in order to later develop an algorithm. Moreover, after presenting the platform with the help of which data were collected from users, it will proceed to the analysis of these collected data and to the presentation of the particular characteristics, in subchapthers 4.2 Analysis of time and key events collected from users, 4.3 Acquisition and initial processing of input data from 80 volunteers (how typing on the keyboard) and 4.4 Analysis of keys collected from users. It shows how they were processed using an algorithm written in the C programming language. The subchapter 4.5, Processing the input data so as to generate a user pattern for each user, presents the structures that store user typing data. The subchapter 4.6, Keys distribution analysis, presents the analysis of the collected keys. The subchapter 4.7, Differences between users, graphically displays the typing pattern for different users. This chapter addresses the validation of O1 from the first chapter of this thesis.

4.1 Platform for collecting data about keyboard typing from 80 volunteers

To research in the field of keystroke dynamics biometrics the researchers need input data obtained from computer users in different real situations. The necessary data are represented by the keys typed on the keyboard but also by the times at which they are pressed. The time when a certain key is pressed, respectively the time when a certain key is raised. The difference between these times is the keystroke time. Another important piece of information is the time between two keys. The difference between the time a key was released and the time a next key was pressed [IAP21a].

This information can only be obtained in a restrained or controlled environment, with the consent of those participating to this experiment. The agreement of the participants is necessary because it exists a possibility to form the initial text that the user typed on the keyboard with access to this data, and if, for example, a user is monitored while sending e-mails or doing other activities, the information may be confidential.

In the literature there are several sets of data that are accessible for research purposes. In the first phase, the author used these data sets. Most are represented by texts in English, obtained from the educational environment, by researchers from their university colleagues or from students.

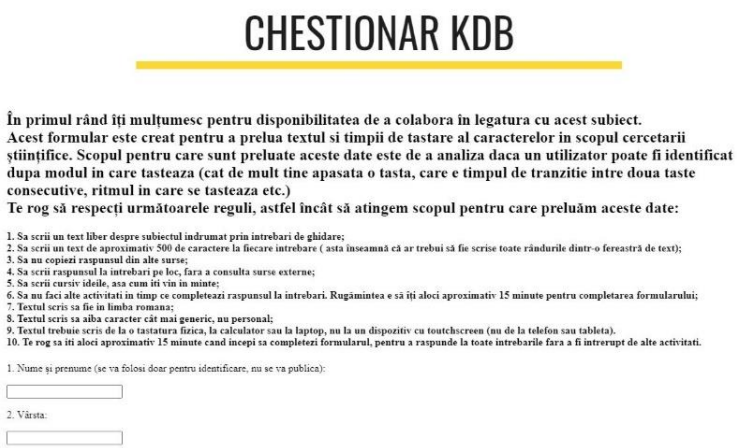
Some data sets are retrieved by a specific program, in a special environment made for this purpose. Others are made to monitor everything that is typed on a computer, regardless of the program used at one time by the user. It monitors everything typed on the keyboard and typing times whether the user is writing e-mails, writing in a Word, Excel document or programming on a computer in a certain programming environment.

On the other hand, for the purpose of the research the author developed their own environment to obtain data from volunteers. The author has created a web environment for taking over keys and typing times in JavaScript. A form is created that takes over the keys and typing times while completing a form on a web page [IAP21a]. The website was created on the sites.google.com platform. The web platform can be accessed at https://sites.google.com/view/cataliniapa.

To capture the keys and typing times the author created a web form through which users were invited to answer several generic questions. The text entered from the keyboard by each user should be written freely by each user, without the need to reproduce a specific predefined text. At each text box, a series of generic questions were formulated to guide the user to a certain topic in the text he completed. The questions asked were about the weather, the ideal day or the educational system. To form the database for research is not relevant the topic of the text, but the way it is written.

The text written by users is in Romanian. Most datasets in the literature are texts captured from users who have written in English [IAP21a].

Figure 4.1 shows the online form that each user filled in during the experiment.


CHESTIONAR KDB

În primul rând îți mulțumesc pentru disponibilitatea de a colabora în legatura cu acest subiect.
Acest formular este creat pentru a prelua textul si timpii de tastare al caracterelor in scopul cercetarii științifice. Scopul pentru care sunt preluate aceste date este de a analiza daca un utilizator poate fi identificat dupa modul in care tasteaza (cat de mult tine apasata o tasta, care e timpul de tranzitie intre doua taste consecutive, ritmul in care se tasteaza etc.)
Te rog să respecți următoarele reguli, astfel încât să atingem scopul pentru care preluăm aceste date:

1. Sa scrii un text liber despre subiectul indrumat prin intrebari de ghidare;
2. Sa scrii un text de aproximativ 500 de caractere la fiecare intrebare (asta înseamnă că ar trebui să fie scrise toate rândurile dintr-o fereastră de text);
3. Sa nu copiezi raspunsul din alte surse;
4. Sa scrii raspunsul la intrebari pe loc, fara a consulta surse externe;
5. Sa scrii cursiv ideile, asa cum iti vin in minte;
6. Sa nu faci alte activitati in timp ce completeazi raspunsul la intrebari. Rugămintea e să îți aloci aproximativ 15 minute pentru completarea formularului;
7. Textul scris sa fie in limba romana;
8. Textul scris sa aiba caracter cât mai generic, nu personal;
9. Textul trebuie scris de la o tastatura fizica, la calculator sau la laptop, nu la un dispozitiv cu toutchscreen (nu de la telefon sau tableta).
10. Te rog sa iti aloci aproximativ 15 minute cand incepi sa completezi formularul, pentru a raspunde la toate intrebarile fara a fi intrerupt de alte activitati.

1. Nume și prenume (se va folosi doar pentru identificare, nu se va publica):

2. Vârsta:

Figure 4.1 The form that the users filled in

After completing all the fields in the form, in order to send the captured data, the consent regarding the takeover for the purpose of scientific research of the participants was obtained. Two questions answered by users from the form are in Figure 4.2. First one is about weather and second one about the ideal day.

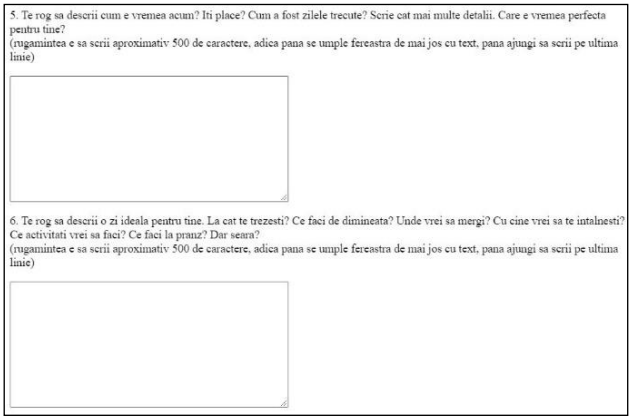


Figure 4.2 Two questions answered by users from the form

Each user was instructed to pursue the following rules when filling out the form:

1. To write a free text about the subject managed by guidance questions;
2. Write a text of about 500 characters for each question (this means that all the lines in a text window should be filled);
3. Do not copy the answer from other sources;
4. Write the answer to the questions on the spot, without consulting external sources;
5. Write ideas fluently, as they come to mind;
6. Do not do other activities while completing the answer to the questions. The request is to allocate about 15 minutes to complete the form;
7. The written text must be in Romanian;
8. The written text should be as generic as possible, not personal;
9. The text should be written from a physical keyboard, computer or laptop, not a touchscreen device (not a phone or tablet).
10. Please take about 15 minutes to complete the form to answer all questions without being interrupted by other activities.

In one of the questions on the form, users were asked to describe the scene in the Figure 4.3, in as much detail as possible.

Figure 4.3 Each user has described the scene of the picture at one of the questions

To send the data the user pressed the send button so by pressing this button the data captured during the completion of the form were uploaded in a google sheet file as shown in Figure 4.4. The file is organized in 3 columns, as follows: the first column contains the key codes of the captured keys separated by commas, the second column shows the time each key was pressed, and the third column captures the time between two consecutive keys. Times are separated by commas, as is the key codes.

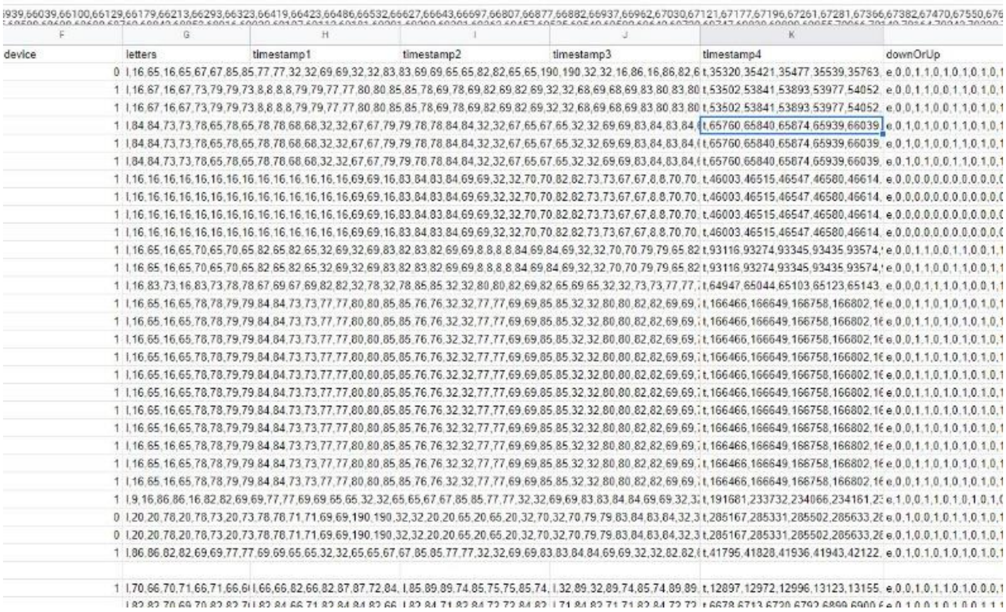


Figure 4.4 Google sheet with the key codes, timestamps and key events from users

To retrieve the data about keys and typing times the author wrote a script in JavaScript. The script is at Code 4.1. To transfer the data from the form to Google Sheet is used the api from apispreadsheets.com.

Code 4.1 The script used to get the keys and typing times from the users

```
<script>
var letters = [];
var timestamp = [];
var event = [];
var name = [];
var age = [];
var tel = [];
var mail = [];
var gender = [];
var gen = [];
var device = [];
var dev = [];
var clk0 = $.now();
var start = null;
var end = null;
```

```
$('#in1').keydown(function (f) {
        start = $.now()-clk0;
        letters.push(f.keyCode);
        timestamp.push(start);
        event.push(0);
   }).keyup(function (f) {
        end = $.now()-clk0;
        letters.push(f.keyCode);
        timestamp.push(end);
        event.push(1);
});
$("#submit").click(function(){
        name = document.querySelector('#in').value;
        age = document.querySelector('#age').value;
        tel = document.querySelector('#tel').value;
        mail = document.querySelector('#mail').value;
        gender = document.getElementsByName('gender');
        if(gender[0].checked) gen=0;
        if(gender[ZHO12].checked) gen=1;
        if(gender[KIL09].checked) gen=2;
        device = document.getElementsByName('device');
        if(device[0].checked) dev=0;
        if(device[ZHO12].checked) dev=1;
                $.ajax({
                        url:'https://api.apispreadsheets.com/data/5917/',
                        type:'post',
                        data: {
                                "name" : name,
                                "age" : age,
                                "tel" : tel,
                                "mail" : mail,
                                "gender" : gen,
                                "device" : dev,
                                "letters" : letters.join(),
                                "timestamp" : timestamp.join(),
                                "downOrUp" : event.join()
                                },
                        success: function(){
                          alert("Form Data Submitted")
                        },
                        error: function(){
                          alert("There was an error")
                        }
                })
        });
</script>
```

4.2 Analysis of time and key events collected from users

The form created to purchase data sets for research purposes was completed by a number of 80 users. They handed over data for 410,633 key-events [IAP21a]. The comprise time used by all 80 users to complete the form was 23 hours, 28 minutes and 19 seconds. For each user, the overall completion time for the form was calculated from pressing the first collected key to pressing the last collected key with the following formula:

$$TotalTime = UpEventLastKey - DownEventFirstKey \quad (4.1)$$

The maximum time to complete the form was spent by user 55, setting aside approximately one hour and 20 minutes to provide 5088 key events. The minimum time was allocated by user 50 and user 24, approximately 4 minutes, but providing even fewer key events, 1237 key events of user 50 and 1121 key events of user 24.

The average time spent by users on the data collection platform is 17 minutes and 36 seconds. Table 4.1 shows the completion times of the form for each user, as well as the average and the total time spent by users to complete. In this regard, the time is expressed not only in milliseconds revealed in the second column of the table, but also in minutes show in the third column of the table.

Table 4.1 Time spent by users to complete the form

User	Time (ms)	Time (min)
Total	**84501613**	**1408.36**
Average	**1056270**	**17.60**
user0001	1095546	18.26
user0002	1177215	19.62
user0003	591972	9.87
user0004	1294786	21.58
user0005	504168	8.40
user0006	443926	7.40
user0007	1076423	17.94
user0008	931092	15.52
user0009	958237	15.97
user0010	869952	14.50
user0011	814627	13.58
user0012	649341	10.82
user0013	570906	9.52
user0014	640486	10.67
user0015	614859	10.25
user0016	923821	15.40
user0017	608300	10.14
user0018	1038345	17.31
user0019	921804	15.36
user0020	570299	9.50
user0021	801803	13.36
user0022	512004	8.53
user0023	1270468	21.17
user0024	278405	4.64
user0025	1324293	22.07
user0026	548089	9.13
user0027	1199995	20.00
user0028	683331	11.39
user0029	538419	8.97
user0030	960423	16.01
user0031	846177	14.10
user0032	684500	11.41
user0033	1044991	17.42
user0034	1504600	25.08
user0035	1298690	21.64
user0036	639693	10.66
user0037	1021443	17.02
user0038	779141	12.99
user0039	728718	12.15
user0040	915012	15.25
user0041	1601844	26.70
user0042	1248721	20.81
user0043	1248392	20.81
user0044	537593	8.96

user0045	1034266	17.24	user0063	996290	16.60
user0046	822826	13.71	user0064	1732492	28.87
user0047	2338553	38.98	user0065	1012393	16.87
user0048	1787919	29.80	user0066	1418069	23.63
user0049	987686	16.46	user0067	1208593	20.14
user0050	249185	4.15	user0068	630424	10.51
user0051	562723	9.38	user0069	854492	14.24
user0052	735564	12.26	user0070	888229	14.80
user0053	829821	13.83	user0071	2803408	46.72
user0054	1488714	24.81	user0072	987762	16.46
user0055	4846025	80.77	user0073	784983	13.08
user0056	553485	9.22	user0074	1601963	26.70
user0057	668374	11.14	user0075	2720502	45.34
user0058	1076782	17.95	user0076	3455548	57.59
user0059	858249	14.30	user0077	965783	16.10
user0060	733605	12.23	user0078	791552	13.19
user0061	1218402	20.31	user0079	786153	13.10
user0062	544233	9.07	user0080	1013715	16.90

The time to complete the form by each user is graphically represented by Figure 4.5. Each user is on the OX axis. On the left side of the chart is placed the user with the shortest completion time of the form, user 50, with a total completion time of 4 minutes and 9 seconds. On the right side of the chart is user 55, who completed the form in one hour, 20 minutes and 45 seconds (80 minutes and 45 seconds). The average time to complete the form is 17 minutes and 36 seconds and it is submitted separately on the graph.

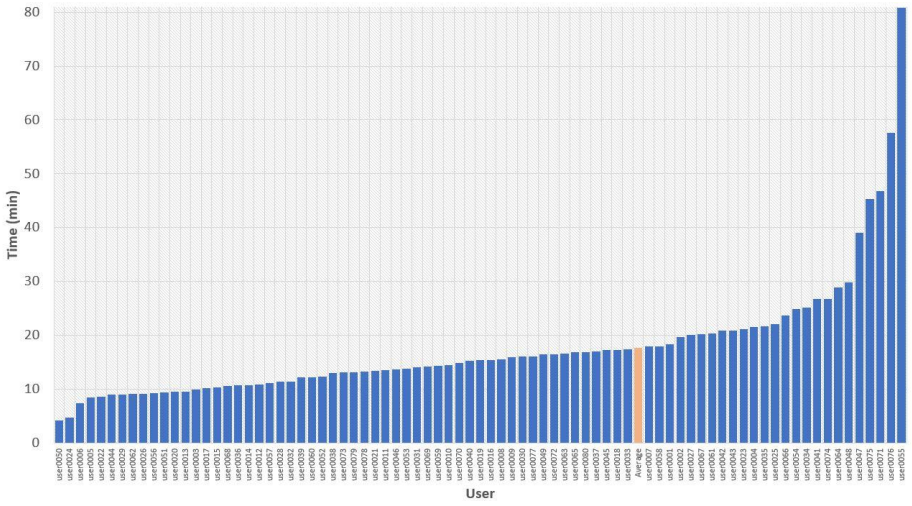


Figure 4.5 Graphical representation of the time spent by each user to complete the form, in ascending order

Table 4.2 shows the total number of key events collected from each of the 80 users who filled out the form. The total number of key events collected from all users is 410,633. The average number per user is 5132 key events. Each key event contains Key Code, Down Event or Up Event and the Time Stamp.

The largest number of key events, a number of 6829, is collected from the 65th user in 16 minutes and 52 seconds. On the second place is user 66, with a number of 6781 key events, collected in a time interval of 23 minutes and 37 seconds.

The fewest key events collected were provided by the users with the shortest time spent on the collection platform, user 24 and user 50, with 1121 key events, respectively 1237 key events. User 36 has also used only 3123 key events, collected in 10 minutes and 39 seconds.

Table 4.2 Number of key events collected from users

User	**TotalKeyEvents**
TotalKeyEvents	**410633**
Average	**5132**
user0001	3905
user0002	5715
user0003	5671
user0004	5992
user0005	4938
user0006	3815
user0007	4422
user0008	5303
user0009	4889
user0010	4021
user0011	6057
user0012	5181
user0013	4696
user0014	5872
user0015	5153
user0016	5312
user0017	4273
user0018	5311
user0019	5379
user0020	4521
user0021	6153
user0022	4612
user0023	3231
user0024	1121
user0025	5083
user0026	5618
user0027	6229
user0028	4550
user0029	3710
user0030	4999
user0031	4871
user0032	5335
user0033	5959
user0034	4977
user0035	5682
user0036	3123
user0037	4942
user0038	5648
user0039	5010
user0040	5759
user0041	5604
user0042	5655
user0043	5766
user0044	5157
user0045	6043
user0046	5762
user0047	5039
user0048	5557
user0049	5112
user0050	1237
user0051	6111
user0052	5587
user0053	5451
user0054	3309
user0055	5088
user0056	4757
user0057	4520
user0058	5336
user0059	5565
user0060	4282
user0061	6606
user0062	4406
user0063	4725
user0064	4896

user0065	6829		user0073	5545
user0066	6781		user0074	6263
user0067	6465		user0075	5809
user0068	5005		user0076	3991
user0069	6034		user0077	5523
user0070	5590		user0078	5863
user0071	5954		user0079	5427
user0072	5642		user0080	5303

Figure 4.6 graphically represents the number of key events collected from each user. Users are sorted in descending order by the total number of key events /user. The average number of key events (5132) collected is highlighted in the graph. The highest number of key events collected is 6829, being provided on the left side of the graph, and the lowest number of key events collected is 1121, being specified on the right side of the graph.

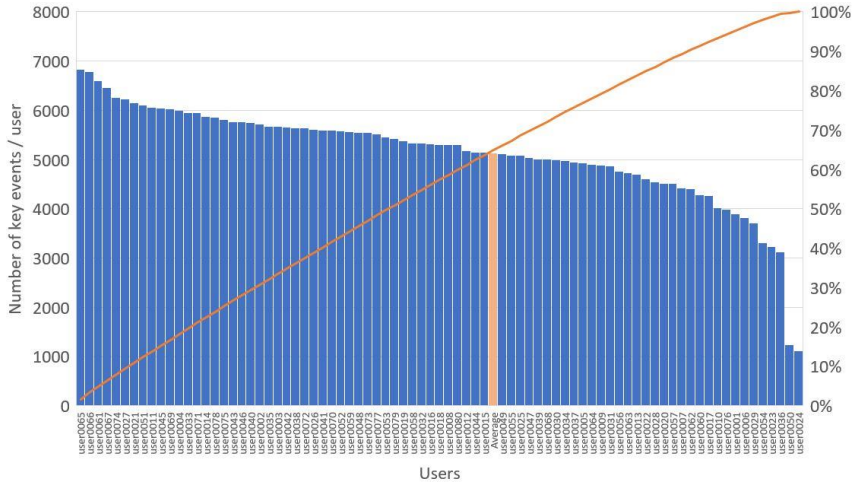


Figure 4.6 Graphical representation of the number of key events collected from each user, in descending order

4.3 Acquisition and initial processing of input data from 80 volunteers (how typing on the keyboard)

The information about the keys and the typing method is transferred in a Google Sheets file in the form of three integer vectors from the form filled in by the users. The first vector contains the code of the keys pressed, the second vector contains the time at which keystrokes occurred, and the third vector contains a vector with digits 0 and 1, which represent Key Down Event and Key Up Event.

In order to correlate the data from the 3 vectors and to create the text file containing the information about key events, it is wrote below the code presented in *Code 4.2*.

Code 4.2 Part of the code that creates key events files

```
int startTime=0, totalTime=0, sumTime=0, nTime=0;
int totalKeyEvents=0, sumTotalKeyEvents=0, nTotalKeyEvents=0;
do
```

```
{
  fscanf(f,"%s",user);
  fscanf(g,"%s",user);
  fscanf(h,"%s",user);
  if(strcmp(user,"-1")==0)
    break;
  if(user[0]=='u' && user[1]=='s' && user[2]=='e' && user[3]=='r')
  {
    strcpy(fileUser,"UsersKeyEvents/");
    strcat(fileUser,user);
    strcat(fileUser,".txt");
    m=fopen(fileUser,"w");
    fprintf(k,"%s\n",user);
    fprintf(l,"%s",user);
    fprintf(o,"%s",user);
    fscanf(f,"%c%c",&c,&c);
    fscanf(g,"%c%c",&c,&c);
    fscanf(h,"%c%c",&c,&c);
    n=0;
    do
      {
        fscanf(f,"%c%d",&c,&keyEvents[n].letter);
        fscanf(g,"%c%d",&c,&keyEvents[n].timestamp);
        fscanf(h,"%c%d",&c,&keyEvents[n].event);
        if(keyEvents[n].letter!=-1)
        {
          fprintf(k,"%d %d
%d\n",keyEvents[n].letter,keyEvents[n].event,keyEvents[n].timestamp);
          fprintf(m,"%d %d
%d\n",keyEvents[n].letter,keyEvents[n].event,keyEvents[n].timestamp);
        }
        if(n==0)
        {
          startTime=keyEvents[n].timestamp;
          totalKeyEvents=0;
        }
        totalKeyEvents++;
        n++;
      }while(keyEvents[n-1].letter!=-1);
      n--;
      totalTime=keyEvents[n-1].timestamp-startTime;
      sumTime=sumTime+totalTime;
      nTime++;
      fprintf(l,"\t%d\t%.2f\n",totalTime,(float)totalTime/60000);
      sumTotalKeyEvents=sumTotalKeyEvents+totalKeyEvents;
      nTotalKeyEvents++;
      fprintf(o,"\t%d\n",totalKeyEvents);
      fprintf(k,"-1\n");
      fclose(m);
  }
}while(strcmp(user,"-1")!=0);
```

In the algorithm written to analyze the data, for each key event it is used a structure to retain the three information collected. The three pieces of information are collected as integers.

The first information obtained about a key event is the key code. Each key is coded by an integer between 8 and 222. The second information collected about a key event is an integer representing the captured event. The captured event can be Key Down or Key Up. I coded the Key Down event with the number 0, and the Key Up event with the number 1. The third information collected about a key event is the time at which the Key Down or Key Up event occurred, taking the time of the computer system expressed in milliseconds.

The three information collected about a key event were retained in the code with the help of a structure that contains 3 integers. In *Code 4.3* is presented the structure used in the program.

Code 4.3 The structure used to store information about a key event

```
typedef struct {
    int letter;          //key code
    int event;           //0 - key down, 1 - key up
    int timestamp;       // time of event
}keyEvent;
```

Figure 4.7 shows how to store key events collected from users in a text file. The first line of the file contains the name of the first user, and the following lines contain information about each key event separately. On each line the 3 numbers are divided by the SPACE key. After I have been written all the information about a user to the file, I have added a line containing the number -1 in order to point out in this way my step forward to the information about the next user. This format is repeated for each user in the database.

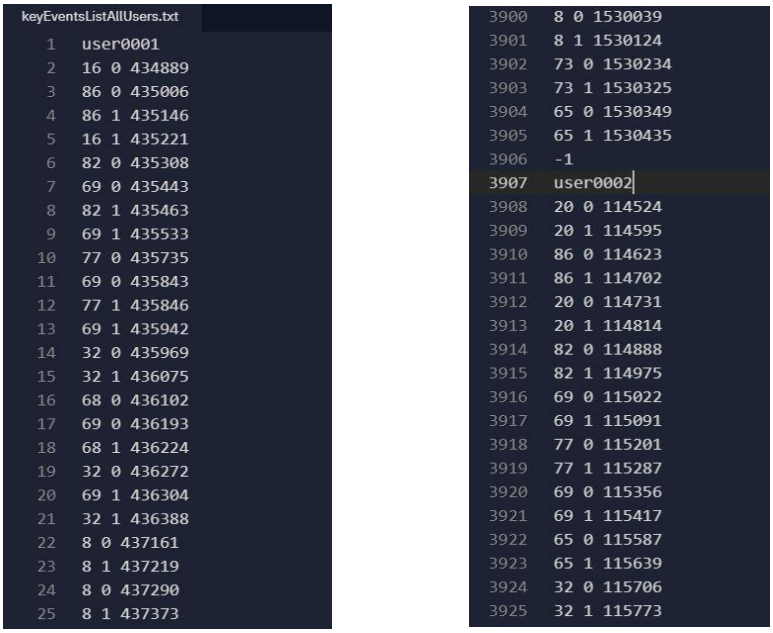


Figure 4.7 The text file that stores key events information

4.4 Analysis of keys collected from users

In order to analyze the collected data, each key event was examined, thus, the text typed by each user was reconstructed. From the initial data containing the key events included each of them 3 types of data: key code, event (down or up) and timestamp, it is obtained information regarding to the total time as each key was pressed (DU), Key Down event time, Key Up event time, Key Code, Key, Previous Key, Next Key, the time between the previous key and the analyzed key, the time between the previously mentioned key and the next key. The order of reconstruction of the text typed by each user took into account the timestamps from each Key Down Event. The total time each key (DU) was pressed was calculated according to the following formula that I will present in the followings:

$$DU = TimeStampKeyUpEvent - TimeStampKeyDownEvent \quad (4.2)$$

The time between two keys (UD) was calculated as the difference between Time of First Key Up Event and Time of Second Key Down Event as in the formula below:

$$UD = TimeStampSecondKeyDownEvent - TimeStampFirstKeyUpEvent \quad (4.3)$$

The keystroke time (DU) will always be a positive number, although the time between two keys (UD) can also be a negative one. A user can press the second key before picking up the first. This way FirstKeyUpEvent can come right after the SecondKeyDownEvent happened.

4.5 Processing the input data so as to generate a user pattern for each user

The structure that stores information about a key is presented below in the Code 4.4 section.

Code 4.4 The structure used to store information about a key

```
typedef struct {
   int letter;                    //Key Code
   int DU, UDprev, UDnext;        //DU = keystroke time, UDprev - previous flight
time, UDnext - next flight time
   int U,D;                       //U - Up timestamp, D - Down timestamp
   int letterPrev, letterNext;    //Key Code of previous and next keys
   char key[20],keyPrev[20],keyNext[20];      //Key, Previous Key and Next Key
}onegraph;
```

Information about a key is extracted from the key events row via the constructOneGraphs function which is presented below in the *Code 4.5* section.

If the time intervals (DU or UD) exceed 999 milliseconds, they end up capped at a higher value. If the time between 2 keys is negative and is less than -199, it will be also capped at such a minimum value. These limitations are chosen due to the fact that there are few cases in which these values are exceeded, for example when a user gets up from the keyboard helps to longer the time between 2 keys (UD), but it is

irrelevant for this study. Also, if, for example, the SHIFT key is pressed and held while other keys are pressed, the pressing time will exceed the 999 milliseconds interval, but this additional data is not of a relevance for the present study as it can vary greatly depending on the length of the word that is typed while the SHIFT key is pressed.

Code 4.5 The function that extracts key information from key events

```
int constructOneGraphs(int n, char user[])
{
  int letter,event,timestamp;
  int i;
  n=0;
  do
  {
    fscanf(f,"%d",&letter);
    if(letter==-1)
    {
      break;
    }
    fscanf(f,"%d%d",&event,&timestamp);
    if(event==1)
    {
     if(n!=0)
     {
      for(i=n-1;i>=0;i--)
      {
        if(oneGraphs[i].letter==letter)
        {
          oneGraphs[i].U=timestamp;
          while(oneGraphs[i].letter==oneGraphs[i-1].letter && oneGraphs[i-1].U==0)
          {
            deleteOneGraph(i-1,n);
            i--;
            n--;
          }
          break;
        }
      }
     }
    }
    if(event==0)
    {
      oneGraphs[n].letter=letter;
      keycodeToKey(oneGraphs[n].key,letter);
      oneGraphs[n].D=timestamp;
      oneGraphs[n].U=0;
      n++;
    }
  }while(letter!=-1);
 n--;
  for(i=0;i<n;i++) //construct oneGraphs
```

```
{
 oneGraphs[i].DU= oneGraphs[i].U - oneGraphs[i].D;
 if(oneGraphs[i].DU>999)
   oneGraphs[i].DU=999;
 if(oneGraphs[i].DU<0)
 {
    deleteOneGraph(i,n);
    n--;
 }
 if(i==0)
 {
   oneGraphs[i].UDprev=0;
   oneGraphs[i].letterPrev=0;
   strcpy(oneGraphs[i].keyPrev,"");
   oneGraphs[i].UDnext=oneGraphs[i+1].D-oneGraphs[i].U;
   oneGraphs[i].letterNext=oneGraphs[i+1].letter;
   strcpy(oneGraphs[i].keyNext,oneGraphs[i+1].key);
  }
  else
  {
   if(i==n-1)
   {
     oneGraphs[i].UDprev=oneGraphs[i].D-oneGraphs[i-1].U;
     oneGraphs[i].letterPrev=oneGraphs[i-1].letter;
     strcpy(oneGraphs[i].keyPrev,oneGraphs[i-1].key);
     oneGraphs[i].UDnext=0;
     oneGraphs[i].letterNext=0;
     strcpy(oneGraphs[i].keyNext,"");
    }
    else
    {
      oneGraphs[i].UDprev=oneGraphs[i].D-oneGraphs[i-1].U;
      oneGraphs[i].letterPrev=oneGraphs[i-1].letter;
      strcpy(oneGraphs[i].keyPrev,oneGraphs[i-1].key);
      oneGraphs[i].UDnext=oneGraphs[i+1].D-oneGraphs[i].U;
      oneGraphs[i].letterNext=oneGraphs[i+1].letter;
      strcpy(oneGraphs[i].keyNext,oneGraphs[i+1].key);
    }
   }
   if(oneGraphs[i].UDnext<-199)
     oneGraphs[i].UDnext=-199;
   if(oneGraphs[i].UDnext>999)
     oneGraphs[i].UDnext=999;
   if(oneGraphs[i].UDprev<-199)
     oneGraphs[i].UDprev=-199;
   if(oneGraphs[i].UDprev>999)
     oneGraphs[i].UDprev=999;
}
for(i=0;i<n;i++)
{
```

```
    fprintf(g, "%s\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%s\t%d\t%s\n",
oneGraphs[i].key, oneGraphs[i].letter, oneGraphs[i].D, oneGraphs[i].U,
oneGraphs[i].DU, oneGraphs[i].UDprev, oneGraphs[i].UDnext,
oneGraphs[i].letterPrev, oneGraphs[i].keyPrev, oneGraphs[i].letterNext,
oneGraphs[i].keyNext);
  }
  return n;
}
```

After running the function presented above, it is created the text file that contains data about each key pressed but also about the relationship with the neighboring keys. The outline of the text file is shown in Figure 4.8. On the first line of the file is the username, while on the following lines are the information about a key separated by the TAB key as follows:

Key, KeyCode, TimeStampKeyDownEvent, TimeStampKeyUpEvent, DUtime, UDtimePreviousKey, UDtimeNextKey, PreviousKeyCode, PreviousKey, NextKeyCode, NextKey.

oneGraphsAllUsers.txt

```
1   user0001.01
2   Shift 16  434889  435221  332 0 -199  0   86  V
3   V 86  435006  435146  140 -199  162 16  Shift 82  R
4   R 82  435308  435463  155 162 -20 86  V 69  E
5   E 69  435443  435533  90  -20 202 82  R 77  M
6   M 77  435735  435846  111 202 -3  69  E 69  E
7   E 69  435843  435942  99  -3  27  77  M 32  Spacebar
8   Spacebar  32  435969  436075  106 27  27  69  E 68  D
9   D 68  436102  436224  122 27  -31 32  Spacebar  69  E
10  E 69  436193  436304  111 -31 -32 68  D 32  Spacebar
11  Spacebar  32  436272  436388  116 -32 773 69  E 8 Backspace
12  Backspace 8 437161  437219  58  773 71  32  Spacebar  8 Backspace
13  Backspace 8 437290  437373  83  71  50  8 Backspace 8 Backspace
14  Backspace 8 437423  437514  91  50  121 8 Backspace 78  N
15  N 78  437635  437727  92  121 -21 8 Backspace 65  A
16  A 65  437706  437807  101 -21 53  78  N 83  S
17  S 83  437860  437942  82  53  153 65  A 80  P
18  P 80  438095  438225  130 153 -31 83  S 65  A
19  A 65  438194  438305  111 -31 744 80  P 190 .
20  . 190 439049  439100  51  744 101 65  A 32  Spacebar
21  Spacebar  32  439201  439262  61  101 136 190 . 16  Shift
22  Shift 16  439398  439668  270 136 -1  32  Spacebar  80  P
23  P 80  439667  439685  18  -1  92  16  Shift 76  L
24  L 76  439777  439892  115 92  92  80  P 79  O
25  O 79  439984  440057  73  92  119 76  L 85  U
```

Figure 4.8 The file used to store information about a key

4.6 Keys distribution analysis

Using information obtained from the 410.633 of key events the author rebuilt the characters typed by each user at the keyboard. A total of 200,299 keys were typed on the keyboard [IAP21a]. The number of key events is more than double the keys because a certain key that has been held down for a long time, such as SHIFT or BACKSPACE generates more than 2 key events. In this case, several Down Key Events and only one Up Key Event will be generated, instead it is a single keystroke.

A total of 100 different keys were monitored. The frequency with which the keys appeared in the text is shown in Figure 4.9 and Figure 4.10.

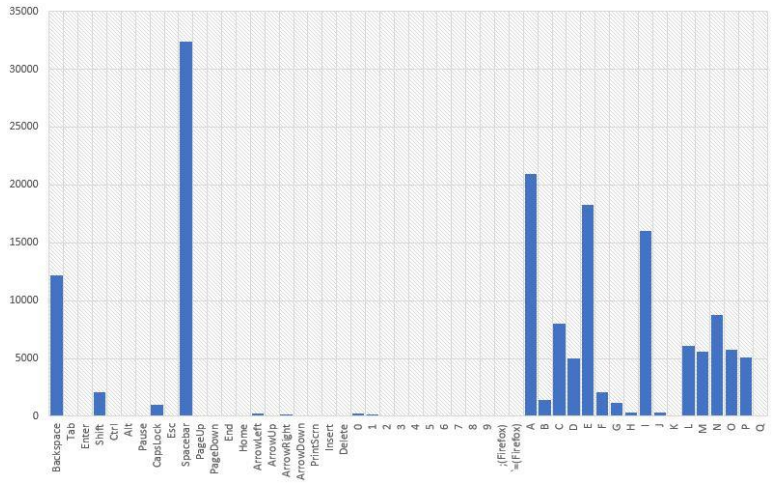


Figure 4.9 The frequency with which the first 50 keys appeared in the text

In the Figure 4.9 appear the first 50 keys, in ascending order of the Key Code, and in the Figure 4.10 appear the other 50 keys, with higher Key Code.

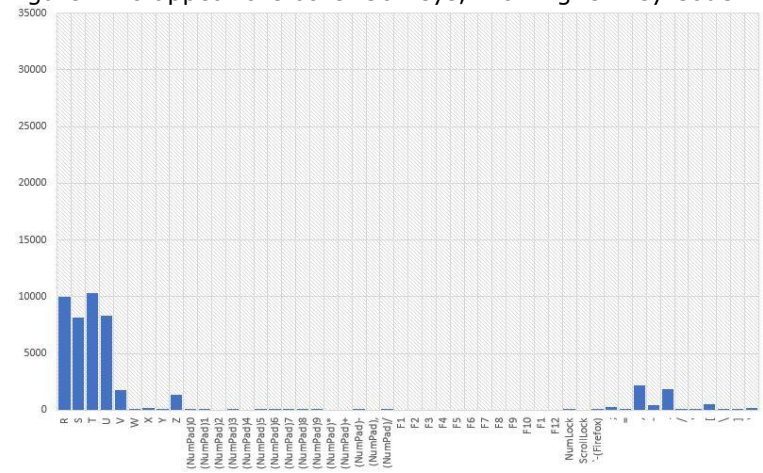


Figure 4.10 The frequency with which the last 50 keys appeared in the text

The key that was pressed most often by users in the experiment was the SPACE key. The SPACE key has been pressed 32,387 times in total. Of the total keys,

it represents the percentage of 16.17%. The next 3 frequently used keys are the vowels A, E and I. A was used 20,965 times and represents 10.47% of the total keys. E has been pressed 18,256 times and represents 9.11% of the total keys. The I key has been pressed 15,994 times and represents 7.99% of the total keys. The BACKSPACE key is also frequently pressed, which has been pressed 12,195 times.

In Table 4.3 are all the keys pressed by users in the order of their frequency in the data set collected.

Table 4.3 The keys typed by users, in descending order of frequency

Key	Key Code	Total number	Percent.
TOTAL		**200299**	
Space bar	32	32387	16,17
A	65	20965	10,47
E	69	18256	9,11
I	73	15994	7,99
Backspace	8	12195	6,09
T	84	10292	5,14
R	82	10030	5,01
N	78	8750	4,37
U	85	8370	4,18
S	83	8210	4,1
C	67	7982	3,99
L	76	6087	3,04
O	79	5780	2,89
M	77	5556	2,77
P	80	5083	2,54
D	68	4982	2,49
,	188	2159	1,08
F	70	2108	1,05
Shift	16	2054	1,03
.	190	1848	0,92
V	86	1811	0,9
B	66	1449	0,72
Z	90	1328	0,66
G	71	1124	0,56
CapsLock	20	988	0,49
[	219	540	0,27
-	189	422	0,21
H	72	363	0,18
J	74	351	0,18
;	186	260	0,13
ArrowLeft	37	230	0,11
0	48	220	0,11
X	88	212	0,11
'	222	200	0,1
Arrow Right	39	188	0,09
]	221	162	0,08
1	49	156	0,08
\	220	98	0,05
Ctrl	17	94	0,05
Enter	13	88	0,04
Alt	18	74	0,04
2	50	74	0,04
K	75	69	0,03
9	57	67	0,03
Y	89	60	0,03
/	191	53	0,03
=	187	52	0,03
3	51	46	0,02
W	87	42	0,02
Delete	46	38	0,02
Arrow Down	40	37	0,02
8	56	36	0,02
5	53	34	0,02
Arrow Up	38	28	0,01

7	55	28	0,01
(NumPad)-	109	28	0,01
6	54	27	0,01
- Firefox	173	20	0,01
'	192	19	0,01
NumLock	144	15	0,01
4	52	14	0,01
Tab	9	10	0,005
; Firefox	59	10	0,005
Q	81	7	0,003
(NumPad)8	104	7	0,003
(NumPad)1	97	6	0,003

= Firefox	61	4	0,002
(NumPad)7	103	4	0,002
(NumPad)/	111	4	0,002
End	35	3	0,001
(NumPad)0	96	3	0,001
PageDown	34	2	0,001
(NumPad)3	99	2	0,001
Home	36	1	0,0005
(NumPad)5	101	1	0,0005
(NumPad)6	102	1	0,0005
(NumPad)9	105	1	0,0005

The most common 30 keys used by users are represented graphically in Figure 4.11. The first 30 keys represent 98.73% of the keys used.

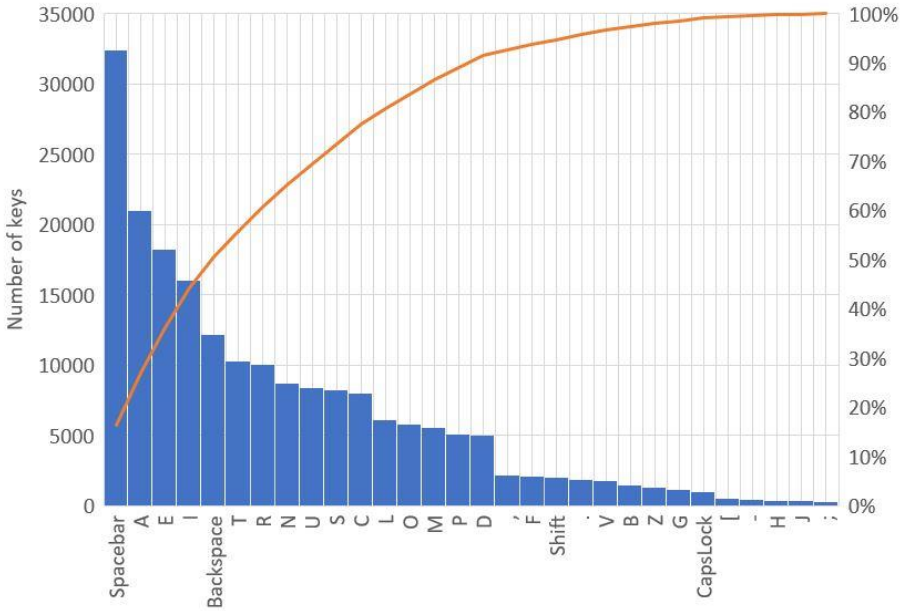


Figure 4.11 Graphical representation of the most used keys

Out of 100 keys that were monitored were not used 23 keys. Table 4.4 shows the keys that were not used in the dataset. Next to each key is their specific key-code.

Table 4.4 The keys not used at all by users

	Key	**KeyCode**	**Total number**	**Percentage**
1	Pause	19	0	0%
2	Esc	27	0	0%
3	PageUp	33	0	0%
4	PrintScrn	44	0	0%
5	Insert	45	0	0%
6	(NumPad)2	98	0	0%
7	(NumPad)4	100	0	0%
8	(NumPad)*	106	0	0%
9	(NumPad)+	107	0	0%
10	(NumPad).	110	0	0%
11	F1	112	0	0%
12	F2	113	0	0%
13	F3	114	0	0%
14	F4	115	0	0%
15	F5	116	0	0%
16	F6	117	0	0%
17	F7	118	0	0%
18	F8	119	0	0%
19	F9	120	0	0%
20	F10	121	0	0%
21	F1	122	0	0%
22	F12	123	0	0%
23	ScrollLock	145	0	0%

Analyzing studies carried out regarding the use of characters in Romanian, the conclusion is that the database collects respect the general rules, this database accurately reproduces the general characteristic of the Romanian language. According to the study conducted in [LAI12], the most used consonants in Romanian are the consonants R and T, while the least used are X and J, except for the letters K, Q, W and Y, which are not specific to the language. The data set falls within these rules, the most used consonants being T and R, and the least used consonants being J, X, K, Q, W and Y.

The distribution of letters of the English alphabet (a-z) in the dataset is shown in Table 4.5.

Table 4.5 The letters, in descending order of frequency

	Key	KeyCode	Total number	Percentage
	TOTAL		**145261**	**72,52%**
1	A	65	20965	14,43%
5	E	69	18256	12,57%
9	I	73	15994	11,01%
20	T	84	10292	7,09%
18	R	82	10030	6,9%
14	N	78	8750	6,02%
21	U	85	8370	5,76%
19	S	83	8210	5,65%
3	C	67	7982	5,49%
12	L	76	6087	4,19%
15	O	79	5780	3,98%
13	M	77	5556	3,82%
16	P	80	5083	3,5%
4	D	68	4982	3,43%
6	F	70	2108	1,45%
22	V	86	1811	1,25%
2	B	66	1449	1%
26	Z	90	1328	0,91%
7	G	71	1124	0,77%
8	H	72	363	0,25%
10	J	74	351	0,24%
24	X	88	212	0,15%
11	K	75	69	0,05%
25	Y	89	60	0,04%
23	W	87	42	0,03%
17	Q	81	7	0,005%

In Figure 4.12 is rendered graphically the distribution of letters used. It can be seen in the graph that the first five letters as frequency, represent more than 50% of the total letters. The letters A, E, I, T and R are the most frequently used in the text by users. These are three vowels and two consonants. They are followed by five consonants: S, C, N, L and M. The most common letters are the letters that a user finds faster on the keyboard and have the shortest keystroke times.

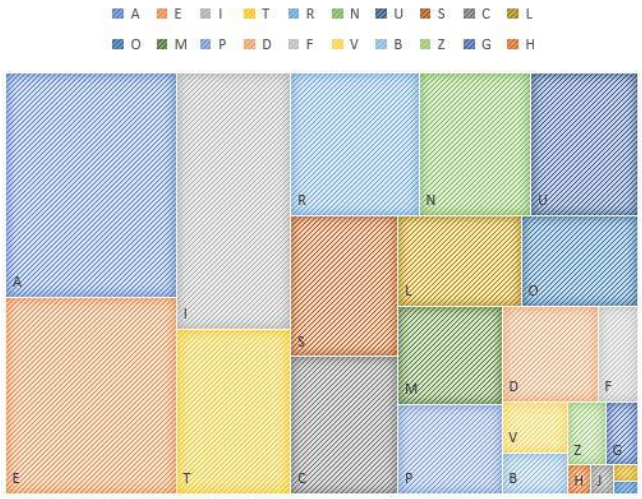


Figure 4.12 Graphical representation of letters frequency

4.7 Differences between users

Each user has his own unique way to type text on the keyboard. This pattern is specific and does not change during a writing session or short term. The typing pattern may change over time or may differ if the same user uses different keyboards. The differences between different users, on the other hand, can be analyzed even visually, as for example in Figure 4.13. The graph shows the typing times for user0001 and user0002 from the database. The graph shows how the differences between the typing times for user0001 are larger, both the average of the times and the standard deviation. Most of the time intervals for user0001 are between 50 and 150 milliseconds. Instead, user0002 has a smaller difference between keystrokes. At user0002 most of the time intervals are in the range of 50-75 milliseconds [IAP21a].

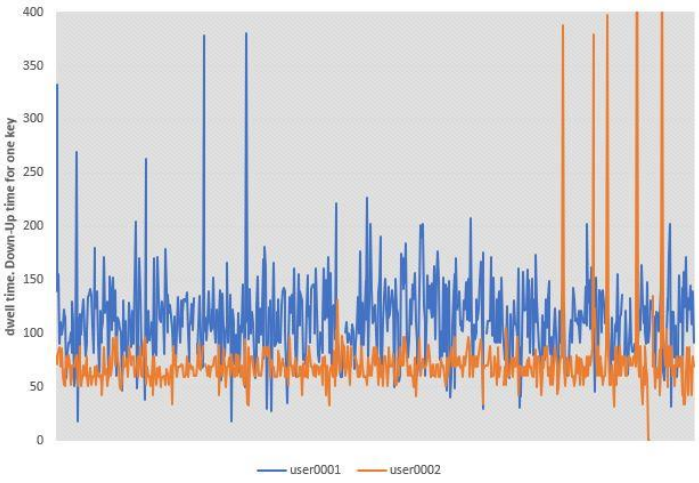


Figure 4.13 Typing pattern from two different users [IAP21a]

Figure 4.14 shows the first 1000 time intervals between two consecutive keys, flight time (UD time). This time interval can also have negative values, while the pressing time of a single key cannot have negative values. A negative value is taken when the second key in a di-graph is pressed before the first key is raised. The figure shows the times for three users: user0001, user0002 and user0003. We can see how user0001 has the most negative time values, while user0003 has the most time values close to 0. The time value can be close to 0 when the second key is pressed exactly when the first he gets up. User0002 has the fewest negative time intervals, even their average being the highest of the time averages of the 3 users analyzed. In the analysis of the typing pattern, both the times when the keys are pressed and the times between two consecutive keys are analyzed.

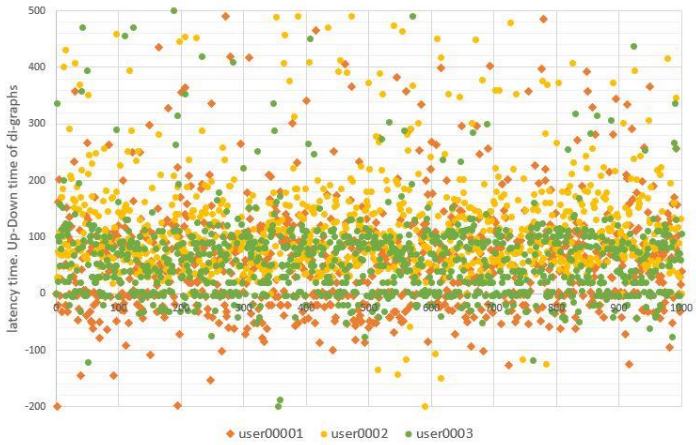


Figure 4.14 Time interval for flight time for three different users

A user's profile in terms of testing can be achieved based on the most frequent time intervals. Figure 4.15 graphically represents the modes of distribution of typing time (DU time) for a number of 7 users: user0006, user0007, user0008, user0009, user0010, user0011 and user0012. The time distribution is a normal distribution, close to a Gaussian distribution or a Laplace distribution. In contrast, both the mean and the standard deviation differ from user to user.

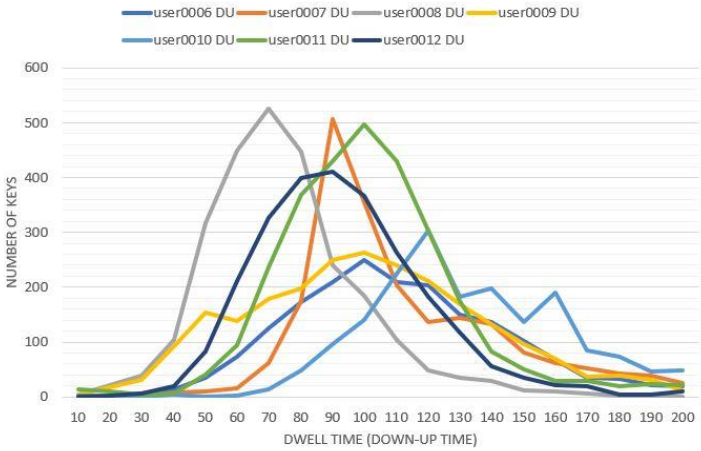


Figure 4.15 Key time distribution for seven different users

The distribution of time intervals between two consecutive keys is represented for a total of five different users in Figure 4.16. It is observed for two users, user0056 and user0059, a maximum of number of key intervals at the value 0 on the graph. Also, user0056 has the most negative intervals. A distribution of time intervals totally different to the other four users has user0055. Times are distributed at higher values. This means that user0055 is typing at a slower pace.

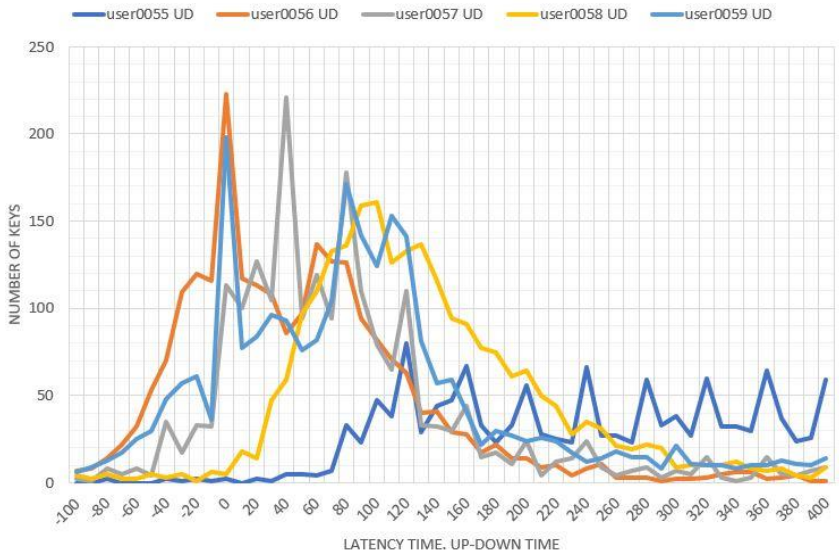


Figure 4.16 The distribution of time intervals between two consecutive keys

The differences between users can be seen from the graphs presented in this chapter. These differences help us to form a pattern of each user and to be able to identify it according to these particular characteristics.

4.8 Conclusions

This chapter was about the data set collected for the present research. The first subchapter 4.1, Platform for collecting data about keyboard typing from 80 volunteers, presented the platform for collecting data about keyboard typing and how data was collected from 80 users. After presenting the platform with the help of which data were collected from users, it proceeded to the analysis of these collected data and presented the particular characteristics, in subchapters 4.2 Analysis of time and key events collected from users, 4.3 Acquisition and initial processing of input data from 80 volunteers (how typing on the keyboard) and 4.4 Analysis of keys collected from users. It showed how they were processed using an algorithm written in the C programming language. The subchapter 4.5, Processing the input data so as to generate a user pattern for each user, presented the structures that store user typing data. The subchapter 4.6, Keys distribution analysis, presented the analysis of the collected keys and the subchapter 4.7, Differences between users, graphically

displayed the typing pattern for different users. This chapter addressed the validation of O1 from the first chapter of this thesis.

The next chapter presents the free-text continuous authentication algorithm based on keystroke dynamics developed for processing the data obtained from the users and presented in this chapter.

5 ALGORITHM DEVELOPMENT FOR KEYSTROKE DYNAMICS AUTHENTICATION

The previous chapter was about the data set collected for the present research. It presented the platform for collecting data about keyboard typing, how data was collected from 80 users and the data analysis.

In this chapter is presented the authentication algorithm based on keystroke dynamics. First of all, the algorithm developed for processing the data obtained from the users is presented. The algorithm simulates user authentication based on keystroke dynamics and measures the obtained performances. In the chapter it is presented the architecture of the algorithm in the subchapter 5.1 The architecture of the authentication algorithm and the structure of the algorithm in the subchapter 5.2 The structure of the authentication algorithm. The development of this algorithm is established by O2.

5.1 The architecture of the authentication algorithm

The architecture of the keystroke dynamic authentication system has two important parts. The first is the system training phase part, part in which users enroll in the system providing data on how to type. In this phase a pattern is created for each user and is stored in the database to be used in the continuous authentication phase. The second part is the continuous authentication phase. In this phase the system continuously verifies the users connected with a valid username and password. Throughout the time a user is logged in to the account, the system takes data from it on the typing mode and continuously compares the resulting pattern with the pattern in the database. As long as there is acceptable similarity between the two patterns the user remains logged in to the system. When the system finds that the two patterns are no longer similar, the one taken from the user logged in to the account and the one from the database, the system generates an alarm signal and the user is removed from the account. He can re-enter the account by re-entering the username and password [IAP21a]. The architecture of the keystroke dynamic authentication system described in this phrase is visually represented in Figure 5.1, the scheme adapted by the author starting from the figure made in the paper [PIL15].

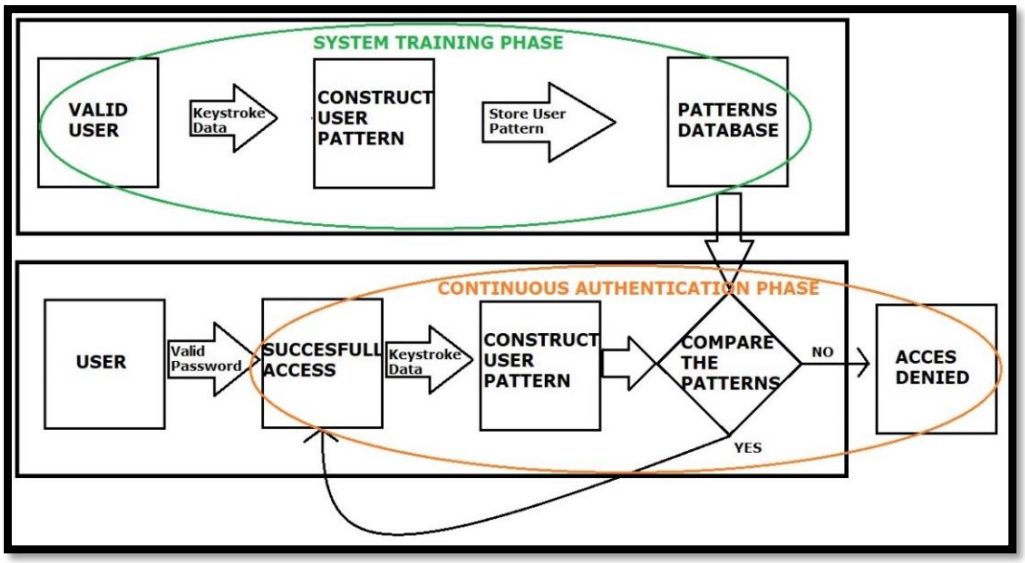


Figure 5.1 The architecture of the keystroke dynamics authentication system [IAP21a]

5.2 The structure of the authentication algorithm

The developed algorithm is presented in Appendix 1 to this thesis. The collection and initial processing of input data are the first steps taken in order to obtain key events of each user. This data represents the input data for the continuous authentication algorithm based on keystroke dynamics. Once the sample size is set, the next step is to divide the user data into key event sequences. The algorithm transforms key events into information about keys and information about diagrams, and then forms the time vectors needed to calculate distances. After the steps described above have been completed, the distances between the vectors are calculated, in order to establish the similarity between two users. Four types of distances are used: Euclidean distance, Manhattan distance, R distance and A distance. With these distances calculated for each user in the database, we proceed to simulate the authentication in the system, in turn by each user in the database. Following the simulation of the authentication in the system, four performance indicators of the algorithm are generated: False Acceptance Rate (FAR), False Rejection Rate (FRR), True Acceptance Rate (TAR) and True Rejection Rate (TRR). Based on these, the Equal Error Rate (EER) can be calculated, the main indicator of the performance of the algorithms used in this thesis. Also, to view the performances, two graphs are generated: FAR and FRR chart and ROC curve.

In the Figure 5.2 are presented the structure of the authentication algorithm based on keystroke dynamics.

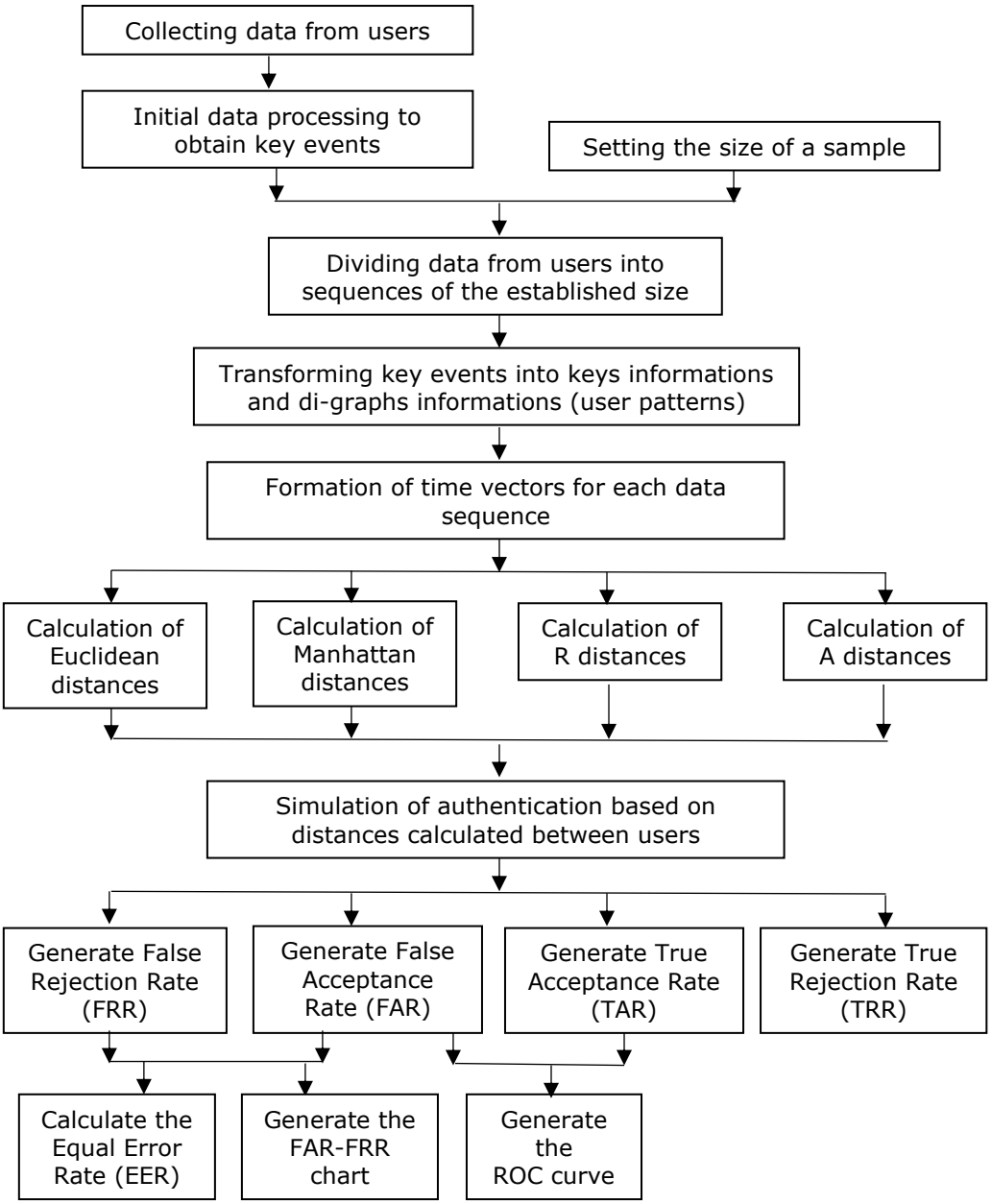


Figure 5.2 The structure of the authentication algorithm based on keystroke dynamics

5.3 Conclusions

In this chapter was presented the authentication algorithm based on keystroke dynamics. First of all, the algorithm developed for processing the data obtained from the users was presented. The algorithm simulates user authentication based on keystroke dynamics and measures the obtained performances. In this chapter was presented the architecture of the algorithm, in the subchapter 5.1 The architecture of the authentication algorithm, and the structure of the algorithm, in the subchapter 5.2 The structure of the authentication algorithm. The development of this algorithm was established by O2.

The next chapter, Chapter 6 - Experiments and results, will presents the simulation with the algorithm presented in this chapter various possibilities of accessing accounts by users who provided the data for research purposes. The next chapter will present a series of experiments that are performed that aim to obtain better results in terms of the degree of success with which authentication based on keystroke dynamics is done.

6 EXPERIMENTS AND RESULTS - SIMULATION OF SYSTEM AUTHENTICATION BY GENUINE USERS OR IMPOSTORS

In the previous chapter was presented the authentication algorithm based on keystroke dynamics, its architecture and its structure. If in the previous chapters was presented the path validation O1 and O2, in this chapter it is followed the validation of the other two objectives of the present thesis: O3 and O4.

In this chapter it is presented a series of experiments performed to measure the performance of the written algorithm for the purpose of this research and to analyze the results obtained. Gradually, experiments with the keystroke time of a single key, in the subchapter 6.1, and experiments with di-graphs, in the subchapter 6.2, are presented. Both in the analysis of the characteristics with a single key and with a di-graph, the degree of Equal Error Rate (EER) is calculated in order to appreciate the performances of the algorithms. The results are presented in the case of experiments using Euclidean distance (in the subchapters 6.1.1 and 6.2.3), Manhattan distance (in the subchapters 6.1.2 and 6.2.4), R distance (in the subchapter 6.1.3) and A distance (in the subchapters 6.1.4 and 6.2.5). The chapter also investigates, in the subchapter 6.1.5 The sample size, the differences in performance if the pattern is built for each user with various sample sizes, starting from 200 key events / pattern and up to 3000 key events / pattern. At the end of the chapter, following all the experiments performed and presented, the author proposes, in the subchapter 6.4, Proposing new metrics for calculating distances between users, the modification of two metrics obtaining new metrics for calculating the distances between two vectors that have higher performances than the classical calculation methods. For the two new metrics, the performances obtained in terms of Equal Error Rate (EER) are presented. By proposing these metrics, O3 is validated. It also proposes, in the subchapter 6.5, Proposed user pattern, a structure for retaining a user's pattern, a structure that takes up small memory and requires little time to perform all the necessary calculations in the algorithms. By proposing the user pattern, O4 is validated. In the end of the Chapter, in the subchapter 6.6 Comparison of the related works, the performances obtained in the present research are compared with those obtained by other authors in their researches.

6.1 Experiments with the keystroke time of a single key

Within the scientific research for this thesis, the author has created a database with a number of 410,633 key events from 80 users.

In the first experiment it was divided the data obtained from 80 users into sets of 1000 key events, obtaining 370 data sequences. From each set it was made a pattern that indicates the calculations of the average keystroke times, made for each key separately. It was taking into account only the keys that contain a letter (a-z), thus obtained for each user a vector of 27 real numbers, each number representing the average of the pressing times of that letter [IAP21a].

6.1.1 Experiments with Euclidian distance

To calculate the similarities between two vectors (two distinct users or two vectors obtained from the text of the same user) it was applied the calculation of the Euclidean distance. In this way it was acquired values between 0 (for the same vector, as expected) and 642.36, the largest distance calculated between the vectors. Under these conditions, it was simulated a number of 136,161 attempts to access the 370 accounts by the other 369 users.

Under these circumstances, it was considered that the user has successfully accessed the account if the Euclidean distance (calculated between the vector resulting from the user's key events and the vector resulting from the key events of the account to be accessed) was less than a certain threshold. In case it was higher than the certain threshold, it was considered that he failed to access the account. *Code 6.1* shows the function that calculate Euclidian distance.

Code 6.1 calculateEuclidianDistance() function

```
int calculateEuclidianDistances()
{
  int i,j,k;
  float dist;
  for(i=0;i<nPatterns;i++)
  {
    for(j=i;j<nPatterns;j++)
    {
      dist=0;
      for(k=65;k<=90;k++)
        if(patterns[i].distributions[k].mean!=0 &&
patterns[j].distributions[k].mean!=0)
          {
           dist=dist+pow(patterns[i].distributions[k].mean-
patterns[j].distributions[k].mean, 2);
          }
      dist=sqrt(dist);
      patterns[i].distance[j]=dist;
      patterns[j].distance[i]=dist;
    }
  }
  return 1;
}
```

In the end of the first experiment, it was calculated the performance indicators established in the literature: False Acceptance Rate (FAR), False Rejection Rate (FRR), TAR (True Acceptance Rate), TRR (True Rejection Rate), ERR (Equal Error Rate). So

as to calculate these indicators, the value of the threshold below which the account cannot be accessed went from 0 to 643. The results are shown in Figure 6.1 and Figure 6.2.

The ERR (Equal Error Rate) value calculated for this experiment is 17.5%, at a limit imposed at a maximum distance of 76. The ERR value is at the intersection, on the graph of FAR and FRR.

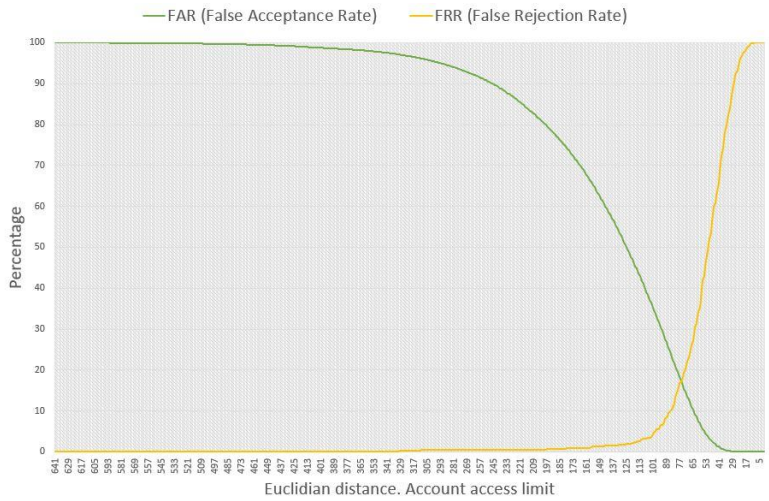


Figure 6.1 FAR and FRR for Euclidian distance

To calculate FAR and FRR values was used function calculateFARandFRR() from *Code 6.2*.

Code 6.2 calculateFARandFRR() function

```
int calculateFARandFRR()
{
  int i,j,k, sw=0, iERR=0;
  float ERR=0, FAR=0, FRR=0, TAR=0, TRR=0, TA=0, TR=0, FA=0, FR=0;
  for(i=1;i<1000;i++)
  {
    TA=0, TR=0, FA=0, FR=0;
    for(j=0;j<nPatterns;j++)
    {
      for(k=j+1;k<nPatterns;k++)
      {
        if(patterns[j].user[4]==patterns[k].user[4] &&
patterns[j].user[5]==patterns[k].user[5] &&
patterns[j].user[6]==patterns[k].user[6] &&
patterns[j].user[7]==patterns[k].user[7])
        {
          if(patterns[j].distance[k]<i)
            TA++;
          else
            FR++;
        }
```

```
        else
        {
          if(patterns[j].distance[k]<i)
            FA++;
          else
            TR++;
        }
      }
    }
    FAR=FA/(FA+TR)*100;
    FRR=FR/(FR+TA)*100;
    TAR=TA/(TA+FR)*100;
    TRR=TR/(TR+FA)*100;

    if(FAR>FRR && sw==0)
    {
      ERR=FAR;
      iERR=i;
      sw=1;
    }
    printf("%d\t%.2f\t%.2f\t%.2f\t%.2f\n",i,FAR,FRR,TAR,TRR);
  }
  printf("ERR= %.2f in %d",ERR, iERR);
  return 1;
}
```

In the Figure 6.2 is the ROC curve resulted for Euclidian distance algorithm. It shows the relation between True Acceptance Rate (TAR) and False Acceptance Rate (FAR).

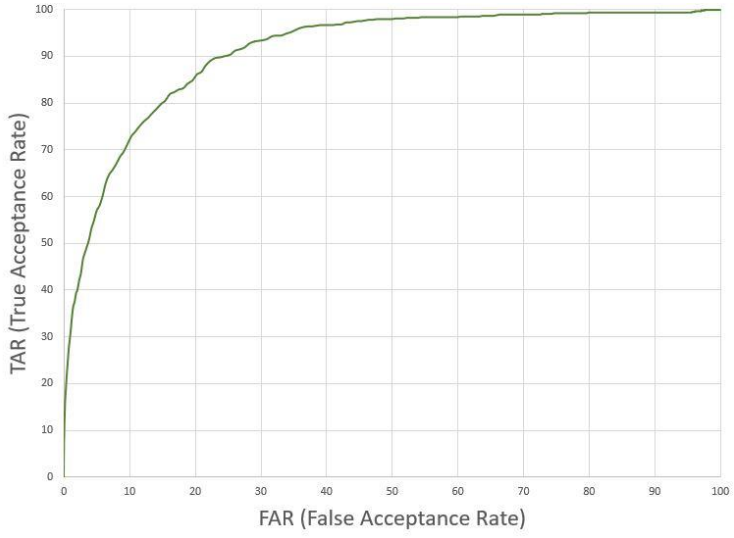


Figure 6.2 ROC Curve for Euclidian distance

6.1.2 Experiments with Manhattan distance

In this section, to calculate the similarities between two vectors (two distinct users or two vectors obtained from the text of the same user) it was applied the calculation of the Manhattan distance. In this way it was acquired values between 0 (for the same vector, as expected) and the largest distance calculated between the vectors. Under these conditions, it was simulated a number of 136,161 attempts to access the 370 accounts by the other 369 users.

Under these circumstances, it was considered that the user has successfully accessed the account if the Manhattan distance (calculated between the vector resulting from the user's key events and the vector resulting from the key events of the account to be accessed) was less than a certain threshold. In case it was higher than the certain threshold, it was considered that he failed to access the account. *Code 6.3* shows the function that calculate Manhattan distance.

Code 6.3 The function that calculates the Manhattan distances

```
float calculateManhattanDistances()
{
  int i,j,k;
  float dist,max=0;
  for(i=0;i<nPatterns;i++)
  {
    for(j=i;j<nPatterns;j++)
    {
      dist=0;
      for(k=65;k<=90;k++)
        if(patterns[i].distributions[k].mean!=0 &&
patterns[j].distributions[k].mean!=0)
          {
           dist=dist+fabs(patterns[i].distributions[k].mean-
patterns[j].distributions[k].mean);
          }
      patterns[i].distance[j]=dist;
      patterns[j].distance[i]=dist;
      if(max<dist)
            max=dist;
    }
  }
  return max;
}
```

It was calculated the performance indicators established in the literature: False Acceptance Rate (FAR), False Rejection Rate (FRR), TAR (True Acceptance Rate), TRR (True Rejection Rate), ERR (Equal Error Rate). So as to calculate these indicators, the value of the threshold below which the account cannot be accessed went from 0 to 643. The results are shown in Figure 6.3 and Figure 6.4.

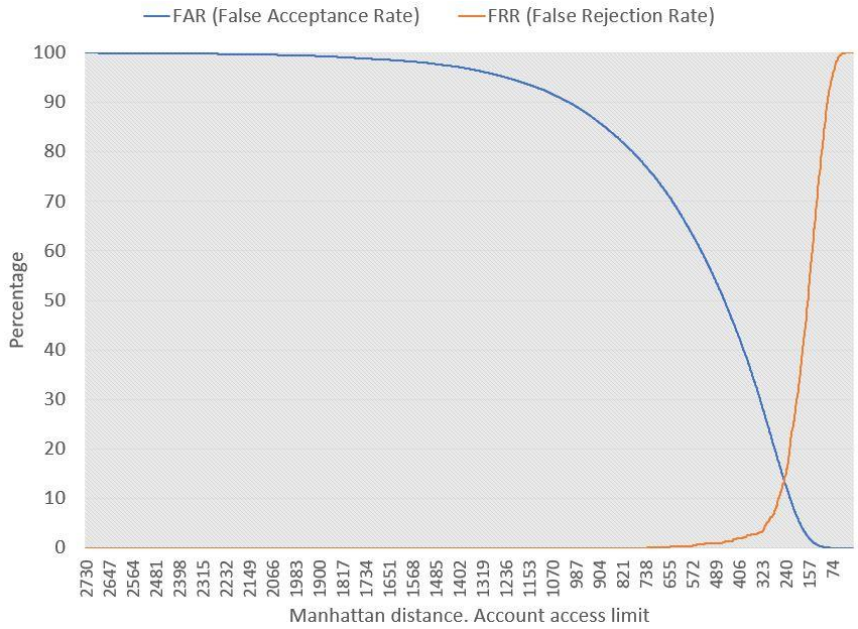


Figure 6.3 FAR and FRR for Manhattan distance

In the Figure 6.4 is the ROC curve resulted for Euclidian distance algorithm. It shows the relation between True Acceptance Rate (TAR) and False Acceptance Rate (FAR).

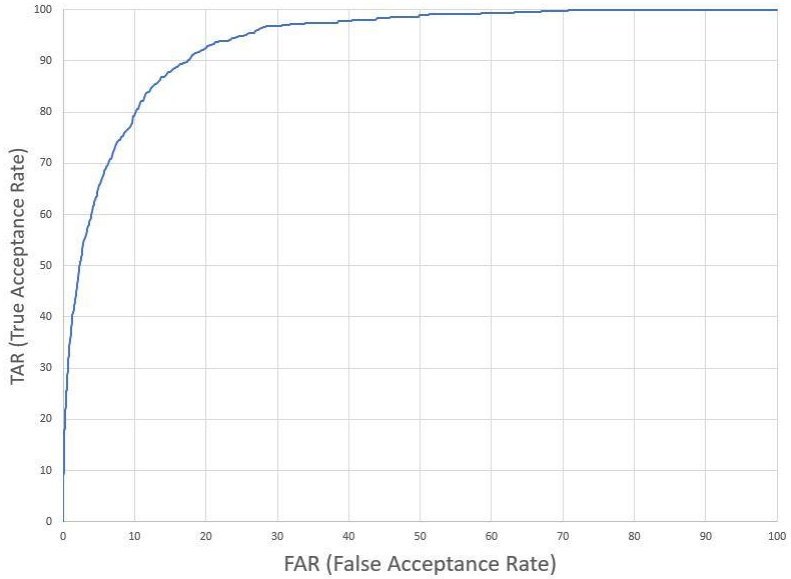


Figure 6.4 ROC curve Manhattan distance

The ERR (Equal Error Rate) value calculated for this experiment was 13.78%, at a limit imposed at a maximum Manhattan distance of 246. The ERR value was at the intersection, on the graph, of FAR and FRR.

In the Figure 6.5 are the ROC curves for Euclidian (green) and Manhattan (blue) distances

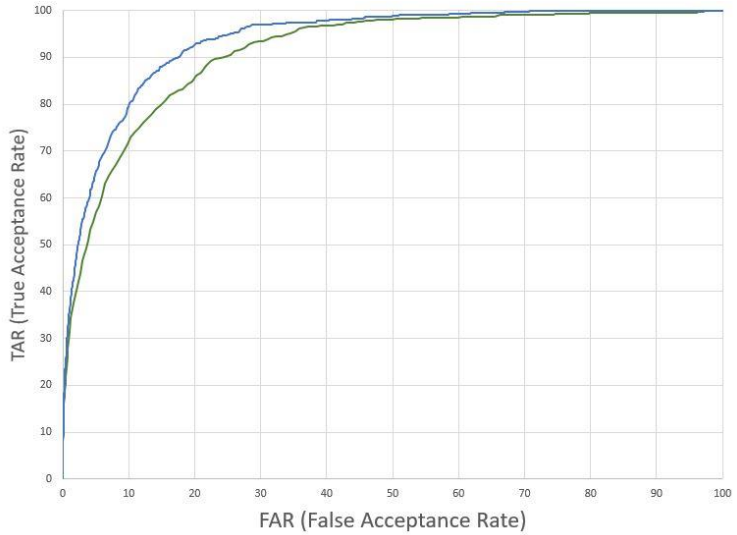


Figure 6.5 ROC curves for Euclidian (green) and Manhattan (blue) distances

6.1.3 Experiments with R distance

In this section, to calculate the similarities between two vectors (two distinct users or two vectors obtained from the text of the same user) it was applied the calculation of the R distance. In this way it was acquired values between 0 (for the same vector, as expected) and the largest distance calculated between the vectors. Under these conditions, it was simulated a number of 136,161 attempts to access the 370 accounts by the other 369 users.

Under these circumstances, it was considered that the user has successfully accessed the account if the R distance (calculated between the vector resulting from the user's key events and the vector resulting from the key events of the account to be accessed) was less than a certain threshold. In case it was higher than the certain threshold, it was considered that he failed to access the account. *Code 6.4* shows the function that calculate R distance.

Code 6.4 The function that calculates the R distances

```
float calculateRdistances()
{
  int i,j,k,k2,sw;
  float dist,max=0;
  distribution aux;

  for(i=0;i<nPatterns;i++)
```

```
  {
   for(k=0;k<223;k++)
   {
    patterns[i].distributions[k].keyCode=k;
   }
   patternsR[i]=patterns[i];
   sw=0;
   while(sw==0)
   {
    sw=1;
    for(j=65;j<=90;j++)
    {
     if(patternsR[i].distributions[j].mean < patternsR[i].distributions[j+1].mean)
     {
      aux=patternsR[i].distributions[j];
      patternsR[i].distributions[j]=patternsR[i].distributions[j+1];
      patternsR[i].distributions[j+1]=aux;
      sw=0;
     }
    }
   }
  }

  for(i=0;i<nPatterns;i++)
  {
   for(j=i;j<nPatterns;j++)
   {
    dist=0;
    for(k=65;k<=90;k++)
     for(k2=65;k2<=90;k2++)

if(patternsR[i].distributions[k].keyCode==patternsR[j].distributions[k2].keyCode)
       {
        if(patternsR[i].distributions[k].mean!=0 &&
patterns[j].distributions[k2].mean!=0)
        {
         dist=dist+abs(k2-k);
        }
        break;
       }
    patternsR[i].distance[j]=dist;
    patternsR[j].distance[i]=dist;
    if(max<dist)
     max=dist;
   }
  }
  return max;
}
```

It was calculated the performance indicators established in the literature: False Acceptance Rate (FAR), False Rejection Rate (FRR), TAR (True Acceptance Rate), TRR (True Rejection Rate), ERR (Equal Error Rate). So as to calculate these

indicators, the value of the threshold below which the account cannot be accessed went from 0 to 643. The results are shown in Figure 6.6 and Figure 6.7.

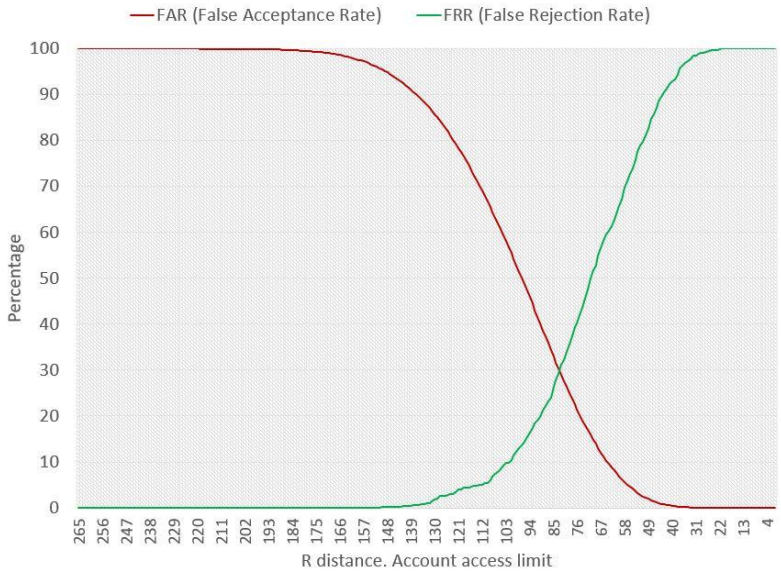


Figure 6.6 FAR and FRR for R distance

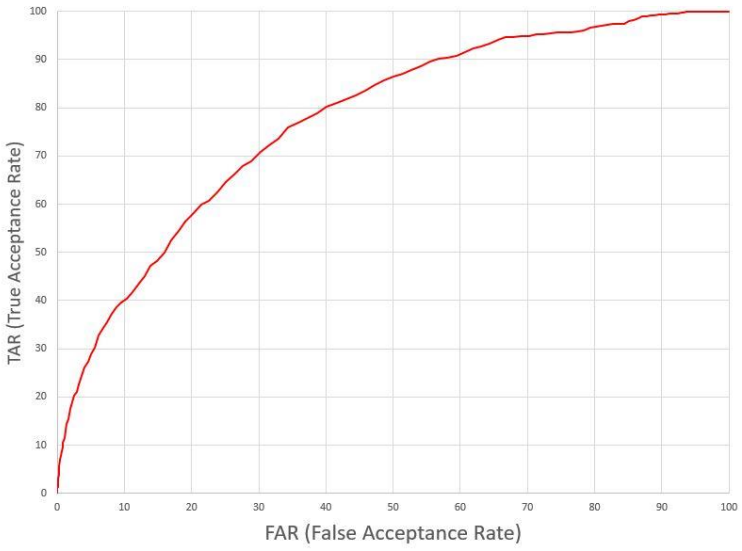


Figure 6.7 ROC curve for R distnace

The ERR (Equal Error Rate) value calculated for this experiment is 30.32%, at a limit imposed at the maximum R distance of 83. The ERR value is at the intersection, on the graph, of FAR and FRR.

In the Figure 6.8 are the ROC curves for Euclidian (green), Manhattan (blue) and R (red) distances.

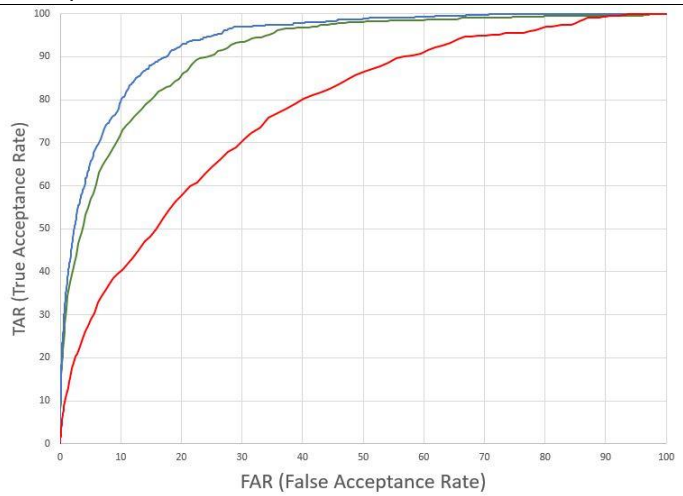


Figure 6.8 ROC curves for Euclidian (green), Manhattan (blue) and R (red) distances

6.1.4 Experiments with A distance

In this section, to calculate the similarities between two vectors (two distinct users or two vectors obtained from the text of the same user) it was applied the calculation of the A distance. In this way it was acquired values between 0 (for the same vector, as expected) and the largest distance calculated between the vectors. Under these conditions, it was simulated a number of 136,161 attempts to access the 370 accounts by the other 369 users.

Under these circumstances, it was considered that the user has successfully accessed the account if the A distance (calculated between the vector resulting from the user's key events and the vector resulting from the key events of the account to be accessed) was less than a certain threshold. In case it was higher than the certain threshold, it was considered that he failed to access the account. *Code 6.5* shows the function that calculate A distance.

Code 6.5 The function that calculates the A distances

```
float calculateADistances()
{
  int i,j,k;
  float dist,max=0, t=1.25;
  for(i=0;i<nPatterns;i++)
  {
    for(j=i;j<nPatterns;j++)
    {
      dist=0;
      for(k=65;k<=90;k++)
        if(patterns[i].distributions[k].mean!=0 &&
patterns[j].distributions[k].mean!=0)
        {
          if(patterns[i].distributions[k].mean > patterns[j].distributions[k].mean)
            {
```

```
            if(patterns[i].distributions[k].mean/patterns[j].distributions[k].mean < t)
              dist++;
           }
          else
          {
           if(patterns[j].distributions[k].mean/patterns[i].distributions[k].mean < t)
              dist++;
          }
         }
     dist=1-dist/27;
     patterns[i].distance[j]=dist;
     patterns[j].distance[i]=dist;
     if(max<dist)
           max=dist;
   }
  }
  return max;
}
```

It was calculated the performance indicators established in the literature: False Acceptance Rate (FAR), False Rejection Rate (FRR), TAR (True Acceptance Rate), TRR (True Rejection Rate), ERR (Equal Error Rate). So as to calculate these indicators, the value of the threshold below which the account cannot be accessed went from 0 to 643. The results are shown in Figure 6.9 and Figure 6.10.

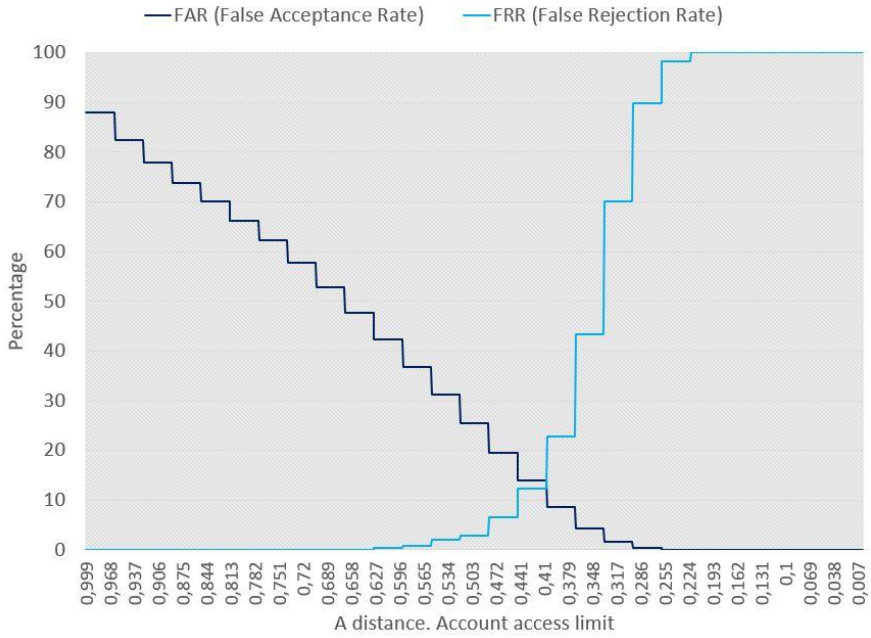


Figure 6.9 FAR and FRR for A distance

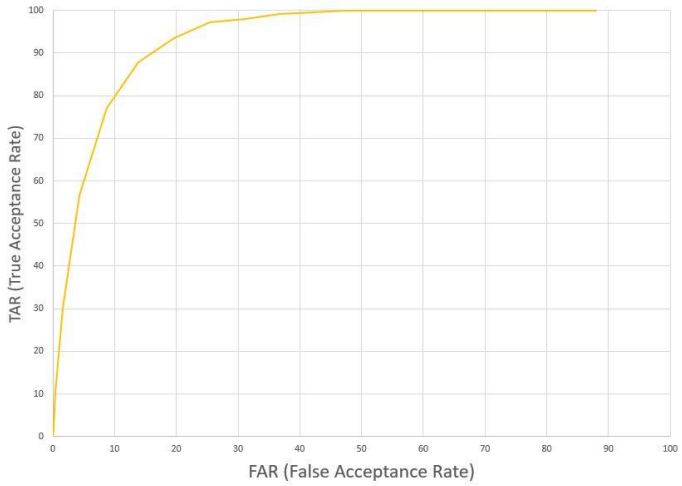


Figure 6.10 ROC curve for A distance

The ERR (Equal Error Rate) value calculated for this experiment is 13.90%, at a limit imposed at the maximum A distance of 0,41. The ERR value is at the intersection, on the graph, of FAR and FRR.

In the Figure 6.11 are the ROC curves for Euclidian (green), Manhattan (blue) and R (red) and A (yellow) distances.

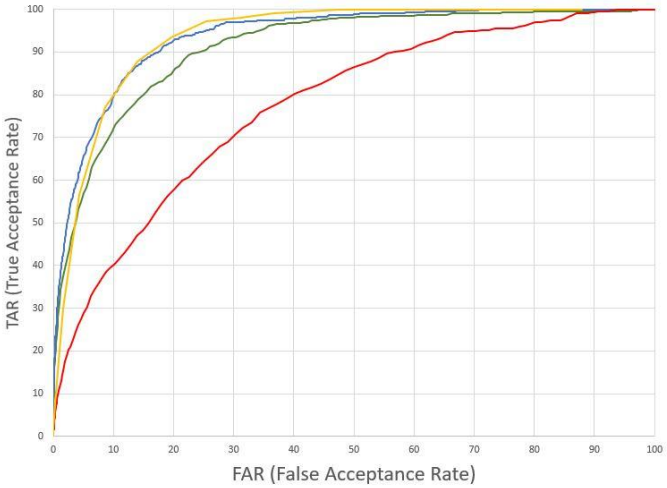


Figure 6.11 ROC curves for Euclidian (green), Manhattan (blue), R (red) and A (yellow) distances

It depends on what value we choose for the constant t. The authors of [KIL09] concluded that the best value for t is 1.25. We performed the first test with t = 1.25 according to the conclusions of the study that proposes A distance. After gradually changing the coefficient t, on the interval 1-3, with a step of 0.01, a more efficient

value for t was identified in point 1.13. At this value of t, the calculated value of EER is minimal, of 11.85%. In Table 6.1 are the values of the EER at each increment of the coefficient t. Above 1.5, the higher the coefficient t, the higher the EER rate.

Table 6.1 EER (Equal Error Rate) value at different value of t coefficient

t coefficient	**EER**
1,01	29,73
1,02	21,93
1,03	18,31
1,04	15,86
1,05	**14,07**
1,06	20,03
1,07	17,73
1,08	15,74
1,09	**13,86**
1,1	**12,25**
1,11	15,66
1,12	**13,68**
1,13	**11,83**
1,14	**14,57**
1,15	**12,65**
1,16	15,12
1,17	**12,79**
1,18	15,16
1,19	**12,64**
1,2	**14,84**
1,21	16,9
1,22	**13,8**
1,23	15,7
1,24	17,64
1,25	**13,9**
1,26	15,71
1,27	17,56
1,28	19,35
1,29	21,18
1,3	16,23
1,31	17,81
1,32	19,47
1,33	21,07
1,34	22,69
1,35	24,33
1,36	25,95
1,37	27,55
1,38	29,1
1,39	21,39
1,4	22,74
1,41	24,08
1,42	25,36
1,43	26,72
1,44	28,17
1,45	29,48
1,46	30,87
1,47	32,16
1,48	33,5
1,49	34,74

In Figure 6.12 is a graph that contain the EER (Equal Error Rate) value at different value of t coefficient.

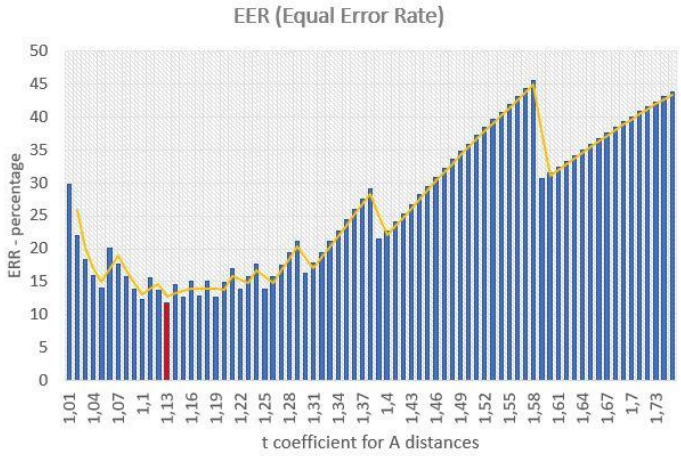


Figure 6.12 EER (Equal Error Rate) value at differrent value of t coefficient

In the Figure 5.13 is ROC curve for A distance with the best t coefficient, t=1.13.

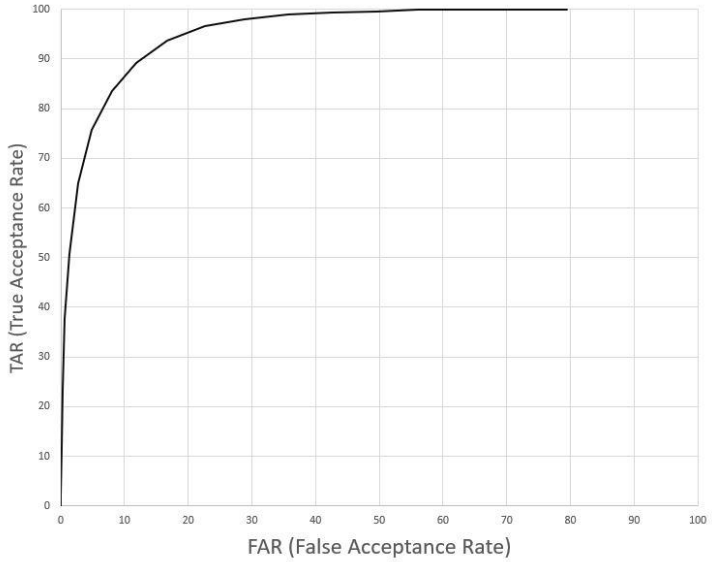


Figure 6.13 ROC curve for the best t coefficient

In the Figure 6.14 are the ROC curves for Euclidian (green), Manhattan (blue) and R (red), A with t=1.25 (yellow), and A with t=1.13 (black) distances.

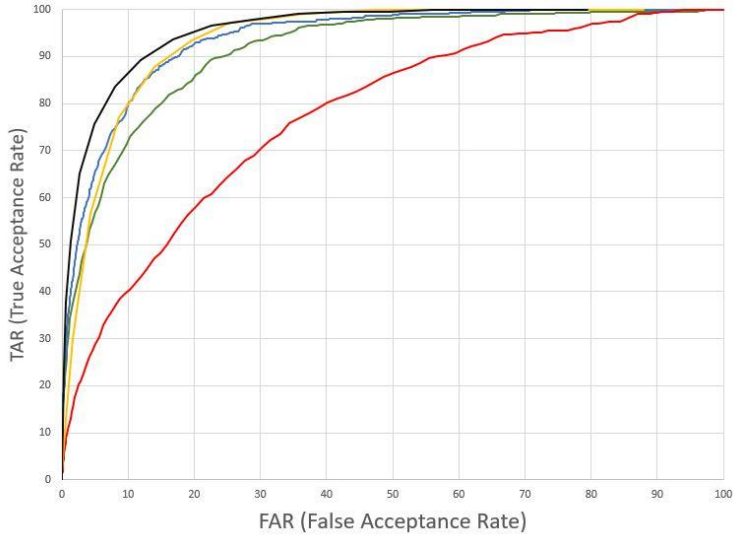


Figure 6.14 ROC curves for Euclidian (green), Manhattan (blue), R (red), A (t=1.25) (yellow) and A (t=1.13) (black) distances

6.1.5 The sample size

With the algorithm presented in *Code 6.6*, the data obtained from the 80 uses could be divided into samples of different sizes in order to test the performances of the algorithms at various sample sizes.

Code 6.6 The algorithm that make the sample

```
int main(void)
{
  int i,j;
  char c,user[20], fileUser[20], line[30], localKeyEventsList[150000];

  f=fopen("keyEventsListAllUsers.txt","r");
  g=fopen("keyEventsListSegmentationAllUser.txt","w");

  do{
       fscanf(f,"%s",user);
       if(strcmp(user,"-1")==0)
          break;
       fscanf(f,"%c",&c);
       if(user[0]=='u' && user[1]=='s' && user[2]=='e' && user[3]=='r')
       {
         j=1;
         do
         {
           strcpy(localKeyEventsList,"");
           for(i=0;i<LENGTH;i++)
           {
             fgets(line,30,f);
             if(strcmp(line,"-1\n")==0)
               break;
             strcat(localKeyEventsList,line);
           }
           if(i==LENGTH)
           {
             if(j<10)
               {
                 fprintf(g,"%s.0%d\n%s-1\n",user,j,localKeyEventsList);
                 printf("%s.0%d\n",user,j);
               }
             else
               {
                 fprintf(g,"%s.%d\n%s-1\n",user,j,localKeyEventsList);
                 printf("%s.%d\n",user,j);
               }
           }
           else
           {
             break;
           }
           j++;
```

```
      }while(i==LENGTH);
    }
  }while(strcmp(user,"-1")!=0);
  fprintf(g,"-1");
 return 0;
}
```

The first tests performed and presented up to this paragraph were performed at a sample size length of 500 keys, ie at 1000 key events. Figure 6.15 shows the results obtained according to the Equal Error Rate (EER). It is observed that the performance of the algorithms increases with the increase of the sample size length from which results pattern for the user [IAP21b].

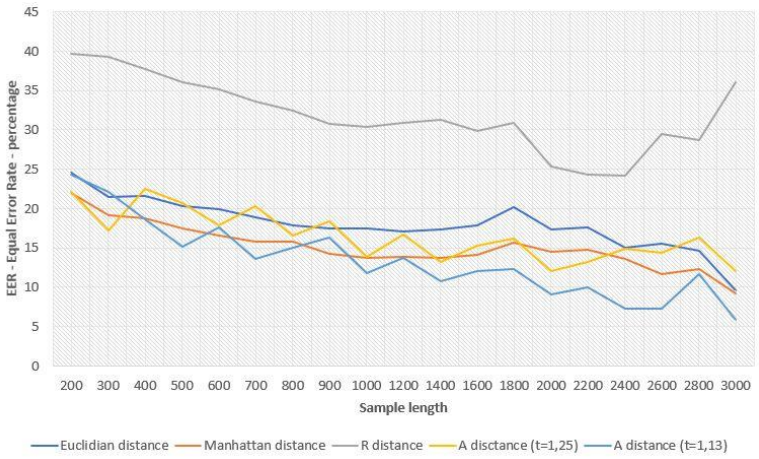


Figure 6.15 EER decreases as the sample size is larger [IAP21b]

In the case of all distances calculated in the experiment, a trend of increasing performance is seen in direct proportion to the increase of sample size. EER values are recorded in Table 5.2. These vary from 39.68%, in the case of R distance at 200 key events (100 keys), to 5.83% in the case of A distance, t = 1.13 and 3000 key events (1500 keys).

Table 6.2 EER values depending on the sample size and distance used

Sample size (key events)	**200**	**500**	**1000**	**1600**	**2000**	**2600**	**3000**
Euclidian distance	24,53	20,31	17,5	17,86	17,31	15,59	**9,59**
Manhattan distance	22,03	17,52	13,78	14,17	14,5	11,71	**9,16**
R distance	39,68	36,01	30,32	29,89	**25,37**	29,48	36
A distance (t=1,25)	22,12	20,68	13,9	15,24	12,11	14,42	**12,02**
A distance (t=1,13)	24,31	15,2	11,83	12,06	9,07	7,28	**5,83**

Next, tests will be performed at a sample length of 2000 key events. That means a sample size of 1000 keys.

Tests have been done to see if ERR improves with the modification of various variables. Instead of calculating the distances according to the average DU on each letter, the same distances were calculated but according to the average of standard deviations for each letter. The results were much poorer. Likewise, the results remained weaker when the sum of the 2 distances was made, the first based on the average of the DU times and the second based on the average of the standard deviations.

Another set of tests were done using the UD times of numbers and letters. Previous tests used only the 27 letters of the English alphabet, omitting the other characters. The success rate of the tests in this case was lower than if only the letters of the alphabet were used. In contrast, the difference was not as large as if the standard deviation were used as a benchmark instead of the mean time.

The results obtained were weaker even if the times recorded on all system keys were used, and not only on the letters of the English alphabet.

It was noticed that different results are obtained depending on the number of characters analyzed when calculating the distances. In the following tests, the number of letters on which the analysis was performed was further limited. The 5 most used letters were selected first. The 5 most used letters are: A, E, I, T and R. In these cases, there have been improvements to the methods using Euclidian Distance and Manhattan Distance, instead of worsening the results for A Distance.

The 10 most used letters were then selected, according to statistics. The 10 most used letters are: A, E, I, T, R, N, U, S, C and L. In this case much better results are obtained for Euclidian Distance and Manhattan Distance.

The most used 15 letters are: A, E, I, T, R, N, U, S, C, L, O, M, P, D and F. In this case much better results are obtained for Euclidian Distance if Manhattan Distance. Instead, for Manhattan Distance, the best EER rate was obtained at this sample size. Figure 6.16 shows the FAR and FRR graph.

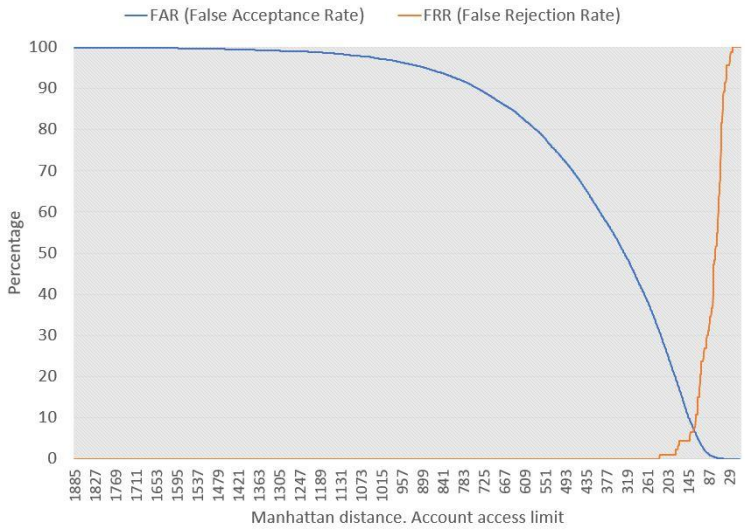


Figure 6.16 FAR and FRR for Manhattan distance for the first 15 letters

It can be seen on the graph in Figure 6.16 that the EER has a value of 7.74%, the lowest obtained in this experiment at this sample size. And in the generated curve ROC graph it is observed that the algorithm is more efficient in this case. Figure 6.17 generates the ROC curve graph for Manhattan Distance that takes into account the 15 most used letters.

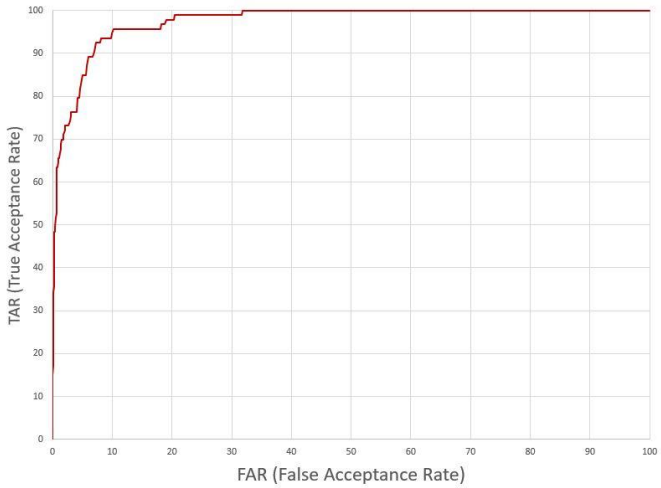


Figure 6.17 ROC curve for Manhattan distance for the first 15 letters

Overlapping the ROC curve chart over the other ROC curves, it can be seen that it has the best performance in all points on the chart. Figure 6.18 shows the performance of these parameters.

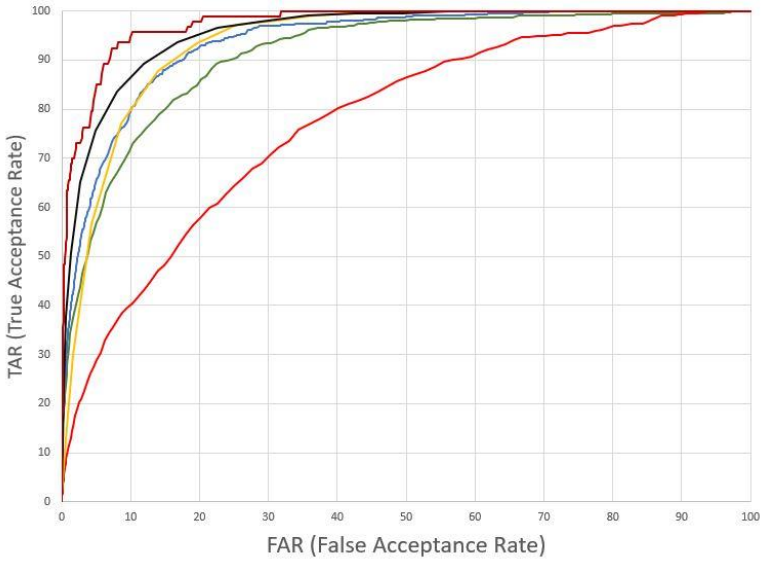


Figure 6.18 ROC curves with different distances

Another approach was to analyze standard deviation on each key, for each user. The lower the standard deviation, the more it means that the user types the letter in the same way. In order to have an image on the entire database, standard deviation of all users was made. This time, the letters that fall into the lowest standard deviation have been selected. In Table 6.3 are the average values of the standard deviation on each letter.

Table 6.3 Standard deviation for each letter

Key	Standard deviation
I	1,03
E	1,12
A	1,15
U	1,27
T	1,3
N	1,33
O	1,42
L	1,57
M	1,75
R	1,75
S	1,86
P	1,9
C	2,24
D	2,37
G	3,23
V	3,33
F	3,45
B	4,09
Z	5,32
H	5,68
J	6,27
X	7,57
K	8,39
Y	8,9
Q	12,37
W	12,53

By modifying the test parameters, the EER performances presented in Table 6.4 were obtained. Analyzing the data from the two tables, it was concluded that the best results can be obtained either with the algorithm using A Distance, with t = 1.13, and comparing the times obtained for all keys collected, or the algorithm using Manhattan Distance or Euclidean with the most common 14 letters.

Table 6.4 EER values in different conditions

	All Keys	Only Letters (a-z)	Only Letters (a-z) and digits (0-9)
Euclidian distance	25,82%	18,53%	17,31%
Manhattan distance	17,33%	14,28%	14,50%
A disctance (t=1,25)	11,67%	12,11%	13,85%
A distance (t=1,13)	**7,47%**	9,07%	9,85%

Table 6.5 presents the test results for the 4 algorithms used, on the first line of the table, in various scenarios. This time the tests were done only on the keys which are letters (a-z), omitting the other keys (signs, spaces, numbers, etc.). Significant improvements are observed when only certain letters are selected. At the first test, all 27 letters were included in the test.

A first series of tests were made for the most used in keys by users. There is a major improvement in the EER indicator for Euclidean Distance and Manhattan Distance when only the first 14 letters are used (A, E, I, T, R, N, U, S, C, L, O, M, P and D). The best performance was obtained at the EER value of 6.71%.

The second series of tests were done for letters that have the smallest standard deviation.

There is a major improvement in the EER indicator for Euclidean Distance and Manhattan Distance, as in the first test, when only the first 14 letters are used. The best performance was obtained at the EER value of 6.80%.

Table 6.5 EER values in different conditions

	Euclidian distance	Manhattan distance	A disctance (t=1,25)	A distance (t=1,13)
All Letters (a-z)	18,53%	14,28%	12,11%	9,07%
The most frequent keys (first 15 keys)	8,1%	7,74%	13,26%	11,2%
The most frequent keys(14)	**6,71%**	**7,13%**	14,37%	8,57%
The most frequent keys(13)	8,03%	7,78%	11,08%	10,93%
The most frequent keys(12)	7,85%	7,67%	8,64%	7,70%
The most frequent keys(11)	7,84%	8,18%	9,53%	9,10%
The most frequent keys(10)	8,14%	8,2%	10,01%	10,86%
The most frequent keys(9)	8,77%	8,64%	11,21%	13,73%
The most frequent keys(8)	9,10%	9,30%	12,54%	9,40%
The most frequent keys(5)	11,24%	9,92%	25,41%	23,85%
The smallest standard deviation keys(first 15 keys)	8,9%	7,66%	12,54%	12,24%
The smallest standard deviation keys (14)	**6,80%**	**7,20%**	15,09%	9,90%
The smallest standard deviation keys (13)	8,15%	8,67%	8,43%	7,47%
The smallest standard deviation keys (12)	8,44%	8,66%	9,44%	9,70%
The smallest standard deviation keys (10)	10,07%	8,77%	12,18%	8,48%
The smallest standard deviation keys (5)	8,49%	9,03%	17,68%	10,03%

Figure 6.19 shows graphically the values obtained from the tests, presented in the tables above.

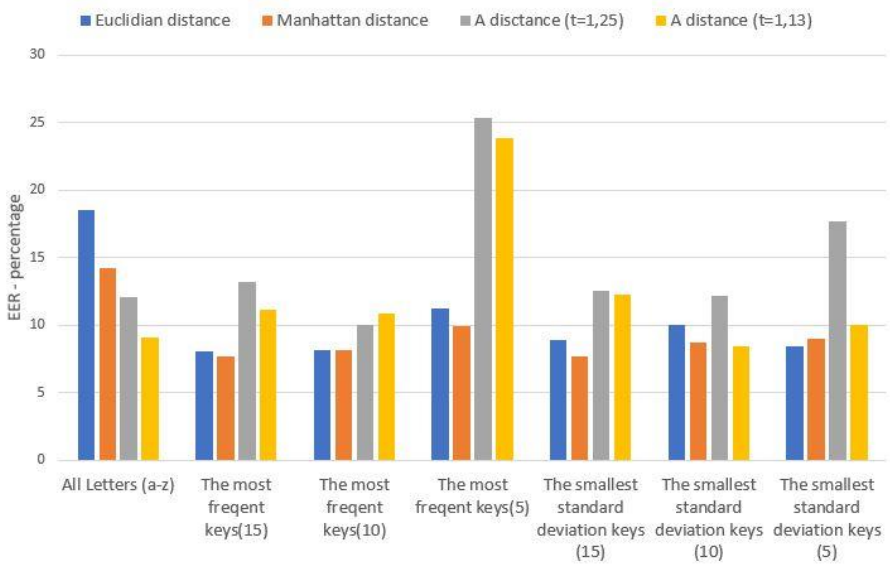


Figure 6.19 EER values in different conditions

6.2 Experiments with di-graphs

After the analysis performed on the individual characters of the data set and obtaining the presented results, the formation and analysis of di-graphs was continued. Di-graphs are pairs of two consecutive characters. The characteristics of a di-graph are: the total time of the di graph (DUtotal), the time of pressing the first key (DU1), the time of pressing the key 2 (DU2), the time between the two keys (UD), the time between the two down events (DD) and the time between the two up events (UU). The structure of a di-graph is presented in *Code 6.7*.

Code 6.7 A di-graph struct

```
typedef struct {
    int letter1, letter2;
    float DUtotal, DU1, DU2, DD, UU, UD;
    char key1[20],key2[20];
    char word[50];
}digraph;
digraph diGraphs[MAX];
```

In order to transform the information held up to this point into information relevant to di-graph, the constructDiGraphs (n, user) function presented in Code 6.8 was used.

Code 6.8 The function that builds the di-graph

```
int constructDiGraphs(int n, char user[])
```

```
{
  int i;
  for(i=0;i<n-1;i++) //construct diGraphs
          {
            diGraphs[i].letter1=oneGraphs[i].letter;
            diGraphs[i].letter2=oneGraphs[i+1].letter;
            strcpy(diGraphs[i].key1, oneGraphs[i].key);
            strcpy(diGraphs[i].key2, oneGraphs[i+1].key);
            diGraphs[i].DU1=oneGraphs[i].DU;
            diGraphs[i].DU2=oneGraphs[i+1].DU;
            diGraphs[i].UD=oneGraphs[i+1].UDprev;
            diGraphs[i].DD=diGraphs[i].UD+diGraphs[i].DU1;
            diGraphs[i].UU=diGraphs[i].UD+diGraphs[i].DU2;
            diGraphs[i].DUtotal=diGraphs[i].DU1+diGraphs[i].UD+diGraphs[i].DU2;
            constructWord(diGraphs[i].word,i,2);
fprintf(h,"%s\t%s\t%d\t%d\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%s\n",diGraphs
[i].key1,diGraphs[i].key2,diGraphs[i].letter1,diGraphs[i].letter2,diGraphs[i].DU1,diG
raphs[i].DU2,diGraphs[i].UD,diGraphs[i].DD,diGraphs[i].UU,diGraphs[i].DUtotal,diGr
aphs[i].word);
          }
  return 1;
}
```

Di-graph analysis takes into account the order in which characters are typed by users. From the database collected from users, a total number of 200,227 di-graphs could be created and analyzed. The total number of unique di-graphs is 1,530. This means that there are only 1,530 unique 2-character combinations. The most used di-graphs in the text are presented in Table 6.6 These are di-graphs that appear in texts taken from users more than 1000 times each.

Table 6.6 The most used di-graphs

No.	Key Code 1	Key Code 2	Key 1	Key 2	No. of aparitions	Average of DUtotal time
1	8	8	Backspace	Backspace	6938	3,39
2	65	32	A	Spacebar	6663	3,39
3	69	32	E	Spacebar	6630	3,27
4	73	32	I	Spacebar	4034	6,33
5	32	83	Spacebar	S	3866	8,69
6	73	78	I	N	3121	7,31
7	32	67	Spacebar	C	3104	11,09
8	82	69	R	E	3027	6,39
9	32	80	Spacebar	P	2911	12,97
10	32	68	Spacebar	D	2846	11,58
11	32	65	Spacebar	A	2654	12,4

12	65	82	A	R	2606	9,38
13	84	69	T	E	2453	8,19
14	68	69	D	E	2343	8,45
15	67	65	C	A	2289	8,93
16	32	73	Spacebar	I	2170	16,85
17	65	84	A	T	2160	11,51
18	85	32	U	Spacebar	2034	11,61
19	188	32	,	Spacebar	2012	12,51
20	32	77	Spacebar	M	1846	18,05
21	84	65	T	A	1812	12,01
22	78	32	N	Spacebar	1769	12,09
23	83	84	S	T	1752	13,84
24	84	73	T	I	1631	13,38
25	82	65	R	A	1540	15,28
26	83	73	S	I	1526	12,71
27	69	65	E	A	1514	17,45
28	78	84	N	T	1508	16
29	83	65	S	A	1498	13,93
30	82	73	R	I	1468	14,03
31	84	32	T	Spacebar	1457	16,65
32	69	83	E	S	1452	18,25
33	67	69	C	E	1435	14,54
34	77	65	M	A	1380	14,44
35	69	82	E	R	1373	16,97
36	32	8	Spacebar	Backspace	1372	46,92
37	85	78	U	N	1325	17,41
38	190	32	,	Spacebar	1311	28,81
39	32	70	Spacebar	F	1287	27,34
40	76	65	L	A	1272	15,25
41	85	76	U	L	1258	21,55
42	32	16	Spacebar	Shift	1210	52,88
43	80	69	P	E	1189	16,93
44	84	82	T	R	1170	17,07
45	67	85	C	U	1167	16,37
46	76	32	L	Spacebar	1154	21,6
47	32	69	Spacebar	E	1153	30,22

48	76	69	L	E	1117	16,62
49	32	76	Spacebar	L	1106	32,08
50	65	67	A	C	1103	22,21
51	80	82	P	R	1071	21,22
52	69	78	E	N	1063	20,61
53	65	78	A	N	1046	21,16
54	65	76	A	L	1043	21,54
55	32	79	Spacebar	O	1035	37,73
56	79	82	O	R	1021	22,86

From the table it can extract the most common combination of 2 keys is di-graph Backspace-Backspace with a number of 6,938 occurrences. This key combination is pressed even the fastest by a user, on average in 3.39 milliseconds. The next 4 frequency key combinations are letter combinations and the SPACE key. Di-graphs: A-Space, E-Space, I-Space and Space-S. The first two-letter combination that appears most frequently is the I-N di-graph with a number of 3121 occurrences. A total of 56 di-graphs appear in the text more than 1000 times. It is observed in Figure 6.20 that the total typing time of a di-graph increases as the frequency of using the 2 keys decreases. The figure shows the total typing time of each di-graph. On the left are di-graphs the most common, while on the right the least common. The increasing trend of typing time is observed.

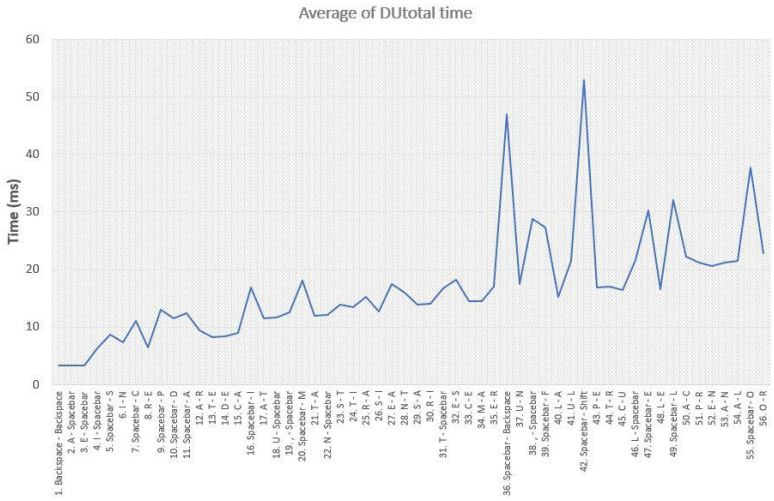


Figure 6.20 Di-graph time increases with decreasing frequency of use

6.2.1 Creating the user pattern

For each unique combination of a di-graph, the mean and standard deviation were calculated for all 5 calculated times. The structure that retains the information about a unique di-graphs at the level of each user is presented in *Code 6.9*.

Code 6.9 The pattern struct

```
typedef struct{
  int letter1,letter2;
  double meanDUtotal, meanDU1, meanDU2, meanDD, meanUU, meanUD;
  int nr;
  double stdDevDUtotal, stdDevDU1, stdDevDU2, stdDevDD, stdDevUU, stdDevUD;
  float minDUtotal, maxDUtotal;
 }userDiPattern;

typedef struct{
   char user[20];
   userDiPattern pattern[10000];
   int nPattern;
   int sampleSize;
   float distance[5000]; //distance from other users
 }diPattern;
 diPattern diPatterns[5000];
 int nDiPatterns=0;
```

Using the constructDiPattern (user, n) function, the required data about each unique graph for each user was calculated. For each combination of 2 keys, the averages of the 5 time intervals specific to a di-graph were calculated. In addition to the average, the minimum and maximum values of the times were retained and the standard deviation for each of the 6 time intervals specific to a di-graph was calculated. With the help of these data, it was possible to calculate the similarity between users for each unique graph.

In the vector distance[5000] within diPatterns[] the distance calculated in various ways between the resulting vectors for each two users will be retained. The shorter the calculated distance, the more likely it is to be accessed by the right user. The function that builds the patterns is in *Code 6.10*.

Code 6.10 The function that build the patterns

```
int constructDiPattern(char user[], int n)
{
  int i,j;
  strcpy(diPatterns[nDiPatterns].user,user);
  diPatterns[nDiPatterns].sampleSize=n-1;
  for(i=0;i<n-1;i++)
  {
    for(j=0;j<diPatterns[nDiPatterns].nPattern;j++)
    {
      if(diGraphs[i].letter1 == diPatterns[nDiPatterns].pattern[j].letter1 &&
diGraphs[i].letter2 == diPatterns[nDiPatterns].pattern[j].letter2)
        break;
    }
        diPatterns[nDiPatterns].pattern[j].letter1=diGraphs[i].letter1;
        diPatterns[nDiPatterns].pattern[j].letter2=diGraphs[i].letter2;
        diPatterns[nDiPatterns].pattern[j].nr++;
        diPatterns[nDiPatterns].pattern[j].meanDUtotal+=diGraphs[i].DUtotal;
        diPatterns[nDiPatterns].pattern[j].meanDU1+=diGraphs[i].DU1;
        diPatterns[nDiPatterns].pattern[j].meanDU2+=diGraphs[i].DU2;
```

```
        diPatterns[nDiPatterns].pattern[j].meanDD+=diGraphs[i].DD;
        diPatterns[nDiPatterns].pattern[j].meanUU+=diGraphs[i].UU;
        diPatterns[nDiPatterns].pattern[j].meanUD+=diGraphs[i].UD;
        if(diPatterns[nDiPatterns].pattern[j].minDUtotal>diGraphs[i].DUtotal ||
j==diPatterns[nDiPatterns].nPattern)
          diPatterns[nDiPatterns].pattern[j].minDUtotal=diGraphs[i].DUtotal;
        if(diPatterns[nDiPatterns].pattern[j].maxDUtotal<diGraphs[i].DUtotal ||
j==diPatterns[nDiPatterns].nPattern)
          diPatterns[nDiPatterns].pattern[j].maxDUtotal=diGraphs[i].DUtotal;
        if(j==diPatterns[nDiPatterns].nPattern)
          diPatterns[nDiPatterns].nPattern++;
    }
    for(j=0;j<diPatterns[nDiPatterns].nPattern;j++)
    {
diPatterns[nDiPatterns].pattern[j].meanDUtotal/=diPatterns[nDiPatterns].pattern[j].
nr;
diPatterns[nDiPatterns].pattern[j].meanDU1/=diPatterns[nDiPatterns].pattern[j].nr;
diPatterns[nDiPatterns].pattern[j].meanDU2/=diPatterns[nDiPatterns].pattern[j].nr;
diPatterns[nDiPatterns].pattern[j].meanDD/=diPatterns[nDiPatterns].pattern[j].nr;
diPatterns[nDiPatterns].pattern[j].meanUU/=diPatterns[nDiPatterns].pattern[j].nr;
diPatterns[nDiPatterns].pattern[j].meanUD/=diPatterns[nDiPatterns].pattern[j].nr;
    }
    for(j=0;j<diPatterns[nDiPatterns].nPattern;j++)
    {
      for(i=0;i<n;i++)
      {
        if(diGraphs[i].letter1 == diPatterns[nDiPatterns].pattern[j].letter1 &&
diGraphs[i].letter2 == diPatterns[nDiPatterns].pattern[j].letter2)
        {
diPatterns[nDiPatterns].pattern[j].stdDevDUtotal+=pow(diPatterns[nDiPatterns].pat
tern[j].stdDevDUtotal+diGraphs[i].DUtotal,2);
diPatterns[nDiPatterns].pattern[j].stdDevDU1+=pow(diPatterns[nDiPatterns].patter
n[j].stdDevDU1+diGraphs[i].DU1,2);
diPatterns[nDiPatterns].pattern[j].stdDevDU2+=pow(diPatterns[nDiPatterns].patter
n[j].stdDevDU2+diGraphs[i].DU2,2);
diPatterns[nDiPatterns].pattern[j].stdDevDD+=pow(diPatterns[nDiPatterns].pattern
[j].stdDevDD+diGraphs[i].DD,2);
diPatterns[nDiPatterns].pattern[j].stdDevUU+=pow(diPatterns[nDiPatterns].pattern
[j].stdDevUU+diGraphs[i].UU,2);
diPatterns[nDiPatterns].pattern[j].stdDevUD+=pow(diPatterns[nDiPatterns].pattern
[j].stdDevUD+diGraphs[i].UD,2);
        }
      }
    }
    for(j=0;j<diPatterns[nDiPatterns].nPattern;j++)
    {
diPatterns[nDiPatterns].pattern[j].stdDevDUtotal=sqrt(diPatterns[nDiPatterns].patte
rn[j].stdDevDUtotal/diPatterns[nDiPatterns].pattern[j].nr);
diPatterns[nDiPatterns].pattern[j].stdDevDU1=sqrt(diPatterns[nDiPatterns].pattern[
j].stdDevDU1/diPatterns[nDiPatterns].pattern[j].nr);
```

```
diPatterns[nDiPatterns].pattern[j].stdDevDU2=sqrt(diPatterns[nDiPatterns].pattern[
j].stdDevDU2/diPatterns[nDiPatterns].pattern[j].nr);
diPatterns[nDiPatterns].pattern[j].stdDevDD=sqrt(diPatterns[nDiPatterns].pattern[j]
.stdDevDD/diPatterns[nDiPatterns].pattern[j].nr);
diPatterns[nDiPatterns].pattern[j].stdDevUU=sqrt(diPatterns[nDiPatterns].pattern[j]
.stdDevUU/diPatterns[nDiPatterns].pattern[j].nr);
diPatterns[nDiPatterns].pattern[j].stdDevUD=sqrt(diPatterns[nDiPatterns].pattern[j]
.stdDevUD/diPatterns[nDiPatterns].pattern[j].nr);
  }
  nDiPatterns++;
  return 1;
}
```

6.2.2 Authentication accuracy

Authentication accuracy is assessed with Equal Error Rate (EER), the percentage at which False Acceptance Rate (FAR) and False Rejection Rate (FRR) have equal value. Another indicator of algorithm performance, in addition to EER, is, according to [ZHO12] [KIL09] Zero Miss False Acceptance Rate (ZMFAR). ZMFAR is represented by the minimum percentage of FRR (False Rejection Rate) when FAR (False Alarm Rate) has the value equal to 0. In Figure 20 are graphically represented the two performance indicators of a user authentication algorithm in the system.

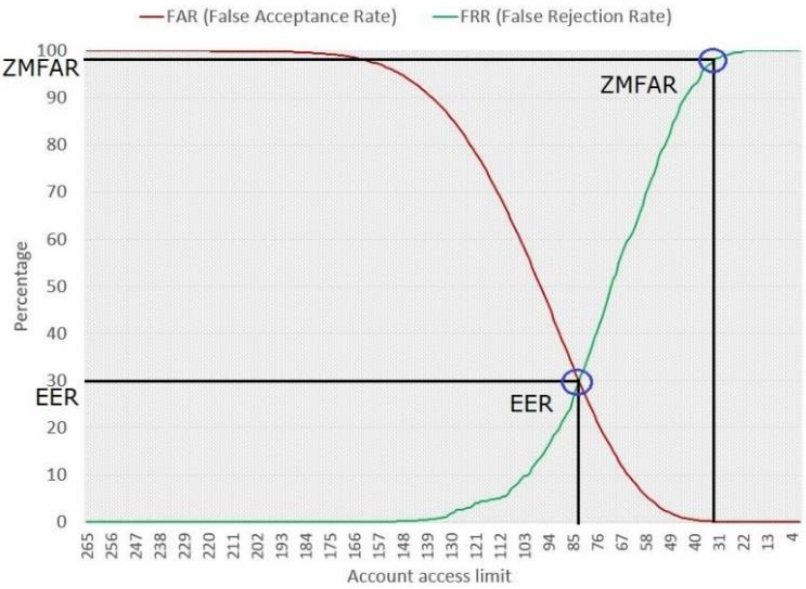


Figure 6.21 Graphical reprezentation of Equal Error Rate (EER) and ZMFAR (Zero Miss False Acceptance Rate)

For clearer information, the Zero Miss False Acceptance Rate (ZMFAR) for the algorithm was further calculated. The best performances obtained according to the two error rates in [ZHO12] and [KIL09] are presented in the Table 6.7.

Table 6.7 Performance for two algorithms from other studies in terms of EER and ZMFAR

Paper	Algorithm	EER (%)	ZMFAR (%)
[ZHO12] Zhong, Deng and Jain	Nearest Neighbor	8.4	40.5
[KIL09] Killourhy and Maxion	Manhattan (scaled)	9.4	46.8

Access tests were further performed for a sample size of 1000 keys (2000 key events). The first results obtained for di-graphs are presented in Table 6.8. The table contains the EER values. It is surprising that from the first tests very good results were obtained for the EER value in the case of the algorithm that uses A distance.

Table 6.8 EER values in different conditions

EER			
keys / distance	All Keys	Only letters	Only letters and digits
Euclidian distance	23,13	23,93	23,86
Manhattan distance	19,66	18,5	18,5
A distance (t=1,25)	6,55	9,48	9,46
A distance (t=1,13)	6,62	8,21	8,21

In Table 6.9 are the values obtained for Zero Miss False Acceptance Rate (ZMFAR) in the case of tests performed whose EER is presented in Table 9. As in the case of EER, the lowest values of ZMFAR were obtained for the algorithm using A distance. The lowest value obtained for ZMFAR is 49.71% in the case of the A distance algorithm, when only di-graphs containing only letters (a-z) were monitored, the other characters being eliminated from the algorithm.

Table 6.9 ZMFAR values in different conditions

ZMFAR			
keys / distance	All Keys	Only letters	Only letters and digits
Euclid distance	53,76	53,76	53,76
Manhattan distance	52,6	53,18	53,18
A distance (t=1,25)	52,6	53,18	53,18
A distance (t=1,13)	52,02	49,71	49,71

Considering the first results obtained for values of di-graphs, it was concluded that if only the letters of the English alphabet (a-z) are analyzed or if the letters of the English alphabet (a-z) are analyzed and the numbers (0-9) the results obtained are similar. For this reason, two types of tests were performed. The first type of tests in which only di-graphs containing only letters were analyzed, and the second type of di-graphs tests consisting of all the keys taken, whether they are letters, numbers, punctuation marks, spaces or other special characters.

6.2.3 Experiments with Euclidian distance at di-graphs

In this section, to calculate the similarities between two vectors (two distinct users or two vectors obtained from the text of the same user) it was applied the calculation of the Euclidean distance at di-graphs. It was considered that the user has successfully accessed the account if the Euclidean distance (calculated between the vector resulting from the user's key events and the vector resulting from the key events of the account to be accessed) was less than a certain threshold. In case it was higher than the certain threshold, it was considered that he failed to access the account. *Code 6.11* shows the function that calculate Euclidian distance at di-graphs.

Code 6.11 The function that calculates Euclidian distance

```
float EuclidianDistanceDiGraph(int first)
{
  int user1,user2,i,j,nr=0;
  float max=0;

   for(user1=0;user1<nDiPatterns;user1++)
  {
    for(user2=0;user2<nDiPatterns;user2++)
    {
      diPatterns[user1].distance[user2]=0;
    }
  }
  for(user1=0;user1<nDiPatterns;user1++)
  {
    for(user2=0;user2<nDiPatterns;user2++)
    {
      nr=0;
      for(i=0;i<diPatterns[user1].nPattern;i++)
        {
          for(j=0;j<diPatterns[user2].nPattern;j++)
          {
            if(diPatterns[user1].pattern[i].letter1 ==
diPatterns[user2].pattern[j].letter1 && diPatterns[user1].pattern[i].letter2 ==
diPatterns[user2].pattern[j].letter2)
            {
               break;
            }
          }
          if(j!=diPatterns[user2].nPattern)
          {
diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanDUtotal-
diPatterns[user2].pattern[j].meanDUtotal,2);
diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanDU1-
diPatterns[user2].pattern[j].meanDU1,2);
diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanDU2-
diPatterns[user2].pattern[j].meanDU2,2);
diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanDD-
diPatterns[user2].pattern[j].meanDD,2);
diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanUU-
```

```
diPatterns[user2].pattern[j].meanUU,2);
diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanUD-
diPatterns[user2].pattern[j].meanUD,2);
          nr+=6;
        }
      }
      diPatterns[user1].distance[user2]=sqrt(diPatterns[user1].distance[user2]);
      if(max<diPatterns[user1].distance[user2])
      {
        max=diPatterns[user1].distance[user2];
      }
    }
  }
 }
 return max;
}
```

In order to be able to compare the performances of the different types of tests, tests were performed to work only with the most used di-graphs. For the algorithm that uses Euclidian distance, accesses of the accounts were simulated and took into account one by one, first the most used di-graph, ie the one consisting of the letters IN, then the first two (IN and RE), then the first 3, 4 etc. The performances obtained for the first 50 tests are found in Table 6.10 and represented graphically in Figure 6.22. The best result obtained in this series of tests is in the case of EER for the analysis of the first 10 di-graphs as well as the frequency of use. The EER value is 16.81%.

Table 6.10 EER values with first # di-graphs

First # di-graphs	EER (%)
1	38,01
2	27,15
3	28,72
4	21,97
5	19,68
6	18,55
7	17,72
8	19,02
9	17,89
10	**16,81**
11	23,04
12	19,04
13	19,85
14	22,54
15	19,82
16	19,89
17	23,04
18	22,66
19	20,4
20	20,74
21	21,45
22	21,77
23	22,64
24	20,9
25	21,6
26	22,35
27	21,75
28	22,95
29	22,02

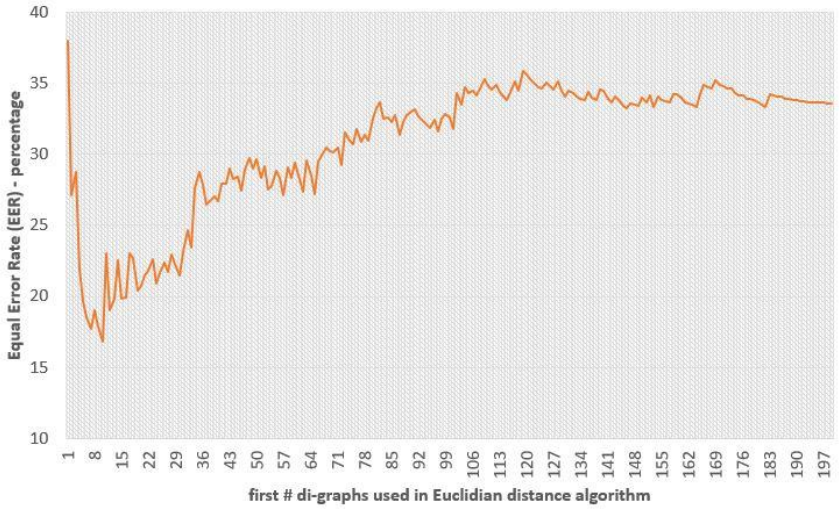


Figure 6.22 EER values with first # di-graphs

6.2.4 Experiments with Manhattan distance at di-graphs

In this section, to calculate the similarities between two vectors (two distinct users or two vectors obtained from the text of the same user) it was applied the calculation of the Manhattan distance at di-graphs. It was considered that the user has successfully accessed the account if the Manhattan distance (calculated between the vector resulting from the user's key events and the vector resulting from the key events of the account to be accessed) was less than a certain threshold. In case it was higher than the certain threshold, it was considered that he failed to access the account. *Code 6.12* shows the function that calculate Manhattan distance at di-graphs.

Code 6.12 The function that calulates Manhattan distance for di-graph

```
float ManhattanDistanceDiGraph(int first)
{
  int user1,user2,i,j,nr=0;
  float max=0;

   for(user1=0;user1<nDiPatterns;user1++)
  {
    for(user2=0;user2<nDiPatterns;user2++)
    {
      diPatterns[user1].distance[user2]=0;
    }
  }
  for(user1=0;user1<nDiPatterns;user1++)
  {
    for(user2=0;user2<nDiPatterns;user2++)
    {
      nr=0;
```

```
    for(i=0;i<diPatterns[user1].nPattern;i++)
     {
      for(j=0;j<diPatterns[user2].nPattern;j++)
      {
       if(diPatterns[user1].pattern[i].letter1 ==
diPatterns[user2].pattern[j].letter1 && diPatterns[user1].pattern[i].letter2 ==
diPatterns[user2].pattern[j].letter2)
        {

if(firstDiPatterns(first,diPatterns[user1].pattern[i].letter1,diPatterns[user1].pattern[i
].letter2))
          {
           break;
          }
        }
      }
      if(j!=diPatterns[user2].nPattern)
      {
diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanDUtotal-
diPatterns[user2].pattern[j].meanDUtotal;
diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanDU1-
diPatterns[user2].pattern[j].meanDU1);
diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanDU2-
diPatterns[user2].pattern[j].meanDU2);
diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanDD-
diPatterns[user2].pattern[j].meanDD);
diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanUU-
diPatterns[user2].pattern[j].meanUU);
diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanUD-
diPatterns[user2].pattern[j].meanUD);
        nr+=1;
      }
     }
     if(max<diPatterns[user1].distance[user2])
     {
      max=diPatterns[user1].distance[user2];
     }
   }
  }
  return max;
}
```

In order to be able to compare the performances of the different types of tests, tests were performed to work only with the most used di-graphs. For the algorithm that uses Manhattan distance, accesses of the accounts were simulated and took into account one by one, first the most used di-graph, ie the one consisting of the letters IN, then the first two (IN and RE), then the first 3, 4 etc. The performances obtained for the first 50 tests are found in Table 6.11 and represented graphically in Figure 6.23. The best result obtained in this series of tests is in the case of EER for the analysis of the first 12 di-graphs as well as the frequency of use. The EER value is 13.89% [IAP21a].

Table 6.11 EER values for Manhattan distance at different numbers of di-graphs

First # di-graphs	EER(%)
1	38,01
2	29,7
3	22,7
4	19,67
5	16,34
6	15,99
7	15,54
8	15,8
9	16,02
10	16,11
11	15,42
12	**13,89**
13	14,31
14	16,3
15	15,78
16	15,51
17	15,55
18	15,26
19	15,25
20	16,29
21	16,91
22	17,75
23	17,56
24	17,57
25	16,73
26	16,48
27	16,23
28	16,6
29	17,39

From the analysis of the graph in Figure 6.23 it is observed that better values for EER are obtained when analyzing a small number of di-graphs but with high frequency in the text in the database. This also helps in the analysis, being faster to analyze a smaller number of elements [IAP21a].

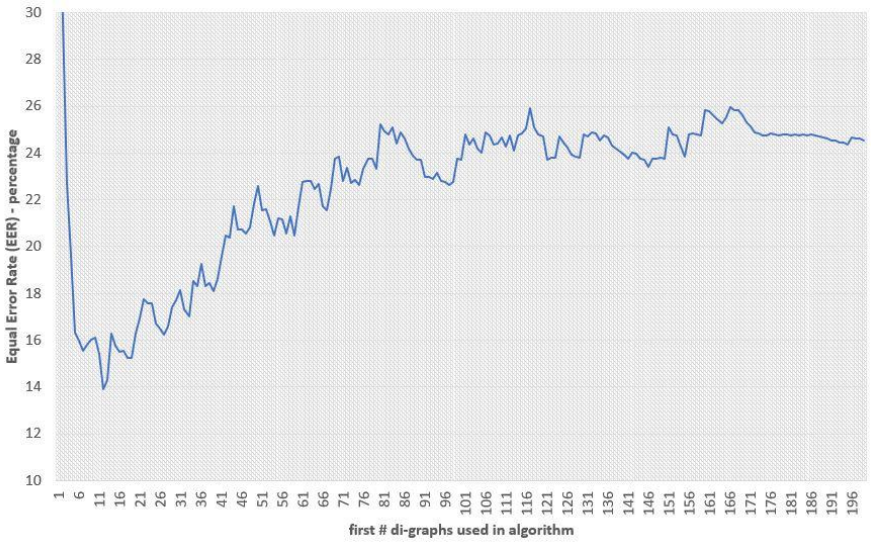


Figure 6.23 EER values for Manhattan distance at different numbers of di-graphs [IAP21a]

6.2.5 Experiments with A distance at di-graphs

In this section, to calculate the similarities between two vectors (two distinct users or two vectors obtained from the text of the same user) it was applied the calculation of the A distance at di-graphs. It was considered that the user has successfully accessed the account if the A distance (calculated between the vector resulting from the user's key events and the vector resulting from the key events of the account to be accessed) was less than a certain threshold. In case it was higher than the certain threshold, it was considered that he failed to access the account. *Code 6.13* shows the function that calculate A distance at di-graphs.

Code 6.13 The function that calulates A distance for di-graph

```
float ADistanceDiGraph(float t, float first)
{
  int user1,user2,i,j,nr=0;
  float max=0;
   for(user1=0;user1<nDiPatterns;user1++)
  {
    for(user2=0;user2<nDiPatterns;user2++)
    {
      diPatterns[user1].distance[user2]=0;
    }
  }
  for(user1=0;user1<nDiPatterns;user1++)
  {
    for(user2=0;user2<nDiPatterns;user2++)
    {
      nr=0;
      for(i=0;i<diPatterns[user1].nPattern;i++)
        {
          for(j=0;j<diPatterns[user2].nPattern;j++)
          {
            if(diPatterns[user1].pattern[i].letter1 ==
diPatterns[user2].pattern[j].letter1 && diPatterns[user1].pattern[i].letter2 ==
diPatterns[user2].pattern[j].letter2)
            {
               nr++;
               break;
            }
          }
          if(j!=diPatterns[user2].nPattern)
          {
            if(diPatterns[user1].pattern[i].meanDUtotal >
diPatterns[user2].pattern[j].meanDUtotal)
            {
              if(diPatterns[user1].pattern[i].meanDUtotal /
diPatterns[user2].pattern[j].meanDUtotal < t)
                 diPatterns[user1].distance[user2]++;
            }
            else
```

```
          {
            if(diPatterns[user2].pattern[j].meanDUtotal /
diPatterns[user1].pattern[i].meanDUtotal < t)
              diPatterns[user1].distance[user2]++;
          }
        }
      }
    diPatterns[user1].distance[user2]=1-diPatterns[user1].distance[user2]/nr;

    if(max<diPatterns[user1].distance[user2])
    {
      max=diPatterns[user1].distance[user2];
    }
   }
  }
  return max;
}
```

The same analysis as for Euclidean distance and Manhattan distance was applied to A distance. Figure 6.24 shows graphically the values obtained for EER when analyzing a different number of di-graphs. In this case, the analysis shows other conclusions than in Euclidean and Manhattan. It turns out that the best performance is not obtained by analyzing a few di-graphs, but by analyzing a large number of di-graphs. The same result is if t takes different values. In this context, in the case of A distance it is more efficient if all the letters are analyzed.

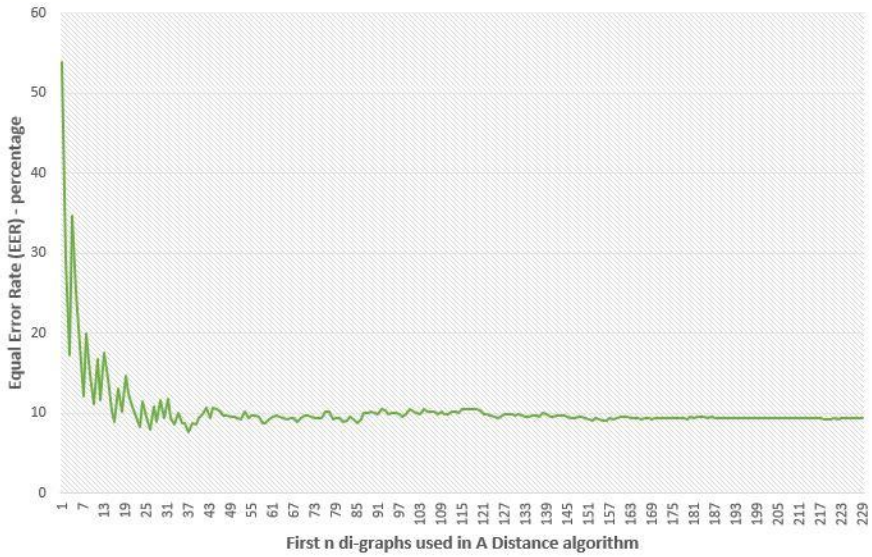


Figure 6.24 EER values for A distance (t=1,25) at different numbers of di-graphs analized

For the algorithm using A distance it was analyzed which values are the most advantageous to set for the constant t. The algorithm was applied for values from 1 to 2 with a step of 0.01 and it was concluded that the best values of EER are obtained

in the interval t = 1.07 - 1.28. The best performance was obtained at t = 1.22. The EER had a value of 5.52%. In these tests all the typed characters were taken into account, not only the letters (a-z). The values obtained for EER are presented in Table 6.12 for values of t between 1 and 1.3.

Table 6.12 EER values with A distance, for different values for t

t value	EER				
1,01	18,56	**1,1**	**5,85**	**1,2**	6,22
1,02	12,6	**1,11**	**5,94**	**1,21**	6,11
1,03	10,11	**1,12**	**5,86**	1,22	**5,52**
1,04	8,51	**1,13**	6,62	**1,23**	**5,81**
1,05	7,47	**1,14**	**5,85**	**1,24**	**5,91**
1,06	7,67	**1,15**	**5,73**	**1,25**	6,62
1,07	**5,83**	**1,16**	**5,78**	**1,26**	6,38
1,08	6,42	**1,17**	**5,78**	**1,27**	6,38
1,09	6,26	**1,18**	6,02	**1,28**	6,95
		1,19	**5,61**	**1,29**	7,67

The graph in Figure 6.25 shows the EER values over the entire examined range (1.00 - 1.99). The graph shows that at lower values of t the best performances are obtained.

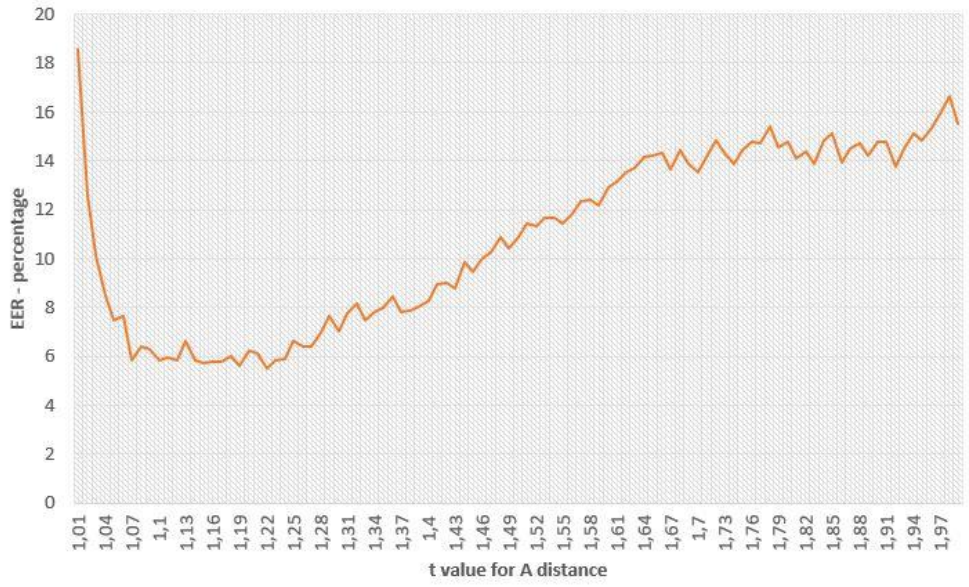


Figure 6.25 EER values with A distance, for different values for t

6.2.6 The results of experiments with di-graphs

The analyzes made up to this point on di-graphs take into account a single time interval of the di-graph, the total time of the di-graphs, DUtotal. The interval between key 1 down event and key 2 up event. The interim conclusions are:

- The Manhattan distance algorithm performs better when analyzing a limited number of di-graphs, those with the highest frequency
- The Euclidean distance algorithm obtains better performances when analyzing a limited number of di-graphs, the ones with the highest frequency
- Manhattan distance algorithm performs better when analyzing di-graphs that contain only letters (a-z)
- The Euclidean distance algorithm performs better when analyzing di-graphs that contain only letters (a-z)
- Under similar conditions, the Manhattan distance algorithm performs better than the Euclidean distance algorithm. Figure 6.26 shows graphically the results of the two algorithms for different numbers of di-graphs used in the calculation of the distance

Given the above conclusions, further, between the two algorithms, only the Manhattan distance algorithm for di-graphs was analyzed.

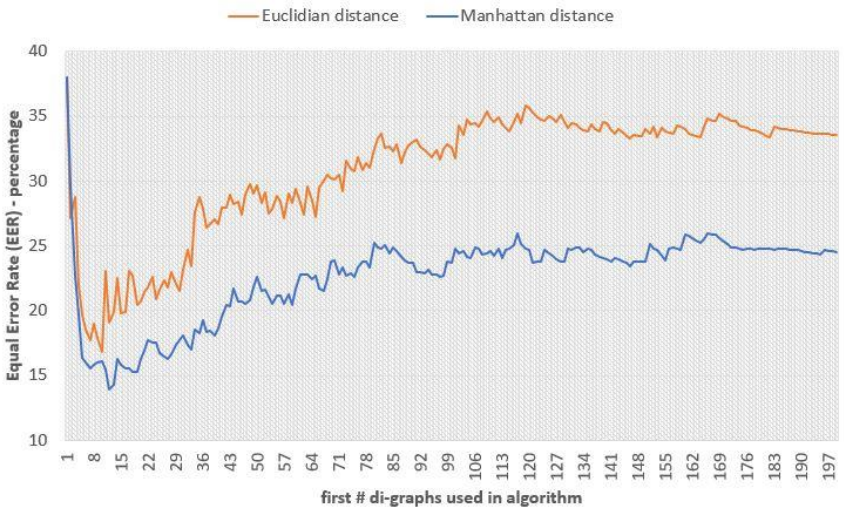


Figure 6.26 Euclidean distance performance compared to Manhattan distance performance

In connection with A distance, the intermediate conclusions are:

- The A distance algorithm performs better when analyzing all keys, not just letters (a-z)
- The A distance algorithm obtains better performances when t is in the range 1.07 - 1.28
- The A distance algorithm performs better when analyzing all key combinations, not just the most common ones

Analyzing the intermediate conclusions, it can be said that the A distance algorithm needs more resources to function, both memory space and time. From these perspectives, the most efficient to use, among the analyzed algorithms, according to the obtained results, is the Manhattan distance algorithm.

6.2.7 Choosing the time components of a di-graph

Next, experiments were performed with different combinations of the times generated by a di-graph, not only with the total time of the di-graph. Table 6.13 shows the performances obtained. The 6 times that a di-graph generates are: the pressing time of the first key (DU1), the pressing time of the second key (DU2), the total time of the di-graph (DUtotal), the time between the two keys (UD), the interval between pressing the first and pressing the second key (DD) and the interval between raising the first key and raising the second key (UU).

Table 6.13 The most efficient combinations of times for calculating the distance [IAP21a]

	Components	**EER (%)**
1	**Dutotal, DU1, DU2**	**5,23**
2	**DU1, DU2, UD**	**5,42**
3	**DU1, DU2**	**5,69**
4	Dutotal, DU1, DU2, UD	6,47
5	All without UU	6,52
6	DU1, DU2, UU, DD	6,61
7	DU1, DU2, UU, DD, UD	7,11
8	All 6 intervals	7,53
9	All without DD	7,58
10	All without UD	7,68
11	DU1	8,46
12	All without DU2	8,7
13	All without DU1	8,75
14	Dutotal + UD	10,06
15	UU, DD, UD	11,15
16	DU2	11,23
17	UD	11,46
18	UU	12,32
19	Dutotal	13,89
20	DD	15,18

The best performances were obtained when using only 3 of the 6 times generated by the di-graph: the pressing time of the first key (DU1), the pressing time of the second key (DU2) and the total time of di-graph (DUtotal). For these components EER = 5.23% was obtained when calculating the distance with Manhattan distance and using the times only from the most frequent 12 di-graphs.

In Figure 6.27 are represented graphically FAR and FRR for the case where the best performance was obtained, calculating the distance using the 3 time intervals of the 6: DU1, DU2 and DUtotal. The EER value obtained in this case is 5.32%. The simulation was performed using only the first 12 di-graphs, the most common [IAP21a].

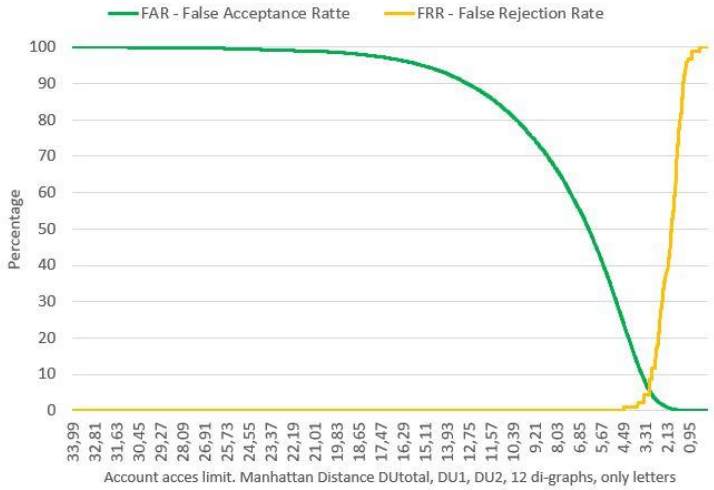


Figure 6.27 FAR and FRR for Manhattan Distance DU1, DU2,UD , first 12 di-graphs, only letters [IAP21a]

The graph in Figure 6.28 shows the ROC curves for the best performing case in this experiment. EER value = 5.32% [IAP21a].

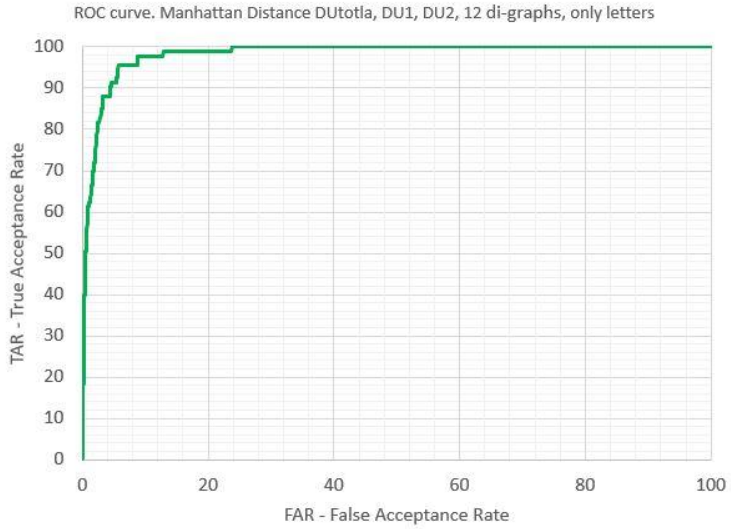


Figure 6.28 ROC curve. Manhattan Distance DU1, DU2,UD , first 12 di-graphs, only letters [IAP21a]

6.3 The Distances between users

The distance calculated between two user patterns is shown in Figure 6.29 for the first 18 users out of a total of 160. The sample size used is 1000 keys (2000 key events). Thus, from the characters taken from the 80 users, 160 patterns resulted in the following way: from those who typed less than 2000 keys, only the first 1000 were valued, and the rest were ignored. Those who typed under 3000 keys generated 2 samples, each of 1000 keys, and those who typed over 3000 keys generated 3 samples each. For each sample, distances were calculated between the pattern generated by it and the other 159 samples. In the figures it can be seen that the smallest distances were calculated between samples from the same user. The sample name consists of: UserNumber.SampleNumber.

user	1,01	2,01	2,02	3,01	3,02	4,01	4,02	5,01	5,02	6,01	7,01	7,02	8,01	8,02	9,01	9,02	10,01	10,02
1,01	0	27,72	24,53	17,2	17,11	16,41	17,23	14,54	15,2	11,66	12,48	13,95	15,76	14,25	21,54	8,95	12,01	10,84
2,01	27,72	0	5,43	12,08	12,43	20,96	20,93	18,15	15,23	28,46	26,99	27,95	15,51	17,8	28,65	24,49	33,56	37,87
2,02	24,53	5,43	0	9,21	9,48	18,48	21,91	16,1	12,79	25,93	24,61	25,43	12,06	15,79	28,75	21,98	29,68	34,57
3,01	17,2	12,08	9,21	0	2,96	21,28	23,38	13,07	12,17	18,52	19,18	22,62	8,26	10,42	24,18	16,56	26,29	27,61
3,02	17,11	12,43	9,48	2,96	0	20,51	22,47	13,35	12,74	17,51	18,9	23,06	8,35	10,14	23,8	16,07	26,75	27,08
4,01	16,41	20,96	18,48	21,28	20,51	0	7,91	16,72	16,02	19,68	17,65	17,33	20,44	20,25	19,83	13,74	19	23,71
4,02	17,23	20,93	21,91	23,38	22,47	7,91	0	18,61	18,02	19,65	18,57	17,75	21,34	21,15	22,15	13,09	19,4	21,82
5,01	14,54	18,15	16,1	13,07	13,35	16,72	18,61	0	4,64	18,36	12,79	16,69	15,95	14,64	26,54	16,46	23,52	24,79
5,02	15,2	15,23	12,79	12,17	12,74	16,02	18,02	4,64	0	19,61	12,95	15,57	14,85	14,98	25,57	16,3	23,88	25,62
6,01	11,66	28,46	25,93	18,52	17,51	19,68	19,65	18,36	19,61	0	18,6	21,93	16,36	14,07	23,44	13,31	16,36	13,75
7,01	12,48	26,99	24,61	19,18	18,9	17,65	18,57	12,79	12,95	18,6	0	6,34	18,53	17,74	25,57	14,14	19,91	17,8
7,02	13,95	27,95	25,43	22,62	23,06	17,33	17,75	16,69	15,57	21,93	6,34	0	20,79	20,75	25,02	14,36	19,72	18,4
8,01	15,76	15,51	12,06	8,26	8,35	20,44	21,34	15,95	14,85	16,36	18,53	20,79	0	5,47	22,88	15,6	24,85	24,37
8,02	14,25	17,8	15,79	10,42	10,14	20,25	21,15	14,64	14,98	14,07	17,74	20,75	5,47	0	21,37	15,66	23,54	22,89
9,01	21,54	28,65	28,75	24,18	23,8	19,83	22,15	26,54	25,57	23,44	25,57	25,02	22,88	21,37	0	14,54	25,79	26,7
9,02	8,95	24,49	21,98	16,56	16,07	13,74	13,09	16,46	16,3	13,31	14,14	14,36	15,6	15,66	14,54	0	13,37	13,23
10,01	12,01	33,56	29,68	26,29	26,75	19	19,4	23,52	23,88	16,36	19,91	19,72	24,85	23,54	25,79	13,37	0	8,18
10,02	10,84	37,87	34,57	27,61	27,08	23,71	21,82	24,79	25,62	13,75	17,8	18,4	24,37	22,89	26,7	13,23	8,18	0

Figure 6.29 Distances between first 18 users from all 160. Green is a small distance and red is a big distance

Figure 6.30 shows the entire database, with all 160 samples, the green color representing the short distance and the red color representing the large distance. The green color shows the small distance between samples from the same users or the red color if the distances are the largest. The figure above shows the upper left part with more details.

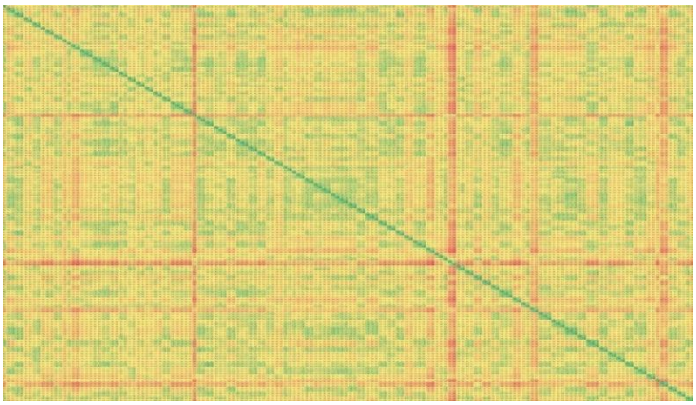


Figure 6.30 Distances between 160 users. Green is a small distance and red is a big distance

6.4 Proposing new metrics for calculating distances between users

This subchapter addresses O3, formulated in the first chapter of this thesis. The performance improvement of two metrics is analyzed. The first metric starts from the distance of Manhattan and proposes a modification of it in case of calculating the distance with the help of the times from one key only. The second metric proposed in this subchapter also starts from the distance of Manhattan, but the times used in the distance calculation are times used in the di-graphs.

6.4.1 New metric for calculating distances based on individual key time

Following the experiments performed in this research and presented in detail in this chapter, research undertaken in other scientific research has concluded that changes in the metrics used to calculate distances can be generated to improve the performance of algorithms.

In the case of calculating distances only from the information of each key, not a di-graph, if a Manhattan distance calculated between two components within two time vectors is very small, it can be ignored, or even lead to a decrease the entire distance, because the probability of coming from the same user is high. In this judgment it was proposed by this paper to adjust the difference between x_i and y_i with a certain percentage of the standard deviation calculated for the first user. It was chosen not to subtract a fixed value, but a certain percentage of the standard deviation because this is a property of all times relative to a certain key, just like their average. In formula 6.1 is the formula that was applied following this reasoning.

$$d(x,y) = \sum_{i=1}^{14} (|x_i - y_i| - C \times \sigma_{x_i}) \tag{6.1}$$

Where d(x,y) is the distance between the two vectors x and y, C is the coefficient and σ_{x_i} is the standard deviation from x_i.

Applying the proposed new formula, different values were generated for the C coefficient and the performance obtained by the algorithm was monitored. In Table 6.14 are the values obtained for the EER for each coefficient in the first column of the table. The green color represents a small value and the red color a higher value of the error. It can be seen in the table that the best performances were obtained in the range C = 0.12-0.38. The best performance was obtained when C = 0.31, respectively EER = 5.33%.

Table 6.14 EER values with modified Manhattan metrics

Coefficient (*C*)	**EER (%)**
0	7,13
0,01	6,87
0,02	6,87
0,03	6,62
0,04	6,53
0,05	6,46
0,06	6,59
0,07	6,12
0,08	6,07
0,09	6,08
0,1	5,81
0,11	5,75
0,12	5,47

0,13	5,41	0,29	5,38	0,45	9,32
0,14	5,58	0,3	5,34	0,46	9,91
0,15	5,54	**0,31**	**5,33**	0,47	10,58
0,16	5,5	0,32	5,38	0,48	11,25
0,17	5,49	0,33	5,47	0,49	12
0,18	5,51	0,34	5,48	0,5	12,64
0,19	5,52	0,35	5,51	0,51	13,37
0,2	5,5	0,36	5,47	0,52	14,18
0,21	5,5	0,37	5,48	0,53	15,01
0,22	5,49	0,38	5,47	0,54	15,77
0,23	5,51	0,39	5,86	0,55	16,45
0,24	5,52	0,4	6,45	0,56	17,2
0,25	5,5	0,41	6,98	0,57	18,1
0,26	5,52	0,42	7,47	0,58	18,86
0,27	5,51	0,43	8,06	0,59	19,65
0,28	5,55	0,44	8,62		

Figure 6.31 shows graphically the EER values obtained for different values of the C coefficient. The values represented in the graph are those in the table above. As well as the table, the graph shows that the best performances were obtained in the range C = 0.12-0.38. The best performance was obtained when C = 0.31, respectively EER = 5.33%.

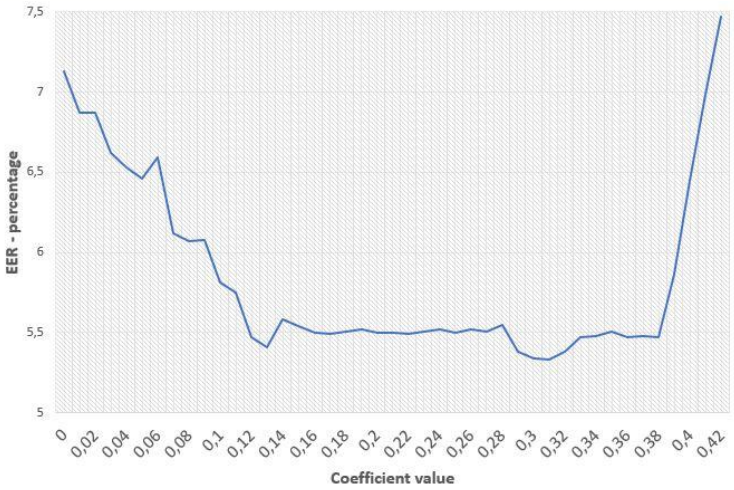


Figure 6.31 EER values with modified Manhatten metrics

Following the experiment described above, the decision was made to propose a new metric, a metric that starts from the distance of Manhattan, but which adjusts

with $.31 \times \sigma_{x_i}$. The new metric proposed in this paper is presented in formula (6.2) below:

$$d(x,y) = \sum_{i=1}^{14} (|x_i - y_i| - 0.31 \times \sigma_{x_i}) \tag{6.2}$$

Where *d(x,y)* is the distance between the two vectors x and y and σ_{x_i} is the standard deviation from x_i.

The proposed new distance metric improves performance coefficient EER of 7.13% to the value of 5.33%. This means a 25.24% improvement in performance.

6.4.2 New metric for calculating distances based on di-graphs times

In this subchapter we propose a second metric, derived from the change in the calculation of the distances between two vectors, based on the times from the di-graphs.

In the experiments performed, the following conclusions were reached:

1. The best performance is obtained using the Manhattan metric for calculating distances.

2. The best performances are obtained when analyzing the times coming only from the most frequent 12 di-graphs

3. The best performances are obtained if only 3 of the 6 times generated by a di-graph are used in the distance calculation, namely: the pressing time of the first key, the pressing time of the second key and the total time al di-graphs.

The formula that summarizes the above is at (6.3):

$$d(x,y) = \sum_{i=1}^{12} |xDU1_i - yDU1_i| + \sum_{i=1}^{12} |xDU2_i - yDU2_i| + \sum_{i=1}^{12} |xDUtotal_i - yDUtotal_i| \tag{6.3}$$

Having this calculation formula as a starting point, changes were made to it in order to obtain better performance. It has been observed that minimizing the share of total time in distance calculation generates better performance. In this context, the scaling of the total distance weight was applied by dividing the Manhattan distance by the coefficient C, as in the formula presented in (6.4), while the weight of the pressing time of the first key and the weight of the pressing time of the second key they remained the same:

$$d(x,y) = \sum_{i=1}^{12} |xDU1_i - yDU1_i| + \sum_{i=1}^{12} |xDU2_i - yDU2_i| + \sum_{i=1}^{12} \left| \frac{xDUtotal_i - yDUtotal_i}{C} \right| \tag{6.4}$$

For the coefficient C, values from 1 to 15 were assigned, but also higher values, the conclusion of the experiment being that the best performances are obtained in the range 2-7. The values obtained for EER are presented in Table 6.15.

Table 6.15 EER values with modified distance metrics

Coefficient	EER (%)
1	5,33
2	3,64
3	**3,27**
4	3,33
5	3,64
6	3,71
7	3,86
8	4,2
9	4,31
10	4,38
11	4,34
12	4,37
13	4,35
14	4,35
15	4,32
100	5,23
1000	5,54

From the values presented in the table it results that the best performance is obtained at the value of C = 3. In this case, the EER value is 3.27%. This value is the best performance obtained in the present thesis. The performance improvement with the modified formula presented in (6.5) is done by 37.47%. From the initial value of performance, calculated with the classic Manhattan metric, by EER = 5.23%, the performance reached 3.27%. The 1.96 percentage point improvement represents a 37.47% improvement in performance.

$$d(x,y) = \sum_{i=1}^{12} |xDU1_i - yDU1_i| + \sum_{i=1}^{12} |xDU2_i - yDU2_i| + \sum_{i=1}^{12} \left| \frac{xDUtotal_i - yDUtotal_i}{3} \right| \quad (6.5)$$

Figure 6.32 graphically represents the values of False Acceptance Rate (FAR) and False Rejection Rate (FRR) for the best performance obtained in the present research. The intersection, on the graph, of FAR and FRR is at the point of EER=3.27%.

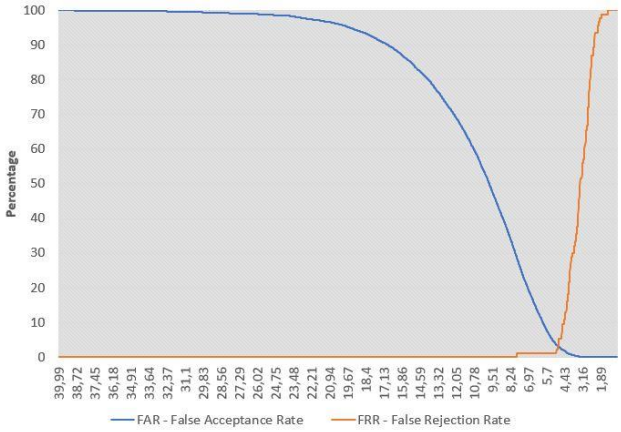


Figure 6.32 FAR and FRR graph for the best performance of this research

Figure 6.33 shows several ROC curves generated from the experiments performed for this paper and described in detail in this chapter. The best performance obtained during the research, with the proposed new metric is on the red graph. It can be seen that it is the best performance from the graph in the figure.

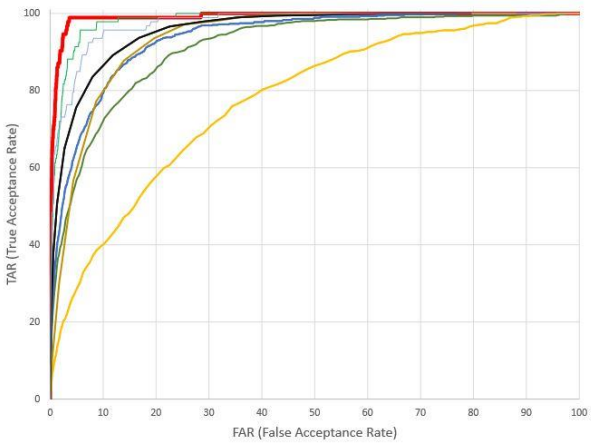


Figure 6.33 ROC curve for the best performance of this research - the red one

6.5 Proposed user pattern

This subchapter follows the O4 approach.

Considering the results obtained during this chapter, an efficient pattern will be proposed both in terms of size and time required for the calculations to combine the two solutions at the end of this chapter. The pattern will contain information about the 14 most used letters (a-z) and the 12 most used di-graphs that contain only letters (a-z). For the 14 letters, the average and the standard deviation of the pressing times of the respective letter will be retained. For the 12 di-graphs, 3 averages will be retained: DU1 average, DU2 average and Dutotal average. The proposal formulated for the retention of the pattern is represented graphically in Table 6.16 :

Table 6.16 Proposed user pattern

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Key	A	E	I	T	R	N	U	S	C	L	O	M	P	D
Key DU mean														
Key DU std. deviation														
di-graph DU1 mean														
di-graph DU2 mean														
di-graph DUtotal mean														
di-graph	IN	RE	AR	TE	DE	CA	AT	TA	ST	TI	RA	SI		

6.6 Comparison of the related works

This subchapter compares the performance obtained with the proposals made in this thesis with the performances obtained in other researches presented in the literature. Table 6.17 presents the performances obtained in other researches, the results were being published and centralized in the paper [TSA19]. The last two lines of the table are the two results obtained by applying the new metrics proposed in this paper.

The performances presented in the table vary from the best EER performance = 0.44% to EER performance = 77%. Of the 11 results presented in the table, the performance obtained in this thesis is among the top three. Analyzing the best performance, by the algorithm Kim et al. [KIM18], of EER = 0.44%, the large volume of training characters it uses is noticeable, while in this paper the analyzed volume is 1000 key / sample. The larger the sample size, the better the performance.

Table 6.17 Comparison of the related works [TSA19]

Method	Year	# of training participants	Experimental time (# of training characters)	Classifier	**EER (%)**
Monrose & Rubin [MON97]	1997	42	7 Weeks	Statistics	**77.0**
Gunetti & Picardi [GUN05]	2005	40	1–2 Months	Statistics	**15.0**
Villani et al. [VIL06]	2006	40	–	Statistics	**3.6**
Davoudi & Kabir [DAV09]	2009	21	1–2 Months	Statistics	**–**
Samura & Nishimura [SAM09]	2009	112	–	Euclidean distance	**5.3**
Park & Cho [PAR10]	2010	35	–	Statistics	**8.9**
Messerman et al. [MES11]	2011	55	12 Months	Statistics	**2.0**
Alsultan et al. [ALS16]	2016	21	– (7200 characters)	SVM and decision tree	**–**
Alsultan et al. [ALS18]	2017	25	– (7200 characters)	SVM and decision tree	**–**
Alsultan et al. [ALS17]	2017	30	– (1000 characters)	SVM and decision tree	**–**
Kim et al. [KIM18]	2018	150	– (18,000 English and 7000 Korean characters)	Five algorithms	**0.44**
Tsai & Shih (KD) [TSA19]	2018	100	2 Weeks (1000 characters)	Statistics	**20.0**

Tsai & Shih (KD & KC-Map) [TSA19]	2018	100	2 Weeks (1000 characters)	Statistics + KCMS	**15.7**
The approach proposed in this research	**2020**	**80**	**1 Session (1000 characters)**	**Modified distance for individual keys**	**5.33**
The approach proposed in this research	**2020**	**80**	**1 Session (1000 characters)**	**Modified distance for di-graphs**	**3.27**

Following the comparison of the performances with those from other researches, it can be stated that the proposals made within the present thesis bring improvements to the field, by improving the performances of the authentication algorithm based on keystroke dynamics.

Commercial continuous authentication products based on keystroke dynamic exist. In Romania, Typing DNA is a company, a start-up, that received funds of 6.2 million euros in 2020 to create a typing identity for security [STE20]. They describe their company as follows [TYP21]: ”At TypingDNA we are obsessed with passive authentication and typing biometrics. Our mission is to improve security without compromising user experience. We started research in 2014 and launched publicly in late 2016. Our technology really shines at being the most available online biometric technology - it works with any keyboard, on any device, works passively behind the scenes, and doesn’t need more than one previous sample to start working. We provide typing biometrics authentication as a service, an API that anyone can use for 2FA and fraud prevention use cases. We are present in New York, USA and Romania, EU.”

Comparing the algorithm developed in this paper and the algorithm applied by Typing DNA, the same principle is used, the collection and analysis of the typing mode and are used in the analysis of both a keystroke time and flight time [TYP21]. However, there are also differences listed below [TYP21]: (1) this paper uses the times from the most common 14 typed letters, while the solution from Typing DNA uses the times from the most common 44 keys, (2) this paper uses the time collected from di-graphs, while the solution from Typing DNA does not use sequences of 2 or more keys and (3) this paper applies the distance calculation method, while the solution from Typing DNA uses methods based on machine learning.

6.7 Conclusions

This chapter presented the experiments performed using the algorithm developed and the data collected. The experiments performed in this chapter were divided into two branches. First of all, experiments were performed to calculate the similarity between users only with the help of the times for each key, in subchapter 6.1 Experiments with the keystroke time of a single key. The results were calculated and presented using Euclidean distance (in subchapter 6.1.1), Manhattan distance (in subchapter 6.1.2), R distance (in subchapter 6.1.3) and A distance (in subchapter 6.1.4). Second, experiments were performed to calculate the similarity between two users using the times generated by the di-graphs, in subchapter 6.2 Experiments with di-graphs. Experiments were performed based on Euclidean distance (in subchapter 6.2.3), Manhattan Distance (in subchapter 6.2.4) and A distance (in subchapter

6.2.5). The subchapter 6.3 The distance between users presented the way to calculate the distance between the vectors generated for users.

In subchapter 6.4, Proposing new metrics for calculating distances between users, the two proposals for modified metrics that generate better results, as well as the performance of the Equal Error Rate (EER) indicator, were presented. O3 was addressed in this subchapter. With the help of the two metrics, the performances of the authentication algorithm have been improved by 25.24%, respectively by 37.47%. The best performance obtained with the modified metric was EER = 3.27%.

In subchapter 6.5 Proposed user pattern, the pattern user proposal resulting from the experiments and the obtained performances was presented, so that the information that retains the characteristics of a user to occupy the optimal memory space and can contribute to a fast algorithm. A structure that retains the average and standard deviation, of the keystroke time, for the most frequently used 14 letters and the average of the keystroke times of the first key, of the second key and of the total time for the most frequently used 12 di-graphs was proposed. The structure thus obtained occupies 256 bytes in memory for each user. O4 was addressed in this subchapter.

At the end of the chapter, in subchapter 6.6 Comparison of the related work, the performance obtained with the help of the proposed metrics and other results obtained in similar research in the literature were compared.

The next chapter, Conclusions and Future Works, presents the general conclusions of this thesis as well as research that can be carried out in further research presented in this thesis.

7 CONCLUSIONS AND FUTURE WORKS

The previous chapter, Experiments and results, covered the experiments performed during the present research, detailed the results obtained and parts of the algorithm used for continuous authentication in educational systems.

This chapter summarizes the conclusions drawn from the previous chapters and future research directions in this field, starting from the results presented in this paper. The author's own contributions to the field of keystroke dynamics are presented in the subchapter 7.1.1 The personal contribution: the proposal of two new metrics for calculating the distance between two vectors in order to allow the approximation of the degree of similarity between two patterns from two different users or from the same user. Also, the data collected from the 80 users about how to type on the keyboard is a contribution to the advantage of future researches because they will be available to all researchers interested in conducting investigation in the field. Another own contribution is the proposal of a pattern in order to retain the minimum necessary data about a user so to obtain performances in the continuous authentication.

The last part of this chapter, the subchapter 7.2, Future works, presents the future research directions. The field still needs to be exploited, and future research directions may bring higher performance than those currently obtained.

7.1 Conclusions

The present research aimed to approach the field of continuous authentication using free-text keystroke dynamics, especially for online education platforms. Also, at the beginning of the research, 4 objectives were formulated that were pursued throughout the research.

Each of us has a rhythm, a certain speed, a typing pattern, formed in time and unique while typing on a keyboard. We can differentiate the users of a computer, can identify them or authenticate them in a system only by capturing these details. To analyze a user's typing pattern, we need to capture and process it using an algorithm.

In order to be able to identify a certain user who would now be in front of a computer, using a keyboard, it is necessary, beforehand, to have his typing characteristics in a database. The database is needed in order to compare the typing mode captured live with the patterns of the users enrolled in the respective system, thus, helping to be identified. In other words, the mode of operation is similar to the username and password authentication. The computer users enter their username

and their password, and the system searches them in the database to compare what the user entered with what he has previously registered, in order to make a decision.

The first objective, O1, of this thesis was to collect a database with the typing mode from at least 80 users, in order to test the algorithm from O2, but also to make it available to other interested researchers. The first objective was presented in Chapter 4.

The second objective, O2, of this thesis was to implement an algorithm for authenticating the users of a computer based on the keystroke dynamics, the keyboard typing mode. The second objective was presented in Chapter 5 and in Appendix 1.

The algorithm can process and analyze the text typed on the keyboard in two situations:

1. when a user enters from the keyboard each time the same key combination - fixed text keystroke dynamics

2. when the user types different text each time, based on his freewill - free text keystroke dynamics

The present thesis addressed the second type of analysis - free text keystroke dynamics. This type was chosen because (1) the user can do any kind of computer work, and the algorithm works in the background, without the user having to take additional steps and (2) it is not explored yet so far, and the accuracy performance of user identification of current algorithms can be improved.

The keystroke dynamics authentication algorithm takes over, for each key pressed, the key code, the time when it was pressed and the time at which it was picked up. In this way, for each user we will have a long series of keys and times. By processing this input data, it is possible to identify the user. With the help of the pressing times, respectively of leaving the key, we can easily calculate the total time when a certain key has been pressed or the total time elapsed between two consecutive keys.

The data that can be analyzed within such an algorithm can be subjected to report to:

1. analysis of the characteristics of one key at a time (individual key analysis)

2. analysis of the characteristics of a pair of consecutive keys in one step (di-graph analysis)

3. analysis of the characteristics of a group of three consecutive keys at one step (tri-graph analysis)

4. generic, analysis of the characteristics of a group of n consecutive keys at one step (n-graph analysis)

The present thesis addressed the situation from points 1. one key analysis and 2. di-graph analysis from above. The reason why it was decided to approach these analyses is due to the fact that the analysis can be done on shorter text strings, without having to wait for a user to type a very long text in order to be analyzed. The analysis can be done on shorter strings because it is more likely that a single key or a pair of keys will be repeated several times in two texts to be compared shorter, the probability that a group of 3 consecutive keys, four or even more to appear in both texts.

Regardless of the analysis type of the user's typing mode (one key, di-graph, tri-graph or n-graph analysis) the input data for the authentication algorithm are time vectors (intervals when a key has been pressed, or how long it took to press the next key). The algorithm will process this input data to decide if the user who is now at the computer is the one who claims to be and can log in to the system. Time vectors are vectors of real numbers.

In order to analyze the vectors of real numbers (time vectors) and to decide their similarity, different methods can be approached: (1) distance based classifier, (2) statistical classifier -generic, (3) probability classifier, (4) clustering, (5) machine learning methods - generic, (6) neural networks, (7) fuzzy logic, (8) decision tree, (9) evolutional computing, (10) SVM - support vector machines etc.

The present thesis addressed the distance based classifier method because, although it is the first method used in this field, it is still the most used due to its best results.

The algorithms developed so far in the field of authentication based on keystroke dynamics do not have a success rate of 100%. They cannot identify or authenticate without error, even in small proportions. The errors that it can produce when authenticating a user in the system can be False Acceptance or False Rejection, when it allows the access of an impostor user or if it does not allow the access of the real user in the system. Depending on the failure rate, two performance indicators of the algorithm are generated: False Acceptance Rate (FAR) and False Rejection Rate (FRR). Generating values of the two performance indicators, their intersection on the graph brings about a much more generic performance indicator: Equal Error Rate (EER).

The third objective, O3, of this thesis was to propose at least two new metrics for calculating the distances between two vectors that generate better performance compared to the Equal Error Rate (EER) performance indicator than the classical methods. The third objective was presented at the end of Chapter 6, in subchapter 6.4 Proposing new metrics for calculating distances between users. With the two metrics, the performances of the authentication algorithm are improved by 25.24%, respectively by 37.47%. The best performance obtained with the modified metric is EER = 3.27%.

In order for the authentication algorithm based on keystroke dynamics to work efficiently and in real time, while a user types on the keyboard, it is necessary both the execution time to be as short as possible, but also the database about the pattern of enrolled users in the system to be as supple as possible. Regarding the structure of the information in the database, it needs to be the most relevant about the typing of a user, also easy to access.

The fourth objective, O4, of this thesis was to propose a data structure as efficient as possible, which should contain the most relevant information about the typing mode of a user. The fourth objective was presented at the end of Chapter 6, in subchapter 6.5 Proposed user pattern. It is proposed a structure that retains the average and standard deviation, of the keystroke time, for the most frequently used 14 letters and the average of the keystroke times of the first key, of the second key and of the total time for the most frequently used 12 di-graphs. The structure obtained occupy 256 bytes in memory for one user.

Given that the authentication method using keystroke dynamics has a certain vulnerability, a certain error rate, as well as any other authentication system, a two-step authentication is required, with two different methods, thus, the keystroke dynamics authentication method can be used successfully as the second mandatory authentication method.

This authentication method can be successfully applied as a second mandatory method in the case of authentication within online education platforms and especially during exams, when the user's identity must be confirmed throughout the session, not only once at the beginning. This additional verification, in addition to the username and password, as the first method of authentication, for example, would be required for two more reasons:

1. The educational systems with Massive Open Online Courses (MOOC) have seen a great growth from its appearance until today, reaching tens of millions of users, there are exams on these platforms with thousands of students at the same time and

2. The medical crisis generated by the SARS-CoV-2 virus in 2020 provoked unprecedented travel restrictions around the globe, and the educational system was contrived to turn into an online one.

7.1.1 The personal contributions

The personal contributions presented in this research are:

1. A free-text keystroke dynamics algorithm for continuous authentication has been developed. The algorithm can be found in Appendix 1 - Free-text keystroke dynamics algorithm for continuous authentication and it was presented in Chapter 4.

2. It was created a database with typing mode from 80 users, 410.000 key events, a total time of approximately 24 hours for the acquisition of the necessary data. Detailed in Chapter 5

3. A modify Manhattan distance metric has been proposed, calculated on the most used 14 letters. The proposed new distance metric improves performance coefficient EER from 7.13% to the value of 5.33%. This means a 25.24% improvement in performance. Details about the proposed new metric are in the subchapter 6.5 Proposing new metrics for calculating distances between users, 6.4.1 New metric for calculating distances based on individual key time.

4. A modify distance metric has been proposed, calculated on the most used 12 di-graphs. The proposed new distance metric improves performance coefficient EER from 5.23% to the value of 3.27%. This means a 37,47% improvement in performance. Details about the proposed new metric are in the subchapter 6.4 Proposing new metrics for calculating distances between users, 6.4.2 New metric for calculating distances based on di-graphs times.

5. A structure for user pattern with the efficiency of the space used but also with the premises to make the necessary calculations in a short time has been proposed. The total space occupied by such a pattern for a user is only 256 bytes (64 floats). The proposal formulated for the retention of the pattern is represented in subchapter 6.5 Proposed user pattern.

7.2 Future works

This thesis has reached its established objectives, and the conclusions presented in the previous subchapter open new possibilities to continue research in new directions, such as:

- Expanding the keystroke dynamics database by collecting data from a larger number of users;
- Expanding the database by collecting data from the 80 users in new sessions in order to research the evolution of the typing pattern over time
- Analysis of new algorithms, based on different techniques compared to calculating distances between time vectors
- Applying the metrics proposed in this paper to other databases available from other scientific research
- Analysis of the particularities of the special characters from the Romanian language, which are not found in English: Ă, Î, Â, Ș, Ț.

- Character analysis punctuation, SPACE, ENTER, TAB, BACKSPACE etc.
- Changing data collection conditions: changing the keyboard, under stress, etc.
- Analysis of the word in which the di-graph appears
- Developing keystroke dynamics authentication algorithms based on tri-graphs
- Developing keystroke dynamics authentication algorithms based on n-graphs
- Developing keystroke dynamics authentication algorithms for mobile devices, not only for classic computers and laptops
- Integration of the algorithm developed in existing and functional educational platforms. Testing it in a real educational environment, as well as improving the impact by applying the principles of Design-Based Research can pave the way for new research directions. A first platform in which the developed algorithm can be integrated is in the Moodle platform of the Politehnica University of Timisoara

ACKNOWLEDGEMENTS

This thesis was supported by the Sectoral Operational Programme Human Resources Development POSDRU/159/1.5/S/137516 financed from the European Social Fund and by the Romanian Government.

REFERENCES

References

[ACM12] Association for Computing Machinery ACM, HOW TO CLASSIFY WORKS USING ACM'S COMPUTING CLASSIFICATION SYSTEM, 2012, https://dl.acm.org/ccs

[AHM14] A. A. Ahmed and I. Traore, "Biometric recognition based on free-text keystroke dynamics," IEEE transactions on cybernetics, vol. 44, pp. 458-472, 2014.

[ALI17] Ali, M.L., Monaco, J.V., Tappert, C.C. et al. Keystroke Biometric Systems for User Authentication. J Sign Process Syst 86, 175–190 (2017). https://doi.org/10.1007/s11265-016-1114-9

[ALS16] Alsultan, Arwa & wei, Honq & Warwick, Kevin. (2016). Free-text Keystroke Dynamics Authentication for Arabic Language. IET Biometrics. 5. 10.1049/iet-bmt.2015.0101.

[ALS17] A. Alsultan, K. Warwick, H. Wei, Non-conventional keystroke dynamics for user authentication, Pattern Recognit. Lett. 89 (2017) 53–59.

[ALS18] A. Alsultan, K. Warwick, H. Wei, Improving the performance of free-text keystroke dynamics authentication by fusion, Appl. Soft Comput. 70 (2018) 1024–1033.

[ANI16] Anil K. Jain, Karthik Nandakumar, and Arun Ross. 2016. 50 years of biometric research: Accomplishments, challenges, and opportunities. Pattern Recognition Letters 79 (2016), 80 – 105.

[ARA03] L. C. F. Araújo, L. H. R. Sucupira Jr., M. G. Lizárraga, L. L. Ling, and J. B. T. Yabu-uti, "A fuzzy logic approach in typing biometrics user authentication," in Proc. 1st Indian Int. Conf. Artificial Intelligence, 2003, pp. 1038–1051.

[ARA04] Araújo, Lívia & Sucupira, Luiz & Lizarraga, Miguel & Ling, Lee & Yabu-uti, João. (2004). User Authentication through Typing Biometrics Features. 3072. 694-700. 10.1007/978-3-540-25948-0_94.

[ARW17] Arwa Alsultan, Kevin Warwick, Hong Wei, Non-conventional keystroke dynamics for user authentication, Pattern Recognition Letters, Volume 89, 2017, Pages 53-59, ISSN 0167-8655

[AVA17] Avar Pentel. 2017. Predicting Age and Gender by Keystroke Dynamics and Mouse Patterns. In Adjunct Publication of the 25th Conference on User Modeling,

Adaptation and Personalization (UMAP '17). Association for Computing Machinery, New York, NY, USA, 381–385. DOI:https://doi.org/10.1145/3099023.3099105

[AYO19] B. Ayotte, J. Huang, M. K. Banavar, D. Hou and S. Schuckers, "Fast Continuous User Authentication Using Distance Metric Fusion of Free-Text Keystroke Data," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 2019, pp. 2380-2388, doi: 10.1109/CVPRW.2019.00292.

[BAN12] Banerjee, Salil & Woodard, D.L.. (2012). Biometric Authentication and Identification Using Keystroke Dynamics: A Survey. Journal of Pattern Recognition Research. 7. 116-139. 10.13176/11.427.

[BAR06] N. Bartlow and B. Cukic, "Evaluating the Reliability of Credential Hardening through Keystroke Dynamics," 2006 17th International Symposium on Software Reliability Engineering, Raleigh, NC, 2006, pp. 117-126, doi: 10.1109/ISSRE.2006.25.

[BER02] BERGADANO, F., GUNETTI, D., AND PICARDI, C. 2002. User authentication through keystroke dynamics. ACM Transactions on Information and System Security 5, 4.

[BHA18] Bhanja, Samit & Das, Abhishek. (2018). Impact of Data Normalization on Deep Neural Network for Time Series Forecasting.

[BOU17] P. Bours and S. Brahmanpally, "Language dependent challenge-based keystroke dynamics," 2017 International Carnahan Conference on Security Technology (ICCST), Madrid, 2017, pp. 1-6, doi: 10.1109/CCST.2017.8167838.

[CAL19] Calot, Enrique & Ierache, Jorge & Hasperué, Waldo. (2019). Document Typist Identification by Classification Metrics Applying Keystroke Dynamics Under Unidealised Conditions. 19-24. 10.1109/ICDARW.2019.70136.

[CHA20] Chang, TY., Tsai, CJ., Yeh, JY. et al. New soft biometrics for limited resource in keystroke dynamics authentication. Multimed Tools Appl 79, 23295–23324 (2020). https://doi.org/10.1007/s11042-020-09042-x

[CLO12] Clow, D. (2012, April). The learning analytics cycle: closing the loop effectively. In Proceedings of the 2nd International Conference on Learning Analytics and Knowledge (pp. 134-138). ACM.

[COL99] O. Coltell, J. M. Badfa, and G. Torres, "Biometric identification system basedin keyboardfiltering," in Proc. IEE 33rd Annu. Int. Carnahan Conf. Security Technology, 1999, pp. 203–209.

[COU20] Coursera Platform, 2020, www.coursera.org

[DAN12] J. Daniel. 2012. Making Sense of MOOCs: Musings in a Maze of Myth, Paradox and Possibility. Technical Report. Korea National Open University.

http://www.tonybates.ca/wp-content/uploads/Making-Sense-of-MOOCs.pdf Retrieved February 2014

[DAS16] Dasgupta, Dipankar & Roy, Arunava & Nag, Abhijit. (2016). Toward the design of adaptive selection strategies for multi-factor authentication. Computers & Security. 63. 10.1016/j.cose.2016.09.004.

[DAV09] H. Davoudi, E. Kabir, A new distance measure for free text keystroke authentication, in: Proceedings of the 14th International CSI Computer Conference, October, 2009, pp. 570–575.

[DEN13] Deng, Yunbin & Zhong, Yu. (2013). Keystroke Dynamics User Authentication Based on Gaussian Mixture Model and Deep Belief Nets. ISRN Signal Processing. 2013. 10.1155/2013/565183.

[DOW14] Downes, S. 2008. Places to go: Connectivism & Connective Knowledge. Innovate 5 (1). http://www.innovateonline.info/index.php?view=article&id=668 January 2014

[DUN08] T. Dunstone and N. Yager. Biometric System and Data Analysis: Design, Evaluation, and Data Mining. Springer, 1 edition, 2008.

[DYF12] Dyckhoff, A. L., Zielke, D., Bültmann, M., Chatti, M. A., & Schroeder, U. (2012). Design and Implementation of a Learning Analytics Toolkit for Teachers. Educational Technology & Society, 15 (3), 58–76.

[EDX20] EdX Platform, 2020, www.edX.org

[FAB08] Fabio Roli, Luca Didaci, and GianLuca Marcialis. 2008. Adaptive Biometric Systems That Can Improve with Use. In Advances in Biometrics, NaliniK. Ratha and Venu Govindaraju (Eds.). Springer London, 447–471.

[FAB97] Fabian Monrose and Aviel Rubin. 1997. Authentication via keystroke dynamics. In Proceedings of the 4th ACM conference on Computer and communications security (CCS '97). Association for Computing Machinery, New York, NY, USA, 48–56. DOI:https://doi.org/10.1145/266420.266434

[FAW06] Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861–874. doi:10.1016/j.patrec.2005.10.010.

[FLI10] Flior, E., & Kowalski, K. (2010, April). Continuous biometric user authentication in online examinations. Paper presented at the ITNG2010 - Seventh International Conference on Information Technology: New Generations, Las Vegas, NV.

[FOR77] G. Forsen, M. Nelson, and R. Staron, Jr. "Personal attributes authentication techniques", Technical Report RADC-TR-77-333, Rome Air Development Center, October 1977.

[GUN05] D. Gunetti and C. Picardi, "Keystroke analysis of free text," ACM Transactions on Information and System Security, vol. 8, pp. 312–347, 2005

[HAB17] M. Habib and J. Alqatawna, "A Proposed Password-Free Authentication Scheme Based on a Hybrid Vein-Keystroke Approach," 2017 International Conference on New Trends in Computing Sciences (ICTCS), Amman, 2017, pp. 173-178, doi: 10.1109/ICTCS.2017.27.

[HAI00] S. Haidar, A. Abbas, and A. K. Zaidi, "A multi-technique approach for user identification through keystroke dynamics," in Proc. IEEE Int. Conf. Systems, Man, and Cybernetics, vol. 2, 2000, pp. 1336–1341.

[HOL13] Holotescu, C., Grosseck, G., Cretu V. 2013. MOOC'S ANATOMY: MICROBLOGGING AS THE MOOC'S CONTROL CENTER, The 9th International Scientific Conference eLearning and software for Education Bucharest, April 25-26, 2013

[HUA16] J. Huang, D. Hou, S. Schuckers and S. Upadhyaya, "Effects of text filtering on authentication performance of keystroke biometrics," 2016 IEEE International Workshop on Information Forensics and Security (WIFS), Abu Dhabi, 2016, pp. 1-6, doi: 10.1109/WIFS.2016.7823899.

[HUA17] J. Huang, D. Hou, S. Schuckers, T. Law and A. Sherwin, "Benchmarking keystroke authentication algorithms," 2017 IEEE Workshop on Information Forensics and Security (WIFS), Rennes, 2017, pp. 1-6, doi: 10.1109/WIFS.2017.8267670.

[IAP14a] **Iapa, A.C.** (2014), Outstanding research in MOOC and future development, Proceedings of the 10th International Scientific Conference "eLearning and Software for Education" Bucharest, Editura Universitatii Nationale de Aparare "Carol I" 2014 Volume 1, 251-254, DOI: 10.12753/2066-026X-14-035

[IAP14b] **IAPA C.**, MOOC and Learning Analytics: interaction evolution, Social Media in Academia: Research and Teaching - SMART 2014, 6, 2014

[IAP21a] **Iapa A.C.**, Cretu V.I., Modified Distance Metric That Generates Better Performance For The Authentication Algorithm Based On Free-Text Keystroke Dynamics, IEEE 15th International Symposium on Applied Computational Intelligence and Informatics, Timisoara, Romania, 2021 – *paper sent, unpublished*

[IAP21b] **Iapa A.C.**, Cretu V.I., Evaluating the performance of authentication algorithms based on keystroke dynamics used in online educational platforms, The 17th International Scientific Conference eLearning and Software for Education, Bucharest, Romania, 2021 – *paper sent, unpublished*

[IBM01] IBM Software Group. Analytics for achievement. Ottawa, Ontario, 2001. http://public.dhe.ibm.com/common/ssi/ecm/en/ytw03149caen/YTW03149CAEN. PDF.

[ILO03] Ilonen, Jarmo. (2003). Keystroke dynamics. Advanced Topics in Information processing–lecture (2003).

[IMP20] Agarwal , A., 2020 Impact Report, edX

[IVA16] Ivanova, Malinka, Holotescu, C., Grosseck, G., **Iapa, C.** "RELATIONS BETWEEN LEARNING ANALYTICS AND DATA PRIVACY IN MOOCs." The International Scientific Conference eLearning and Software for Education. Vol. 3. " Carol I" National Defence University, 2016.

[JAI05] Jain, Anil & Nandakumar, Karthik & Ross, Arun. (2005). Score normalization in multimodal biometric system. Pattern Recognition. 38. 2270-2285. 10.1016/j.patcog.2005.01.012.

[JAI99] A. K. Jain, R. Bolle, and S. Pankanti, Eds., Biometrics: Personal Identification in Networked Society, Kluwer Academic, 1999

[JAY19] Jay R. Young, Randall S. Davies, Jeffrey L. Jenkins & Isaac Pfleger (2019): Keystroke Dynamics: Establishing Keyprints to Verify Users in Online Courses, Computers in the Schools, DOI: 10.1080/07380569.2019.1565905

[JES06] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In Proceedings of the 23rd international conference on Machine learning (ICML '06). Association for Computing Machinery, New York, NY, USA, 233–240. DOI:https://doi.org/10.1145/1143844.1143874

[JOY90]R. Joyce andG. Gupta, "Identity authentication basedon keystroke latencies," Commun. ACM, vol. 33, no. 2, pp. 168–176, 1990. [BLE91] D. Bleha and M. Obaidat, "Dimensionality reduction and feature extraction applications in identifying computer users," IEEE Trans. Syst., Man, Cybern., vol. 21, no. 2, pp. 452–456, Mar.–Apr. 1991.

[JUN20] Junhong Kim, Pilsung Kang, Freely typed keystroke dynamics-based user authentication for mobile devices based on heterogeneous features, Pattern Recognition, Volume 108, 2020, 107556, ISSN 0031-3203

[KAI11] Kai Xi, Yan Tang, Jiankun Hu, Correlation Keystroke Verification Scheme for User Access Control in Cloud Computing Environment, The Computer Journal, Volume 54, Issue 10, October 2011, Pages 1632–1644, https://doi.org/10.1093/comjnl/bxr064

[KAN14] Kang, Jeonil & Nyang, Daehun & Lee, KyungHee. (2014). Two-factor face authentication using matrix permutation transformation and a user password. Information Sciences. 269. 1–20. 10.1016/j.ins.2014.02.011.

[KIL09] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," 2009 IEEE/IFIP International Conference on Dependable Systems & Networks, Lisbon, 2009, pp. 125-134, doi: 10.1109/DSN.2009.5270346.

[KIM18] J. Kim, H. Kim, P. Kang, Keystroke dynamics-based user authentication using freely typed text based on user-adaptive feature extraction and novelty detection, Appl. Soft Comput. 62 (2018) 1077–1087.

[KOC19] Kochegurova, Elena & Luneva, Elena & Gorokhova, Ekaterina. (2019). On Continuous User Authentication via Hidden Free-Text Based Monitoring: Volume 2. 10.1007/978-3-030-01821-4_8.

[LAI12] Laiu-Despău Octavian, Curiozităţi şi amuzamente ale limbii române introducere în ludolingvistică , Editura BrumaR, Timişoara: Brumar, 2012, ISBN 978-973-602-779-6

[LAT11] Latha, L. and S. Thangasamy. "Efficient approach to Normalization of Multimodal Biometric Scores." (2011).

[LEG88] J. Leggett and G. Williams, "Verifying identity via keystroke characterstics," International Journal of Man-Machine Studies, 1988.

[LIM14] Y. M. Lim, A. Ayesh and M. Stacey, "Detecting cognitive stress from keyboard and mouse dynamics during mental arithmetic", Proc. Sci. Inf. Conf. (SAI), pp. 146-152, Aug. 2014.

[LIN97] D. T. Lin, "Computer-access authentication with neural network based keystroke identity verification," in Proc. Int. Conf. Neural Networks, vol. 1, 1997, pp. 174–178.

[LOZ17] Lozhnikov, Pavel & Sulavko, Alexey & Ekaterina, Buraya & Viktor, Pisarenko. (2017). Authentication of Computer Users in Real-Time by Generating Bit Sequences Based on Keyboard Handwriting and Face Features. Voprosy kiberbezopasnosti. 24-34. 10.21681/2311-3456-2017-3-24-34.

[MAA14] A. Maas, C. Heather, C. T. Do, R. Brandman, D. Koller and A. Ng, "Offering verified credentials in massive open online courses: Moocs and technology to advance learning and learning research (ubiquity symposium)", Ubiquity, vol. 2014, no. May, pp. 2, 2014.

[MAT20] MATT WEBER, 2020, "Harvard EdCast: edX Marks the Spot", http://www.gse.harvard.edu/news-impact/2013/11/harvard-edcast-edx-marks-thespot/

[MES11] A. Messerman, T. Mustafic, S. A. Camtepe and S. Albayrak. Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics. Proceedings of IEEE International Joint Conference on Biometrics. 1–8, 2011

[MON00] F. Monrose andA. D. Rubin, "Keystroke dynamics as a biometric for authentication," Future Gen. Comput. Syst., vol. 16, no. 4, pp. 351–359, 2000.

[MON02] Monrose F, Reiter MK, Wetzel S (2002) Password hardening based on keystroke dynamics. Int J Inf Secur 1(2):69–83

[MON06] Montalvao, Jugurta & Freire, Eduardo. (2006). On the equalization of keystroke timing histograms. Pattern Recognition Letters. 27. 1440-1446. 10.1016/j.patrec.2006.01.010.

[MON97] F. Monrose and A. Rubin. Authentication via keystroke dynamics. Proceedings of the 4th ACM Conference on Computer and Communications Security. 48–56, 1997.

[MON99] F. Monrose, M. K. Reiter, andS. Wetzel, "Passwordhardening based on keystroke dynamics," in Proc. 6th ACM Conf. Computer Security, Singapore, Nov. 1999

[NAN05] Nandakumar, Jain, Ross. 2005. Score Normalization in Multimodal Biometric Systems, Pattern Recognition 38, 2270-2285.

[OBA97] M. S. Obaidat and B. Sadoun, "Verification of computer user using keystroke dynamics," IEEE Trans. Syst., Man, Cybern., vol. 27, no. 2, pp. 261–269, Mar.–Apr. 1997.

[PAR10] S. Park, J.P. Cho, User Authentication based on keystroke analysis of long free texts with a reduced number of features, in: Proceedings of IEEE International Conference on Communication Systems, Networks and Applications, July, 2010, pp. 433–435.

[PAT15] PATRO, S GOPAL & Sahu, Kishore Kumar. (2015). Normalization: A Preprocessing Stage. IARJSET. 10.17148/IARJSET.2015.2305.

[PAU19] Paulo Henrique Pisani, Abir Mhenni, Romain Giot, Estelle Cherrier, Norman Poh, et al.. Adaptive Biometric Systems: Review and Perspectives. ACM Computing Surveys, Association for Computing Machinery, 2019, 1, ff10.1145/nnnnnnn.nnnnnnnff. ffhal-02175778

[PIC12] Picciano, A. G. (2012). The Evolution of Big Data and Learning Analytics in American Higher Education. Journal of Asynchronous Learning Networks, 16(3), 9-20.

[PIL15] Pilsung Kang and Sungzoon Cho. 2015. Keystroke dynamics-based user authentication using long and free text strings from various input devices. Inf. Sci. 308, C (July 2015), 72–93. DOI:https://doi.org/10.1016/j.ins.2014.08.070

[POL00] POLEMI, D. 2000. Biometric techniques: review and evaluation of biometric techniques for identification and authentication, including an appraisal of the areas where they are most applicable. Report prepared for the European Commission DG XIII-C.4 on the Information Society Technologies (IST) (Key action 2: New Methods of Work and Electronic Commerce). Report available at: www.cordis.lu/infosec/src/stud5fr.html.

[ROB98] J. A. Robinson, V. M. Liang, J. A. Michael, andC. L. MacKenzie, "Computer user verification login string keystroke dynamics," IEEE Trans. Syst., Man, Cybern., vol. 28, no. 2, pp. 236–241, Mar.–Apr. 1998.

[ROT14] J. Roth, X. Liu and D. Metaxas, "On Continuous User Authentication via Typing Behavior," in IEEE Transactions on Image Processing, vol. 23, no. 10, pp. 4611-4624, Oct. 2014, doi: 10.1109/TIP.2014.2348802.

[RUE97] W. G. de Ru and J. H. P. Eloff, "Enhanced password authentication through fuzzy logic," IEEE Expert, vol. 17, no. 6, pp. 38–45, Nov.–Dec. 1997.

[RYB08] Rybnik, M., Tabedzki, M., & Saeed, K. (2008). A keystroke dynamics based system for user identification. Proceedings of the 2008 Seventh Computer Information Systems and Industrial Management Applications Conference (pp. 225–230). doi:10.1109/CISIM.2008.8

[SAL10] E. Al Solami, C. Boyd, A. Clark, and A. K. Islam, "Continuous Biometric Authentication: Can It Be More Practical?", IEEE Int'l Conf. on High Performance Computing and Communications (HPCC), pp. 647-652, 2010.

[SAL18] S. Salmeron-Majadas, R. S. Baker, O. C. Santos and J. G. Boticario, "A Machine Learning Approach to Leverage Individual Keyboard and Mouse Interaction Behavior From Multiple Users in Real-World Learning Scenarios," in IEEE Access, vol. 6, pp. 39154-39179, 2018, doi: 10.1109/ACCESS.2018.2854966.

[SAM09] T. Samura and H. Nishimura, "Keystroke timing analysis for individual identification in Japanese free text typing," 2009 ICCAS-SICE, Fukuoka, 2009, pp. 3166-3170.

[SCH14] Scheffel, M., Drachsler, H., Stoyanov S., & Specht, M. (2014). Quality Indicators for Learning Analytics. Educational Technology & Society, 17 (4), 117–132.

[SHU13] P. Shukla and R. Solanki, "Web based keystroke dynamics application for identifying emotional state", Int. J. Adv. Res. Comput. Commun. Eng., vol. 2, no. 11, pp. 4489-4493, Nov. 2013.

[SIE11] Siemens, G., & Long, P. (2011). Penetrating the fog: Analytics in learning and education. Educause Review, 46(5), 30-32.

[SIE12] Siemens, G. (2012, April). Learning analytics: envisioning a research discipline and a domain of practice. In Proceedings of the 2nd International Conference on Learning Analytics and Knowledge (pp. 4-8). ACM.

[SOL11] E. A. Solami, C. Boyd, A. Clark and I. Ahmed, "User-representative feature selection for keystroke dynamics," 2011 5th International Conference on Network and System Security, Milan, 2011, pp. 229-233, doi: 10.1109/ICNSS.2011.6060005.

[SPI75] R. Spillane, "Keyboard Apparatus for Personal Identification", IBM Technical Disclosure Bulletin, vol. 17, no. 3346, 1975.

[STE10] D. Stefan and D. Yao. Keystroke-Dynamics Authentication Against Synthetic Forgeries. In International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010.

[STE20] Stefan Koritar. (2020). Romanian startup Typing DNA raises €6.2 million in Series A funding to create 'typing identity' for security (2020).

[SZE03] G. Szekely. E-statistics: The energy of statistical samples. ´ Bowling Green State University, Department of Mathematics and Statistics Technical Report, 3(05):1–18, 2003. 3

[TAB14] Tabak, John (2014), Geometry: The Language of Space and Form, Facts on File math library, Infobase Publishing, p. 150, ISBN 9780816068760

[TEH13] Teh, Pin Shen & Teoh, Andrew & Yue, Shigang. (2013). A Survey of Keystroke Dynamics Biometrics. TheScientificWorldJournal. 2013. 408280. 10.1155/2013/408280.

[TSA14] Tsai CJ, Chang TY, Cheng PC, Lin JH (2014) Two novel biometric features in keystroke dynamics authentication systems for touch screen devices. Sec Commun Netw 7(4):750–758

[TSA19] Tsai, Cheng-Jung & Shih, Kuen-Jhe. (2019). Mining a new biometrics to improve the accuracy of keystroke dynamics-based authentication system on free-text. Applied Soft Computing. 80. 10.1016/j.asoc.2019.03.033.

[TYP21] TypingDNA, Continuous authentication for desktops & laptops with keystroke dynamics technology, typingdna.com, February 2021

[UMP85] D. Umphress and G. Williams, "Identity Verification through Keyboard Characteristics", Int'l J. Man-Machine Studies, Vol. 23, No. 3, pp. 263-273, 1985.

[VAC07] J. R. Vacca. Biometric Technologies and Verification Systems. Butterworth-Heinemann, 1 edition, 2007.

[VAN20] Vandenbosch, B., Most Popular Courses of 2020: A Year of Mental Health, Contract Tracing, and Job-Relevant Skills, Coursera Blog.

[VIL06] M. Villani, C. Tappert, G. Ngo, J. Simone, H.S. Fort, S.H. Cha, Keystroke biometric recognition studies on long-text input under ideal and applicationoriented conditions, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshop, June, 2006, pp. 39–46.

[WON01] F. W. M. H. Wong, A. S. M. Supian, A. F. Ismail, L. W. Kin, andO. C. Soon, "Enhanceduser authentication through typing biometrics with artificial neural networks andk-nearest neighbor algorithm," in Conf. Rec. 35th Asilomar Conf. Signals, Syst., Comput., vol. 2, 2001, pp. 911–915.

[YUE04] Yu, Enzhe & Cho, Sungzoon. (2004). Keystroke dynamics identity verification - Its problems and practical solutions. Computers & Security. 23. 428-440. 10.1016/j.cose.2004.02.004.

[ZAC10] R. Zack, C. Tappert, and S. Cha, "Performance of a long-text-input keystroke biometric authentication system using an improved k-nearest-neighbor classification method", IEEE Int'l Conf. on Biometrics: Theory Applications and Systems (BTAS), pp. 1-6, 2010.

[ZHO12] Y. Zhong, Y. Deng and A. K. Jain, "Keystroke dynamics for user authentication," 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Providence, RI, 2012, pp. 117-123, doi: 10.1109/CVPRW.2012.6239225.

[ZHO15] Zhong, Yu & Deng, Yunbin. (2015). A Survey on Keystroke Dynamics Biometrics: Approaches, Advances, and Evaluations. 10.15579/gcsr.vol2.ch1.

[ZIL98] Zilberman, A.G.: Security method and apparatus employing authentication by keystroke dynamics (1998) United States Patent 6,442,692.

APPENDIX 1 – THE ALGORITHM

The keystroke dynamics authentication algorithm

```
#include <stdio.h>
#include<stdlib.h>
#include <string.h>
#include <math.h>

#define MAX 500000
FILE *f,*m, *g,*h,*k,*l, *o,*p,*q;
double
oneMeanDU=0,oneMeanUD=0,diMeanDUtotal=0,triMeanDUtotal=0,fourMeanDUtotal
=0,oneStdDevDU=0,oneStdDevUD=0,diStdDevDUtotal=0,triStdDevDUtotal=0,fourS
tdDevDUtotal=0;

typedef struct{
  char key[20];
  int keyCode;
}keyAndKeyCode;
keyAndKeyCode keyAndKeyCodes[100];

typedef struct
{
  int nr;
  int keyCode;
  float dist[100]; //distributia intervalelor de timp /10 0=0-9, 1=10-19, 2=20-29,
..., 99=990-999
  double mean;
  double stdDev;
}distribution;
distribution distributions[230];

typedef struct
{
  char user[20];
  float distance[5000]; //distanta la fiecare dintre utilizatorii din lista
  distribution distributions[230];//pe fiecare litera
}pattern;
pattern patterns[5000];
int nPatterns=0;
pattern patternsR[5000];

typedef struct{
  float DU;
```

```
  float UD;
} akd;
akd allKeysDistribution[120];

typedef struct {
    int letter;         //key code
    int event;          //0 - key down, 1 - key up
    int timestamp;      // time of event
}keyEvent;
keyEvent keyEvents[MAX];

typedef struct {
    int letter;   //Key Code
    float DU, UDprev, UDnext; //DU = keystroke time, UDprev - previous flight time,
UDnext - next flight time
    int U,D; //U - Up timestamp, D - Down timestamp
    int letterPrev, letterNext; //Key Code of previous and next keys
    char key[20],keyPrev[20],keyNext[20]; //Keys
}onegraph;
onegraph oneGraphs[MAX];
onegraph allUsersOneGraphs[MAX];

typedef struct {
    int letter1, letter2;
    float DUtotal, DU1, DU2, DD, UU, UD;
    char key1[20],key2[20];
    char word[50];

}digraph;
digraph diGraphs[MAX];

typedef struct{
  int letter1,letter2;
  double meanDUtotal, meanDU1, meanDU2, meanDD, meanUU, meanUD;
  int nr;
  double stdDevDUtotal, stdDevDU1, stdDevDU2, stdDevDD, stdDevUU, stdDevUD;
  float minDUtotal, maxDUtotal;
 }userDiPattern;

 typedef struct{
   char user[20];
   userDiPattern pattern[10000];
   int nPattern;
   int sampleSize;
   float distance[5000];
 }diPattern;

 diPattern diPatterns[5000];
 int nDiPatterns=0;
 diPattern allDiPatterns;
```

```
typedef struct {
    int letter1, letter2, letter3;
    float DUtotal, DU1, DU2, DU3, UD1, UD2;
    float D1D2, D1D3, D2D3, U1U2, U1U3, U2U3, D1U2, D2U3,U1D3;
    char key1[20],key2[20],key3[20];
    char word[50];
}trigraph;
trigraph triGraphs[MAX];

typedef struct {
    int letter1, letter2, letter3, letter4;
    float DUtotal, DU1, DU2, DU3, DU4, UD1, UD2, UD3;
    char key1[20],key2[20],key3[20],key4[20];
    char word[50];
}fourgraph;
fourgraph fourGraphs[MAX];

int openFiles()
{
  int i;
  f=fopen("keysAndTheirKeyCodes.txt","r");
  g=fopen("oneGraphsAllUsers.txt","w");
  h=fopen("diGraphsAllUsers.txt","w");
  k=fopen("triGraphsAllUsers.txt","w");
  l=fopen("fourGraphsAllUsers.txt","w");
  o=fopen("meanAndStdDevAllUsers.txt","w");

fprintf(o,"user\toneMeanDU\toneStdDevDU\toneMeanUD\toneStdDevUD\tdiMeanDUt
otal\tdiStdDevDUtotal\ttriMeanDUtotal\ttriStdDevDUtotal\tfourMeanDUtotal\tfourStd
DevDUtotal\n");
  for(i=0;i<100;i++)
  {
    fscanf(f,"%s%d",keyAndKeyCodes[i].key,&keyAndKeyCodes[i].keyCode);
  }
  fclose(f);
  p=fopen("keysDistributionMeanAndStdDevAllUsers.txt","w");
  fprintf(p,"user\texp");
  for(i=0;i<100;i++)
  {
    fprintf(p,"\t%s",keyAndKeyCodes[i].key);
  }
  q=fopen("KeysDistributionAllUsers.txt","w");
  fprintf(q,"user\texp");
  for(i=0;i<100;i++)
  {
    fprintf(q,"\t%d",i*10);
  }
  fprintf(q,"\nuser\texp");
  for(i=0;i<120;i++)
  {
    fprintf(q,"\t%d",(i-20)*10);
```

```
  }
  f=fopen("keyEventsListAllUsers.txt","r");
  return 1;
}

int keycodeToKey(char key[],int letter)
{
  int i;
  for(i=0;i<100;i++)
  {
    if(keyAndKeyCodes[i].keyCode==letter)
    {
      strcpy(key,keyAndKeyCodes[i].key);
      return 1;
    }
  }
  return 0;
}

void deleteOneGraph(int x,int n)
{
  int i;
  for(i=x;i<n-1;i++)
  {
    oneGraphs[i]=oneGraphs[i+1];
  }
}

int constructWord(char word[],int x,int nr)
{
  int i;
  char construct[50]="";
  for(i=x;i<x+nr;i++)
    if(oneGraphs[i].letter<65 || oneGraphs[i].letter>90)
      {
        strcpy(word,"");
        return 0;
      }
  i=x;
  while(i>=0 && oneGraphs[i].letter>=65 && oneGraphs[i].letter<=90)
  {
    i--;
  }
  i++;
  while(oneGraphs[i].letter>=65 && oneGraphs[i].letter<=90 &&
strlen(construct)<49)
  {
    strcat(construct,oneGraphs[i].key);
    i++;
  }
  strcpy(word,construct);
```

```
  return 1;
}

int meanAndStdDev(int n)
{
  int i;
  for(i=0;i<n;i++)
  {
    oneMeanDU=oneMeanDU+oneGraphs[i].DU;
    if(i<n-1)
    {
      oneMeanUD=oneMeanUD+oneGraphs[i].UDnext;

      diMeanDUtotal+=diGraphs[i].DUtotal;
    }
    if(i<n-2)
      triMeanDUtotal+=triGraphs[i].DUtotal;
    if(i<n-3)
      fourMeanDUtotal+=fourGraphs[i].DUtotal;
  }
  oneMeanDU=oneMeanDU/n;
  oneMeanUD/=(n-1);
  diMeanDUtotal/=(n-1);
  triMeanDUtotal/=(n-2);
  fourMeanDUtotal/=(n-3);
  for(i=0;i<n;i++)
  {
    oneStdDevDU+=pow(oneGraphs[i].DU-oneMeanDU,2);
    if(i<n-1)
    {
      oneStdDevUD+=pow(oneGraphs[i].UDnext-oneMeanUD,2);

      diStdDevDUtotal+=pow(diGraphs[i].DUtotal-diMeanDUtotal,2);
    }
    if(i<n-2)
      triStdDevDUtotal+=pow(triGraphs[i].DUtotal-triMeanDUtotal,2);

    if(i<n-3)
      fourStdDevDUtotal+=pow(fourGraphs[i].DUtotal-fourMeanDUtotal,2);
  }
  oneStdDevDU=sqrt(oneStdDevDU/n);
  oneStdDevUD=sqrt(oneStdDevUD/(n-1));
  diStdDevDUtotal=sqrt(diStdDevDUtotal/(n-1));
  triStdDevDUtotal=sqrt(triStdDevDUtotal/(n-2));
  fourStdDevDUtotal=sqrt(fourStdDevDUtotal/(n-3));

fprintf(o,"\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\n",oneM
eanDU,oneStdDevDU,
oneMeanUD,oneStdDevUD,diMeanDUtotal,diStdDevDUtotal,triMeanDUtotal,triStdDev
DUtotal,fourMeanDUtotal,fourStdDevDUtotal);
  return 1;
```

```
}

int ZNormalization(int n)
{
  int i;
  meanAndStdDev(n);
  for(i=0;i<n;i++)
  {
    oneGraphs[i].DU=fabs(oneGraphs[i].DU-oneMeanDU)/oneStdDevDU;
    oneGraphs[i].UDnext=(oneGraphs[i].UDnext-oneMeanUD)/oneStdDevUD;
    oneGraphs[i].UDprev=(oneGraphs[i].UDprev-oneMeanUD)/oneStdDevUD;
  }
  return 1;
}

int minMaxNormalization(int n)
{
  int i;
  float minDU,maxDU,minUD, maxUD;
  for(i=0;i<n;i++)
  {
    if(i==0)
    {
      minDU=oneGraphs[i].DU;
      maxDU=oneGraphs[i].DU;
      minUD=oneGraphs[i].UDnext;
      maxUD=oneGraphs[i].UDnext;
    }
    if(minDU > oneGraphs[i].DU)
    {
      minDU=oneGraphs[i].DU;
    }
    if(maxDU < oneGraphs[i].DU)
    {
      maxDU=oneGraphs[i].DU;
    }
    if(i!=n-1)
    {
      if(minUD > oneGraphs[i].UDnext)
      {
        minUD=oneGraphs[i].UDnext;
      }
      if(maxUD < oneGraphs[i].UDnext)
      {
        maxUD=oneGraphs[i].UDnext;
      }
    }
  }
  for(i=0;i<n;i++)
  {
    oneGraphs[i].DU=(oneGraphs[i].DU-minDU)/(maxDU-minDU);
```

```
    oneGraphs[i].UDnext=(oneGraphs[i].UDnext-minUD)/(maxUD-minUD);
    oneGraphs[i].UDprev=(oneGraphs[i].UDprev-minUD)/(maxUD-minUD);
  }
  return 1;
}

int copyInPattern(char user[])
{
  int i;
  strcpy(patterns[nPatterns].user,user);
  for(i=0;i<230;i++)
  {
    patterns[nPatterns].distributions[i]=distributions[i];
  }
  nPatterns++;
  return 1;
}

float calculateEuclidianDistances()
{
  int i,j,k;
  float dist,max=0,distStdDev;
  for(i=0;i<nPatterns;i++)
  {
    for(j=i;j<nPatterns;j++)
    {
      dist=0;
      distStdDev=0;
      for(k=65;k<=90;k++)
       if(k==65 || k==69 || k==73 || k==84 || k==82 || k==78 || k==85 || k==83
|| k==67 || k==76 || k==79 || k==77 || k==80 || k==68)
        {if(patterns[i].distributions[k].mean!=0 &&
patterns[j].distributions[k].mean!=0)
          {
            dist=dist+pow(patterns[i].distributions[k].mean-
patterns[j].distributions[k].mean,2);
            distStdDev=distStdDev+pow(patterns[i].distributions[k].stdDev-
patterns[j].distributions[k].stdDev,2);
          }
        if(k==57)
          k=64;
        }
      dist=sqrt(dist);
      distStdDev=sqrt(distStdDev);
      patterns[i].distance[j]=dist;
      patterns[j].distance[i]=dist;
      if(max<dist)
        max=dist;
    }
  }
  return max;
```

```
}

float calculateManhattanDistances(float coefficient)
{
  int i,j,k;
  float dist,max=0, distStdDev;
  for(i=0;i<nPatterns;i++)
  {
    for(j=0;j<nPatterns;j++)
    {
      dist=0;
      distStdDev=0;
      for(k=65;k<=90;k++)
      {
        if(k==65 || k==69 || k==73 || k==84 || k==82 || k==78 || k==85 ||
k==83 || k==67 || k==76 || k==79 || k==77 || k==80 || k==68)
        if(patterns[i].distributions[k].mean!=0 &&
patterns[j].distributions[k].mean!=0)
          {
            dist=dist+fabs(patterns[i].distributions[k].mean-
patterns[j].distributions[k].mean)-coefficient*patterns[i].distributions[k].stdDev;
            distStdDev=distStdDev+fabs(patterns[i].distributions[k].stdDev-
patterns[j].distributions[k].stdDev);
          }
          if(k==57)
            k=64;
        }
      patterns[i].distance[j]=dist;
      if(max<dist)
            max=dist;
    }
  }
  return max;
}

float calculateBhattacharyyaDistances()
{
  int i,j,k;
  float dist,max=0;
  for(i=0;i<nPatterns;i++)
  {
    for(j=i;j<nPatterns;j++)
    {
      dist=0;
      for(k=48;k<=90;k++)
        {if(patterns[i].distributions[k].mean!=0 &&
patterns[j].distributions[k].mean!=0)
          {
          dist=dist+sqrt(patterns[i].distributions[k].mean*patterns[j].distributions[k].
          mean);
          }
```

```
        //dist=log(dist);
      patterns[i].distance[j]=dist;
      patterns[j].distance[i]=dist;
      if(max<dist)
        max=dist;
      if(k==57)
        k=64;
        }
    }
  }
  return max;
}

float calculateRdistances()
{
  int i,j,k,k2,sw;
  float dist,max=0;
  distribution aux;
  for(i=0;i<nPatterns;i++)
  {
    for(k=0;k<223;k++)
    {
      patterns[i].distributions[k].keyCode=k;
    }
    patternsR[i]=patterns[i];
    sw=0;
    while(sw==0)
    {
      sw=1;
      for(j=65;j<=90;j++)
      {
        if(patternsR[i].distributions[j].mean < patternsR[i].distributions[j+1].mean)
        {
          aux=patternsR[i].distributions[j];
          patternsR[i].distributions[j]=patternsR[i].distributions[j+1];
          patternsR[i].distributions[j+1]=aux;
          sw=0;
        }
      }
    }
  }
  for(i=0;i<nPatterns;i++)
  {
    for(j=i;j<nPatterns;j++)
    {
      dist=0;
      for(k=65;k<=90;k++)
          for(k2=65;k2<=90;k2++)
          if(patternsR[i].distributions[k].keyCode==patternsR[j].distributions[k2].key
          Code)
          {
```

```
          if(patternsR[i].distributions[k].mean!=0 &&
patterns[j].distributions[k2].mean!=0)
          {
            dist=dist+abs(k2-k);
          }
          break;
        }
      patternsR[i].distance[j]=dist;
      patternsR[j].distance[i]=dist;
      if(max<dist)
        max=dist;
    }
  }
  return max;
}

float calculateADistances(float t)
{
  int i,j,k;
  float dist,max=0;
  for(i=0;i<nPatterns;i++)
  {
    for(j=i;j<nPatterns;j++)
    {
      dist=0;
      for(k=65;k<=90;k++)
      {
        if(k==65 || k==69 || k==73 || k==84 || k==82 || k==78 || k==85 ||
k==83 || k==67 || k==76 || k==79 || k==77 || k==80 || k==68)
        if(patterns[i].distributions[k].mean!=0 &&
patterns[j].distributions[k].mean!=0)
          {
          if(patterns[i].distributions[k].mean > patterns[j].distributions[k].mean)
            {
              if(patterns[i].distributions[k].mean/patterns[j].distributions[k].mean < t)
                dist++;
            }
          else
          {
            if(patterns[j].distributions[k].mean/patterns[i].distributions[k].mean < t)
                dist++;
          }
          }
        if(k==57)
          k=64;
      }
      dist=1-dist/37; //27
      patterns[i].distance[j]=dist;
      patterns[j].distance[i]=dist;
      if(max<dist)
            max=dist;
```

```
    }
   }
   return max;
}

int calculateFARandFRR(float max, char file[])
{
   FILE *d;
   d=fopen(file,"w");
   int i,j,k, sw=0;
   float ERR=0, FAR=0, FRR=0, TAR=0, TRR=0, TA=0, TR=0, FA=0, FR=0, iERR=0,
coeff=1;
   if(max<=10)
     coeff=1000;
   for(i=0;i<=(int)max*(int)coeff;i++)
   {
     TA=0, TR=0, FA=0, FR=0;
     for(j=0;j<nPatterns;j++)
     {
       for(k=j+1;k<nPatterns;k++)
       {
         if(patterns[j].user[4]==patterns[k].user[4] &&
patterns[j].user[5]==patterns[k].user[5] &&
patterns[j].user[6]==patterns[k].user[6] &&
patterns[j].user[7]==patterns[k].user[7])
         {
           if(patterns[j].distance[k]<(float)i/coeff)
             TA++;
           else
             FR++;
         }
         else
         {
           if(patterns[j].distance[k]<(float)i/coeff)
             FA++;
           else
             TR++;
         }
       }
     }
     FAR=FA/(FA+TR)*100;
     FRR=FR/(FR+TA)*100;
     TAR=TA/(TA+FR)*100;
     TRR=TR/(TR+FA)*100;
     if(FAR>FRR && sw==0)
     {
       ERR=FAR;
       iERR=i;
       sw=1;
     }
     fprintf(d,"\n%.3f\t%.2f\t%.2f\t%.2f\t%.2f",(float)i/coeff,FAR,FRR,TAR,TRR);
```

```
  }
  fprintf(d,"\nERR= %.2f in %.2f",ERR, (float)iERR/coeff);
  printf("EER=\t%.2f\t%.2f\n",ERR, (float)iERR/coeff);
  fclose(d);
  return 1;
}

int RcalculateFARandFRR(float max, char file[])
{
  FILE *d;
  d=fopen(file,"w");
  int i,j,k, sw=0;
  float ERR=0, FAR=0, FRR=0, TAR=0, TRR=0, TA=0, TR=0, FA=0, FR=0, coeff=1,
iERR=0;
  if(max<=10)
    coeff=1000;
  for(i=1;i<(int)max*(int)coeff;i++)
  {
    TA=0, TR=0, FA=0, FR=0;
    for(j=0;j<nPatterns;j++)
    {
      for(k=j+1;k<nPatterns;k++)
      {
        if(patternsR[j].user[4]==patternsR[k].user[4] &&
patternsR[j].user[5]==patternsR[k].user[5] &&
patternsR[j].user[6]==patternsR[k].user[6] &&
patternsR[j].user[7]==patternsR[k].user[7])
        {
          if(patternsR[j].distance[k]<(float)i/coeff)
            TA++;
          else
            FR++;
        }
        else
        {
          if(patternsR[j].distance[k]<(float)i/coeff)
            FA++;
          else
            TR++;
        }
      }
    }
    FAR=FA/(FA+TR)*100;
    FRR=FR/(FR+TA)*100;
    TAR=TA/(TA+FR)*100;
    TRR=TR/(TR+FA)*100;
    if(FAR>FRR && sw==0)
    {
      ERR=FAR;
      iERR=i;
      sw=1;
```

```
    }
    fprintf(d,"\n%.2f\t%.2f\t%.2f\t%.2f\t%.2f",(float)i/coeff,FAR,FRR,TAR,TRR);
  }
  fprintf(d,"\nERR= %.2f in %.2f",ERR, iERR/coeff);
  printf("%.2f\t%.2f\n",ERR, iERR/coeff);
  fclose(d);
  return 1;
}

int writeDistances()
{
  int i,j,k;
  FILE *d;
  d=fopen("oneEuclidianDistances.txt","w");
  fprintf(d,"users\t");
  for(i=0;i<nPatterns;i++)
  {
    fprintf(d,"%s\t",patterns[i].user);
  }
  fprintf(d,"\n");
  for(i=0;i<nPatterns;i++)
  {
    fprintf(d,"%s\t",patterns[i].user);
    for(j=0;j<nPatterns;j++)
    {
      fprintf(d,"%.2f\t",patterns[i].distance[j]);
    }
    fprintf(d,"\n");
  }
  fclose(d);
  return 1;
}

int keyDistribution(int n, char user[])
{
  int i,j;
  for(i=0;i<230;i++)
      {
        distributions[i].nr=0;
        distributions[i].mean=0;
        distributions[i].stdDev=0;
        for(j=0;j<100;j++)
        {
          distributions[i].dist[j]=0;
        }
      }
  for(i=0;i<120;i++)
        {
          allKeysDistribution[i].DU=0;
          allKeysDistribution[i].UD=0;
        }
```

```
for(i=0;i<n;i++)
{
  distributions[oneGraphs[i].letter].nr++;
  distributions[oneGraphs[i].letter].mean+=oneGraphs[i].DU;
  if(oneGraphs[i].DU<1000)
  {
    distributions[oneGraphs[i].letter].dist[(int)oneGraphs[i].DU/10]++;
    allKeysDistribution[(int)oneGraphs[i].DU/10].DU++;
  }
  if(oneGraphs[i].UDnext<1000 && oneGraphs[i].UDnext>-200)
  {
    allKeysDistribution[(int)oneGraphs[i].UDnext/10+20].UD++;
  }
}
for(i=0;i<230;i++)
{
  if(distributions[i].nr!=0)
    distributions[i].mean=distributions[i].mean/distributions[i].nr;
}
for(i=0;i<n;i++)
{
  distributions[oneGraphs[i].letter].stdDev+=pow(oneGraphs[i].DU-
distributions[oneGraphs[i].letter].mean,2);
}
for(i=0;i<230;i++)
{
  if(distributions[i].nr!=0)
  distributions[i].stdDev=sqrt((float)distributions[i].stdDev/distributions[i].nr);
}
fprintf(p,"\n%s\tNr",user);
for(i=0;i<100;i++)
{
  fprintf(p,"\t%d",distributions[keyAndKeyCodes[i].keyCode].nr);
}
fprintf(p,"\n%s\tMean",user);
for(i=0;i<100;i++)
{
  fprintf(p,"\t%.2f", distributions[keyAndKeyCodes[i].keyCode].mean);
}
fprintf(p,"\n%s\tStdDev",user);
for(i=0;i<100;i++)
{
  fprintf(p,"\t%.2f",distributions[keyAndKeyCodes[i].keyCode].stdDev);
}
fprintf(q,"\n%s\tDU",user);
for(i=0;i<100;i++)
{
  fprintf(q,"\t%.2f",allKeysDistribution[i].DU);
}
fprintf(q,"\n%s\tUD",user);
for(i=0;i<120;i++)
```

```
  {
    fprintf(q,"\t%.2f",allKeysDistribution[i].UD);
    }
  copyInPattern(user);
  return 1;
}

int constructOneGraphs(int n, char user[])
{
  int letter,event,timestamp;
  int i;
  n=0;
  do
  {
    fscanf(f,"%d",&letter);
    if(letter==-1)
     {
       break;
     }
    fscanf(f,"%d%d",&event,&timestamp);
    if(event==1)
    {
     if(n!=0)
     {
      for(i=n-1;i>=0;i--)
      {
        if(oneGraphs[i].letter==letter)
        {
          oneGraphs[i].U=timestamp;
          while(oneGraphs[i].letter==oneGraphs[i-1].letter && oneGraphs[i-1].U==0)
          {
            deleteOneGraph(i-1,n);
            i--;
            n--;
          }
          break;
        }
      }
     }
    }
    if(event==0)
    {
      oneGraphs[n].letter=letter;
      keycodeToKey(oneGraphs[n].key,letter);
      oneGraphs[n].D=timestamp;
      oneGraphs[n].U=0;
      n++;
    }
  }while(letter!=-1);
 n--;
  for(i=0;i<n;i++) //construct oneGraphs
```

```
 {
  oneGraphs[i].DU= oneGraphs[i].U - oneGraphs[i].D;
  if(oneGraphs[i].DU>999)
    oneGraphs[i].DU=999;
  if(oneGraphs[i].DU<0)
  {
     deleteOneGraph(i,n);
     n--;
  }
  if(i==0)
  {
    oneGraphs[i].UDprev=0;
    oneGraphs[i].letterPrev=0;
    strcpy(oneGraphs[i].keyPrev,"");
    oneGraphs[i].UDnext=oneGraphs[i+1].D-oneGraphs[i].U;
    oneGraphs[i].letterNext=oneGraphs[i+1].letter;
    strcpy(oneGraphs[i].keyNext,oneGraphs[i+1].key);
   }
   else
   {
    if(i==n-1)
    {
      oneGraphs[i].UDprev=oneGraphs[i].D-oneGraphs[i-1].U;
      oneGraphs[i].letterPrev=oneGraphs[i-1].letter;
      strcpy(oneGraphs[i].keyPrev,oneGraphs[i-1].key);
      oneGraphs[i].UDnext=0;
      oneGraphs[i].letterNext=0;
      strcpy(oneGraphs[i].keyNext,"");
     }
     else
     {
       oneGraphs[i].UDprev=oneGraphs[i].D-oneGraphs[i-1].U;
       oneGraphs[i].letterPrev=oneGraphs[i-1].letter;
       strcpy(oneGraphs[i].keyPrev,oneGraphs[i-1].key);
       oneGraphs[i].UDnext=oneGraphs[i+1].D-oneGraphs[i].U;
       oneGraphs[i].letterNext=oneGraphs[i+1].letter;
       strcpy(oneGraphs[i].keyNext,oneGraphs[i+1].key);
     }
    }
    if(oneGraphs[i].UDnext<-199)
      oneGraphs[i].UDnext=-199;
    if(oneGraphs[i].UDnext>999)
      oneGraphs[i].UDnext=999;
    if(oneGraphs[i].UDprev<-199)
      oneGraphs[i].UDprev=-199;
    if(oneGraphs[i].UDprev>999)
      oneGraphs[i].UDprev=999;
  }
  return n;
}
```

```
int constructDiGraphs(int n, char user[])
{
  int i;
  for(i=0;i<n-1;i++) //construct diGraphs
          {
            diGraphs[i].letter1=oneGraphs[i].letter;
            diGraphs[i].letter2=oneGraphs[i+1].letter;
            strcpy(diGraphs[i].key1, oneGraphs[i].key);
            strcpy(diGraphs[i].key2, oneGraphs[i+1].key);
            diGraphs[i].DU1=oneGraphs[i].DU/100;
            diGraphs[i].DU2=oneGraphs[i+1].DU/100;
            diGraphs[i].UD=oneGraphs[i+1].UDprev/100;
            diGraphs[i].DD=diGraphs[i].UD+diGraphs[i].DU1;
             diGraphs[i].UU=diGraphs[i].UD+diGraphs[i].DU2;
          diGraphs[i].DUtotal=(diGraphs[i].DU1+diGraphs[i].UD+diGraphs[i].DU2)/3;
            constructWord(diGraphs[i].word,i,2);
          fprintf(h,"%s\t%s\t%d\t%d\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%s\n",d
          iGraphs[i].key1,diGraphs[i].key2,diGraphs[i].letter1,diGraphs[i].letter2,diGr
          aphs[i].DU1,diGraphs[i].DU2,diGraphs[i].UD,diGraphs[i].DD,diGraphs[i].UU,
          diGraphs[i].DUtotal,diGraphs[i].word);
           }
  return 1;
}

int constructDiPattern(char user[], int n)
{
  int i,j;
  strcpy(diPatterns[nDiPatterns].user,user);
  diPatterns[nDiPatterns].sampleSize=n-1;
  for(i=0;i<n-1;i++)
  {
    for(j=0;j<diPatterns[nDiPatterns].nPattern;j++)
    {
      if(diGraphs[i].letter1 == diPatterns[nDiPatterns].pattern[j].letter1 &&
diGraphs[i].letter2 == diPatterns[nDiPatterns].pattern[j].letter2)
        break;
    }
    if((diGraphs[i].letter1<=90 && diGraphs[i].letter1>=65)  &&
(diGraphs[i].letter2<=90 && diGraphs[i].letter2>=65))
    {
        diPatterns[nDiPatterns].pattern[j].letter1=diGraphs[i].letter1;
        diPatterns[nDiPatterns].pattern[j].letter2=diGraphs[i].letter2;
        diPatterns[nDiPatterns].pattern[j].nr++;
        diPatterns[nDiPatterns].pattern[j].meanDUtotal+=diGraphs[i].DUtotal;
        diPatterns[nDiPatterns].pattern[j].meanDU1+=diGraphs[i].DU1;
        diPatterns[nDiPatterns].pattern[j].meanDU2+=diGraphs[i].DU2;
        diPatterns[nDiPatterns].pattern[j].meanDD+=diGraphs[i].DD;
        diPatterns[nDiPatterns].pattern[j].meanUU+=diGraphs[i].UU;
        diPatterns[nDiPatterns].pattern[j].meanUD+=diGraphs[i].UD;
        if(diPatterns[nDiPatterns].pattern[j].minDUtotal>diGraphs[i].DUtotal ||
j==diPatterns[nDiPatterns].nPattern)
```

```
        diPatterns[nDiPatterns].pattern[j].minDUtotal=diGraphs[i].DUtotal;
      if(diPatterns[nDiPatterns].pattern[j].maxDUtotal<diGraphs[i].DUtotal ||
j==diPatterns[nDiPatterns].nPattern)
        diPatterns[nDiPatterns].pattern[j].maxDUtotal=diGraphs[i].DUtotal;
      if(j==diPatterns[nDiPatterns].nPattern)
        diPatterns[nDiPatterns].nPattern++;
    }
  }
  for(j=0;j<diPatterns[nDiPatterns].nPattern;j++)
  {
diPatterns[nDiPatterns].pattern[j].meanDUtotal/=diPatterns[nDiPatterns].pattern[j].
nr;
diPatterns[nDiPatterns].pattern[j].meanDU1/=diPatterns[nDiPatterns].pattern[j].nr;
diPatterns[nDiPatterns].pattern[j].meanDU2/=diPatterns[nDiPatterns].pattern[j].nr;
diPatterns[nDiPatterns].pattern[j].meanDD/=diPatterns[nDiPatterns].pattern[j].nr;
diPatterns[nDiPatterns].pattern[j].meanUU/=diPatterns[nDiPatterns].pattern[j].nr;
diPatterns[nDiPatterns].pattern[j].meanUD/=diPatterns[nDiPatterns].pattern[j].nr;
  }
  for(j=0;j<diPatterns[nDiPatterns].nPattern;j++)
  {
    for(i=0;i<n;i++)
    {
      if(diGraphs[i].letter1 == diPatterns[nDiPatterns].pattern[j].letter1 &&
diGraphs[i].letter2 == diPatterns[nDiPatterns].pattern[j].letter2)
      {
diPatterns[nDiPatterns].pattern[j].stdDevDUtotal+=pow(diPatterns[nDiPatterns].pat
tern[j].stdDevDUtotal+diGraphs[i].DUtotal,2);
diPatterns[nDiPatterns].pattern[j].stdDevDU1+=pow(diPatterns[nDiPatterns].patter
n[j].stdDevDU1+diGraphs[i].DU1,2);
diPatterns[nDiPatterns].pattern[j].stdDevDU2+=pow(diPatterns[nDiPatterns].patter
n[j].stdDevDU2+diGraphs[i].DU2,2);
diPatterns[nDiPatterns].pattern[j].stdDevDD+=pow(diPatterns[nDiPatterns].pattern
[j].stdDevDD+diGraphs[i].DD,2);
diPatterns[nDiPatterns].pattern[j].stdDevUU+=pow(diPatterns[nDiPatterns].pattern
[j].stdDevUU+diGraphs[i].UU,2);
diPatterns[nDiPatterns].pattern[j].stdDevUD+=pow(diPatterns[nDiPatterns].pattern
[j].stdDevUD+diGraphs[i].UD,2);
      }
    }
  }
  for(j=0;j<diPatterns[nDiPatterns].nPattern;j++)
  {

diPatterns[nDiPatterns].pattern[j].stdDevDUtotal=sqrt(diPatterns[nDiPatterns].patte
rn[j].stdDevDUtotal/diPatterns[nDiPatterns].pattern[j].nr);
diPatterns[nDiPatterns].pattern[j].stdDevDU1=sqrt(diPatterns[nDiPatterns].pattern[
j].stdDevDU1/diPatterns[nDiPatterns].pattern[j].nr);
diPatterns[nDiPatterns].pattern[j].stdDevDU2=sqrt(diPatterns[nDiPatterns].pattern[
j].stdDevDU2/diPatterns[nDiPatterns].pattern[j].nr);
diPatterns[nDiPatterns].pattern[j].stdDevDD=sqrt(diPatterns[nDiPatterns].pattern[j]
.stdDevDD/diPatterns[nDiPatterns].pattern[j].nr);
```

```
diPatterns[nDiPatterns].pattern[j].stdDevUU=sqrt(diPatterns[nDiPatterns].pattern[j]
.stdDevUU/diPatterns[nDiPatterns].pattern[j].nr);
diPatterns[nDiPatterns].pattern[j].stdDevUD=sqrt(diPatterns[nDiPatterns].pattern[j]
.stdDevUD/diPatterns[nDiPatterns].pattern[j].nr);
  }
  nDiPatterns++;
  return 1;
}

int writeMeanAndStdDevDiPattern()
{
  int i,j,user;
  FILE *d;
  d=fopen("DiPatternMeanStdDev.txt","w");
  for(user=0;user<nDiPatterns;user++)
  {
    fprintf(d,"\n%sL1\t",diPatterns[user].user);
    for(i=0;i<diPatterns[user].nPattern;i++)
        fprintf(d,"%d\t",diPatterns[user].pattern[i].letter1);
    fprintf(d,"\n%sL2\t",diPatterns[user].user);
    for(i=0;i<diPatterns[user].nPattern;i++)
        fprintf(d,"%.d\t",diPatterns[user].pattern[i].letter2);
    fprintf(d,"\n%sNr\t",diPatterns[user].user);
    for(i=0;i<diPatterns[user].nPattern;i++)
        fprintf(d,"%d\t",diPatterns[user].pattern[i].nr);
    fprintf(d,"\n%sMean\t",diPatterns[user].user);
    for(i=0;i<diPatterns[user].nPattern;i++)
        fprintf(d,"%.3f\t",diPatterns[user].pattern[i].meanDUtotal);
    fprintf(d,"\n%sStdDev\t",diPatterns[user].user);
    for(i=0;i<diPatterns[user].nPattern;i++)
    {
        fprintf(d,"%.3f\t",diPatterns[user].pattern[i].stdDevDUtotal);
    }
  }
  fclose(d);
  return 1;
}

int constructAllDiPattern()
{
  int i,j,k, user;
  for(user=0;user<nDiPatterns;user++)
  {
    for(i=0;i<diPatterns[user].nPattern;i++)
    {
      for(j=0;j<allDiPatterns.nPattern;j++)
      {
        if(diPatterns[user].pattern[i].letter1 == allDiPatterns.pattern[j].letter1 &&
diPatterns[user].pattern[i].letter2 == allDiPatterns.pattern[j].letter2)
          break;
      }
```

```
        allDiPatterns.pattern[j].letter1=diPatterns[user].pattern[i].letter1;
        allDiPatterns.pattern[j].letter2=diPatterns[user].pattern[i].letter2;
        allDiPatterns.pattern[j].nr+=diPatterns[user].pattern[i].nr;

allDiPatterns.pattern[j].meanDUtotal+=diPatterns[user].pattern[i].meanDUtotal;
        allDiPatterns.pattern[j].meanDU1+=diPatterns[user].pattern[i].meanDU1;
        allDiPatterns.pattern[j].meanDU2+=diPatterns[user].pattern[i].meanDU2;
        allDiPatterns.pattern[j].meanDD+=diPatterns[user].pattern[i].meanDD;
        allDiPatterns.pattern[j].meanUU+=diPatterns[user].pattern[i].meanUU;
        allDiPatterns.pattern[j].meanUD+=diPatterns[user].pattern[i].meanUD;

allDiPatterns.pattern[j].stdDevDUtotal+=diPatterns[user].pattern[i].stdDevDUtotal;
        allDiPatterns.pattern[j].stdDevDU1+=diPatterns[user].pattern[i].stdDevDU1;
        allDiPatterns.pattern[j].stdDevDU2+=diPatterns[user].pattern[i].stdDevDU2;
        allDiPatterns.pattern[j].stdDevDD+=diPatterns[user].pattern[i].stdDevDD;
        allDiPatterns.pattern[j].stdDevUU+=diPatterns[user].pattern[i].stdDevUU;
        allDiPatterns.pattern[j].stdDevUD+=diPatterns[user].pattern[i].stdDevUD;
        if(j==allDiPatterns.nPattern)
          allDiPatterns.nPattern++;
    }
  }
  for(j=0;j<allDiPatterns.nPattern;j++)
      {
        allDiPatterns.pattern[j].meanDUtotal/=allDiPatterns.pattern[j].nr;
        allDiPatterns.pattern[j].meanDU1/=allDiPatterns.pattern[j].nr;
        allDiPatterns.pattern[j].meanDU2/=allDiPatterns.pattern[j].nr;
        allDiPatterns.pattern[j].meanDD/=allDiPatterns.pattern[j].nr;
        allDiPatterns.pattern[j].meanUU/=allDiPatterns.pattern[j].nr;
        allDiPatterns.pattern[j].meanUD/=allDiPatterns.pattern[j].nr;
        allDiPatterns.pattern[j].stdDevDUtotal/=allDiPatterns.pattern[j].nr;
        allDiPatterns.pattern[j].stdDevDU1/=allDiPatterns.pattern[j].nr;
        allDiPatterns.pattern[j].stdDevDU2/=allDiPatterns.pattern[j].nr;
        allDiPatterns.pattern[j].stdDevDD/=allDiPatterns.pattern[j].nr;
        allDiPatterns.pattern[j].stdDevUU/=allDiPatterns.pattern[j].nr;
        allDiPatterns.pattern[j].stdDevUD/=allDiPatterns.pattern[j].nr;
      }
  return 1;
}

int sortDiPatterns()
{
  int i,j,k,sw=1,allKey=0;
  userDiPattern aux;
  sw=1;
  while(sw!=0)
  {
    sw=0;
    for(i=0;i<allDiPatterns.nPattern-1;i++)
    {
      if(allDiPatterns.pattern[i].nr < allDiPatterns.pattern[i+1].nr)
      {
```

```
        aux=allDiPatterns.pattern[i];
        allDiPatterns.pattern[i]=allDiPatterns.pattern[i+1];
        allDiPatterns.pattern[i+1]=aux;
        sw=1;
       }
     }
   }
   for(i=0;i<allDiPatterns.nPattern-1;i++)
   {
     for(j=0;j<100;j++)
     {
       if(keyAndKeyCodes[j].keyCode==allDiPatterns.pattern[i].letter1)
       break;
     }
     for(k=0;k<100;k++)
     {
       if(keyAndKeyCodes[k].keyCode==allDiPatterns.pattern[i].letter2)
       break;
     }
     allKey+=allDiPatterns.pattern[i].nr;
   }
   printf("TOTAL %d taste\n",allKey);
   return 1;
}

int firstDiPatterns(int x,int letter1,int letter2)
{
   int i;
   for(i=0;i<x;i++)
   {
     if(letter1==allDiPatterns.pattern[i].letter1 &&
letter2==allDiPatterns.pattern[i].letter2)
     {
       return 1;
     }
   }
   return 0;
}

float EuclidianDistanceDiGraph(int first)
{
   int user1,user2,i,j,nr=0;
   float max=0;
    for(user1=0;user1<nDiPatterns;user1++)
   {
     for(user2=0;user2<nDiPatterns;user2++)
     {
       diPatterns[user1].distance[user2]=0;
     }
   }
   for(user1=0;user1<nDiPatterns;user1++)
```

```
 {
  for(user2=0;user2<nDiPatterns;user2++)
  {
   nr=0;
   for(i=0;i<diPatterns[user1].nPattern;i++)
    {
     for(j=0;j<diPatterns[user2].nPattern;j++)
     {
      if(diPatterns[user1].pattern[i].letter1 ==
diPatterns[user2].pattern[j].letter1 && diPatterns[user1].pattern[i].letter2 ==
diPatterns[user2].pattern[j].letter2)
       {

if(firstDiPatterns(first,diPatterns[user1].pattern[i].letter1,diPatterns[user1].pattern[i
].letter2))
          {
           break;
          }
       }
     }
     if(j!=diPatterns[user2].nPattern)
     {

diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanDUtotal-
diPatterns[user2].pattern[j].meanDUtotal,2);

diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanDU1-
diPatterns[user2].pattern[j].meanDU1,2);

diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanDU2-
diPatterns[user2].pattern[j].meanDU2,2);

diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanDD-
diPatterns[user2].pattern[j].meanDD,2);

diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanUU-
diPatterns[user2].pattern[j].meanUU,2);

diPatterns[user1].distance[user2]+=pow(diPatterns[user1].pattern[i].meanUD-
diPatterns[user2].pattern[j].meanUD,2);
       nr+=6;
     }
    }
    diPatterns[user1].distance[user2]=sqrt(diPatterns[user1].distance[user2]);
    if(max<diPatterns[user1].distance[user2])
    {
     max=diPatterns[user1].distance[user2];
    }
  }
 }
 return max;
```

```
}

float ManhattanDistanceDiGraph(int first)
{
  int user1,user2,i,j,nr=0;
  float max=0;
   for(user1=0;user1<nDiPatterns;user1++)
  {
    for(user2=0;user2<nDiPatterns;user2++)
    {
      diPatterns[user1].distance[user2]=0;
    }
  }
  for(user1=0;user1<nDiPatterns;user1++)
  {
    for(user2=0;user2<nDiPatterns;user2++)
    {
      nr=0;
      for(i=0;i<diPatterns[user1].nPattern;i++)
        {
          for(j=0;j<diPatterns[user2].nPattern;j++)
          {
            if(diPatterns[user1].pattern[i].letter1 ==
diPatterns[user2].pattern[j].letter1 && diPatterns[user1].pattern[i].letter2 ==
diPatterns[user2].pattern[j].letter2)
           {

if(firstDiPatterns(first,diPatterns[user1].pattern[i].letter1,diPatterns[user1].pattern[i
].letter2))
             {
               break;
             }
           }
         }
         if(j!=diPatterns[user2].nPattern)
         {

diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanDUtotal-
diPatterns[user2].pattern[j].meanDUtotal);

diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanDU1-
diPatterns[user2].pattern[j].meanDU1);

diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanDU2-
diPatterns[user2].pattern[j].meanDU2);

diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanDD-
diPatterns[user2].pattern[j].meanDD);

diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanUU-
diPatterns[user2].pattern[j].meanUU);
```

```
diPatterns[user1].distance[user2]+=fabs(diPatterns[user1].pattern[i].meanUD-
diPatterns[user2].pattern[j].meanUD);
          nr+=1;
        }
       }
       if(max<diPatterns[user1].distance[user2])
       {
        max=diPatterns[user1].distance[user2];
       }
    }
  }
  return max;
}

float ADistanceDiGraph(float t, float first)
{
  int user1,user2,i,j,nr=0;
  float max=0;

   for(user1=0;user1<nDiPatterns;user1++)
  {
    for(user2=0;user2<nDiPatterns;user2++)
    {
     diPatterns[user1].distance[user2]=0;
    }
  }
  for(user1=0;user1<nDiPatterns;user1++)
  {
    for(user2=0;user2<nDiPatterns;user2++)
    {
     nr=0;
     for(i=0;i<diPatterns[user1].nPattern;i++)
       {
        for(j=0;j<diPatterns[user2].nPattern;j++)
        {
          if(diPatterns[user1].pattern[i].letter1 ==
diPatterns[user2].pattern[j].letter1 && diPatterns[user1].pattern[i].letter2 ==
diPatterns[user2].pattern[j].letter2)
          {

if(firstDiPatterns(first,diPatterns[user1].pattern[i].letter1,diPatterns[user1].pattern[i
].letter2))
            {
             nr++;
             break;
            }
          }
        }
        if(j!=diPatterns[user2].nPattern)
        {
```

```
          if(diPatterns[user1].pattern[i].meanDUtotal >
diPatterns[user2].pattern[j].meanDUtotal)
          {
            if(diPatterns[user1].pattern[i].meanDUtotal /
diPatterns[user2].pattern[j].meanDUtotal < t)
               diPatterns[user1].distance[user2]++;
          }
          else
          {
            if(diPatterns[user2].pattern[j].meanDUtotal /
diPatterns[user1].pattern[i].meanDUtotal < t)
              diPatterns[user1].distance[user2]++;
          }
        }
      }
     diPatterns[user1].distance[user2]=1-diPatterns[user1].distance[user2]/nr;

     if(max<diPatterns[user1].distance[user2])
     {
       max=diPatterns[user1].distance[user2];
     }
    }
  }
  return max;
}

int writeDistancesDiGraph()
{
  int user1, user2;
  FILE *d;
  d=fopen("DistancesDiGraphs.txt","w");
  fprintf(d,"user \t");
  for(user1=0;user1<nDiPatterns;user1++)
  {
    fprintf(d,"%s\t",diPatterns[user1].user);
  }
  for(user1=0;user1<nDiPatterns;user1++)
  {
    fprintf(d,"\n%s\t",diPatterns[user1].user);
    for(user2=0;user2<nDiPatterns;user2++)
    {
      fprintf(d,"%.2f\t",diPatterns[user1].distance[user2]);
    }
  }
  fclose(d);
  return 1;
}

int diGraphsFARandFRR(char file[],float max)
{
  FILE *d;
```

```
  d=fopen(file,"w");
  int i,j,k,user1,user2, sw=0, swZMFAR=0;
  float EER=0, FAR=0, FRR=0, TAR=0, TRR=0, TA=0, TR=0, FA=0, FR=0, coeff=1,
iEER=0, ZMFAR, iZMFAR;
  if(max<=100)
    coeff=100;
  if(max<=10)
    coeff=1000;
  for(i=0;i<((int)max+1)*(int)coeff;i++)
  {
    TA=0, TR=0, FA=0, FR=0;
    for(user1=0;user1<nDiPatterns;user1++)
    {
      for(user2=0;user2<nDiPatterns;user2++)
      {
        if(user1!=user2)
        {
          if(diPatterns[user1].user[4]==diPatterns[user2].user[4] &&
diPatterns[user1].user[5]==diPatterns[user2].user[5] &&
diPatterns[user1].user[6]==diPatterns[user2].user[6] &&
diPatterns[user1].user[7]==diPatterns[user2].user[7])
          {
            if(diPatterns[user1].distance[user2]<(float)i/coeff)
              TA++;
            else
              FR++;
          }
          else
          {
            if(diPatterns[user1].distance[user2]<(float)i/coeff)
              FA++;
            else
              TR++;
          }
        }
      }
    }
    FAR=FA/(FA+TR)*100;
    FRR=FR/(FR+TA)*100;
    TAR=TA/(TA+FR)*100;
    TRR=TR/(TR+FA)*100;
    if(FAR>FRR && sw==0)
    {
      EER=FAR;
      iEER=i;
      sw=1;
    }
    if(FAR!=0 && swZMFAR==0)
    {
      ZMFAR=(FRR+ZMFAR)/2;
      iZMFAR=(iZMFAR+i)/2;
```

```
      swZMFAR=1;
    }
    if(FAR==0)
    {
      ZMFAR=FRR;
      iZMFAR=i;
    }
    fprintf(d,"\n%.2f\t%.2f\t%.2f\t%.2f\t%.2f",(float)i/coeff,FAR,FRR,TAR,TRR);
  }
  fprintf(d,"\nEER= %.2f in %.2f",EER, iEER/coeff);
  printf("EER=\t%.2f\t%.2f\n",EER, iEER/coeff);
  printf("ZMFAR=\t%.2f\t%.2f\n", ZMFAR, iZMFAR/coeff);
  fclose(d);
  return 1;
}

int constructTriGraphs(int n, char user[])
{
  int i;
  for(i=0;i<n-2;i++) //construct triGraphs
          {
            triGraphs[i].letter1=oneGraphs[i].letter;
            triGraphs[i].letter2=oneGraphs[i+1].letter;
            triGraphs[i].letter3=oneGraphs[i+2].letter;
            strcpy(triGraphs[i].key1, oneGraphs[i].key);
            strcpy(triGraphs[i].key2, oneGraphs[i+1].key);
            strcpy(triGraphs[i].key3, oneGraphs[i+2].key);
            triGraphs[i].DU1=oneGraphs[i].DU;
            triGraphs[i].DU2=oneGraphs[i+1].DU;
            triGraphs[i].DU3=oneGraphs[i+2].DU;
            triGraphs[i].UD1=oneGraphs[i+1].UDprev;
            triGraphs[i].UD2=oneGraphs[i+2].UDprev;
            triGraphs[i].D1D2=triGraphs[i].UD1+triGraphs[i].DU1;
            triGraphs[i].D1D3=triGraphs[i].D1D2+triGraphs[i].UD2+triGraphs[i].DU2;
            triGraphs[i].D2D3=triGraphs[i].UD2+triGraphs[i].DU2;
            triGraphs[i].U1U2=triGraphs[i].UD1+triGraphs[i].DU2;
            triGraphs[i].U1U3=triGraphs[i].U1U2+triGraphs[i].UD2+triGraphs[i].DU3;
            triGraphs[i].U2U3=triGraphs[i].UD2+triGraphs[i].DU3;
            triGraphs[i].D1U2=triGraphs[i].D1D2+triGraphs[i].DU2;
            triGraphs[i].D2U3=triGraphs[i].D2D3+triGraphs[i].DU3;
            triGraphs[i].U1D3=triGraphs[i].U1U2+triGraphs[i].UD2;
          triGraphs[i].DUtotal=triGraphs[i].DU1+triGraphs[i].UD1+triGraphs[i].DU2+t
          riGraphs[i].UD2+triGraphs[i].DU3;
            constructWord(triGraphs[i].word,i,3);
fprintf(k,"%s\t%s\t%s\t%d\t%d\t%d\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t
%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%s\n",triGraphs[i].key1,triGrap
hs[i].key2,triGraphs[i].key3,triGraphs[i].letter1,triGraphs[i].letter2,triGraphs[i].lette
r3,triGraphs[i].DU1,triGraphs[i].DU2,triGraphs[i].DU3,triGraphs[i].UD1,triGraphs[i].
UD2,triGraphs[i].D1D2,triGraphs[i].D1D3,triGraphs[i].D2D3,triGraphs[i].U1U2,triGr
aphs[i].U1U3,triGraphs[i].U2U3,triGraphs[i].D1U2,triGraphs[i].D2U3,triGraphs[i].U1
D3,triGraphs[i].DUtotal,triGraphs[i].word);
```

```
        }
  return 1;
}
int constructFourGraphs(int n, char user[])
{
  int i;
  for(i=0;i<n-3;i++) //construct 4Graphs
        {
          fourGraphs[i].letter1=oneGraphs[i].letter;
          fourGraphs[i].letter2=oneGraphs[i+1].letter;
          fourGraphs[i].letter3=oneGraphs[i+2].letter;
          fourGraphs[i].letter4=oneGraphs[i+3].letter;
          strcpy(fourGraphs[i].key1, oneGraphs[i].key);
          strcpy(fourGraphs[i].key2, oneGraphs[i+1].key);
          strcpy(fourGraphs[i].key3, oneGraphs[i+2].key);
          strcpy(fourGraphs[i].key4, oneGraphs[i+3].key);
          fourGraphs[i].DU1=oneGraphs[i].DU;
          fourGraphs[i].DU2=oneGraphs[i+1].DU;
          fourGraphs[i].DU3=oneGraphs[i+2].DU;
          fourGraphs[i].DU4=oneGraphs[i+3].DU;
          fourGraphs[i].UD1=oneGraphs[i+1].UDprev;
          fourGraphs[i].UD2=oneGraphs[i+2].UDprev;
          fourGraphs[i].UD3=oneGraphs[i+3].UDprev;
fourGraphs[i].DUtotal=fourGraphs[i].DU1+fourGraphs[i].UD1+fourGraphs[i].DU2+f
ourGraphs[i].UD2+fourGraphs[i].DU3+fourGraphs[i].UD3+fourGraphs[i].DU4;
          constructWord(fourGraphs[i].word,i,4);
fprintf(l,"%s\t%s\t%s\t%s\t%d\t%d\t%d\t%d\t%.2f\t%.2f\t%.2f\t%.2f\t%.2f\t%.
2f\t%.2f\t%.2f\t%s\n",fourGraphs[i].key1,fourGraphs[i].key2,fourGraphs[i].key3,fo
urGraphs[i].key4,fourGraphs[i].letter1,fourGraphs[i].letter2,fourGraphs[i].letter3,fo
urGraphs[i].letter4,fourGraphs[i].DU1,fourGraphs[i].DU2,fourGraphs[i].DU3,fourGra
phs[i].DU4,fourGraphs[i].UD1,fourGraphs[i].UD2,fourGraphs[i].UD3,fourGraphs[i].D
UtotaI,fourGraphs[i].word);
        }
  return 1;
}

int closeFiles()
{
  fprintf(g,"-1\n");
  fprintf(h,"-1\n");
  fprintf(k,"-1\n");
  fprintf(l,"-1\n");
  fclose(f);
  fclose(g);
  fclose(h);
  fclose(k);
  fclose(l);
  fclose(o);
  fclose(p);
  fclose(q);
  return 1;
```

```
}

int main(void)
{
  int i,n,nAll=0;
  int allKey=0;
  char user[20],fileUser[20];
  openFiles();
  do{
        fscanf(f,"%s",user);
        if(strcmp(user,"-1")==0)
            break;
        if(user[0]=='u' && user[1]=='s' && user[2]=='e' && user[3]=='r')
        {
          strcpy(fileUser,"UsersTexts/");
          strcat(fileUser,user);
          strcat(fileUser,".txt");
          m=fopen(fileUser,"w");
          fprintf(g,"%s\n",user);
          fprintf(h,"%s\n",user);
          fprintf(k,"%s\n",user);
          fprintf(l,"%s\n",user);
          fprintf(o,"%s\n",user);
          n=constructOneGraphs(n, user);
          for(i=0;i<n;i++)
           {
           fprintf(g,"%s\t%d\t%d\t%d\t%.2f\t%.2f\t%.2f\t%d\t%s\t%d\t%s\n",oneG
           raphs[i].key, oneGraphs[i].letter, oneGraphs[i].D,
           oneGraphs[i].U,oneGraphs[i].DU,oneGraphs[i].UDprev,oneGraphs[i].UDnext
           ,oneGraphs[i].letterPrev,oneGraphs[i].keyPrev,oneGraphs[i].letterNext,one
           Graphs[i].keyNext);
           }
          for(i=0;i<n;i++)
          {
           allUsersOneGraphs[nAll++]=oneGraphs[i];
          }
          constructDiGraphs(n, user);
          allKey=allKey+n-1;
          constructDiPattern(user,n);
          constructTriGraphs(n, user);
          constructFourGraphs(n, user);
          meanAndStdDev(n);
          keyDistribution(n,user);
          fclose(m);
          fprintf(g,"-1\n");
          fprintf(h,"-1\n");
          fprintf(k,"-1\n");
          fprintf(l,"-1\n");
        }
   }while(strcmp(user,"-1")!=0);
```

```
for(i=0;i<nAll;i++)
  {
    oneGraphs[i]=allUsersOneGraphs[i];
  }
  n=nAll;
  constructDiGraphs(n, "All");
  constructTriGraphs(n, "All");
  constructFourGraphs(n, "All");
  meanAndStdDev(n);
  keyDistribution(n,user);
  float max;
  max=calculateEuclidianDistances();
  writeDistances();
  calculateFARandFRR(max,"FARandFRR_EuclidianDist.txt");
  max=calculateManhattanDistances(0.31);
  printf("%.2f\n",0.31);
  writeDistances();
  calculateFARandFRR(max,"FARandFRR_manhattanDist.txt");
  printf("Manhattan Succes\n");
  max=calculateRdistances();
  writeDistances();
  RcalculateFARandFRR(max,"FARandFRR_R_Dist.txt");
  max=calculateADistances(1.25);
  writeDistances();
  calculateFARandFRR(max,"FARandFRR_A1.25_Dist.txt");
  max=calculateADistances(1.13);
  writeDistances();
  constructAllDiPattern();
  sortDiPatterns();
  writeMeanAndStdDevDiPattern();
  i=10;
  max=EuclidianDistanceDiGraph(i);
  printf("%d\n",i);
  diGraphsFARandFRR("diGraphsFARandFRR_Euclid.txt",max);
  printf("Euclid diGraphs SUCCES\n");
  i=12;
  max=ManhattanDistanceDiGraph(i);
  writeDistancesDiGraph();
  printf("%d\n",i);
  diGraphsFARandFRR("diGraphsFARandFRR_Manhattan.txt",max);
  printf("Manhattan diGraphs SUCCES\n");
  max=ADistanceDiGraph(1.25,i);
  diGraphsFARandFRR("diGraphsFARandFRR_A1.25.txt",max);
  closeFiles();
  return 0;
}
```

SCIENTIFIC ACTIVITY

Web of science papers:

A.C. Iapa, „Outstanding research in mooc and future development”, in *10th International Scientific Conference on eLearning and Software for Education*, Bucharest, ROMANIA, APR 2014, vol. 1, pp. 251-254. (WOS:000357153000035)

A.C. Iapa, „MOOC and Learning Analytics: interaction and evolution”, in *International Conference on Social Media in Academia - Research and Teaching (SMART)*, Timisoara, ROMANIA, SEP 2014, pp. 55-59. (WOS:000367888300009)

M. Ivanova, C. Holotescu, G.Grosseck, **C. Iapa**, „Relations between learning analytics and data privacy in MOOCs”, in *12th International Scientific Conference on eLearning and Software for Education (eLSE)*, APR 2016, vol 3, pp. 13-20. (WOS:000385397100001)

Iapa A.C., Cretu V.I., „Modified Distance Metric That Generates Better Performance For The Authentication Algorithm Based On Free-Text Keystroke Dynamics”, *IEEE 15th International Symposium on Applied Computational Intelligence and Informatics*, Timisoara, Romania, 2021 – *paper sent, unpublished*

Iapa A.C., Cretu V.I., „Evaluating the performance of authentication algorithms based on keystroke dynamics used in online educational platforms”, *The 17th International Scientific Conference eLearning and Software for Education*, Bucharest, Romania, 2021 – *accepted paper, unpublished*

Other papers:

M. Popa and **C. Iapa**, „Embedded weather station with remote wireless control”, *2011 19th Telecommunications Forum (TELFOR)* Proceedings of Papers, Belgrade, 2011, pp. 297-300. (BDI: Scopus, IEEE Xplore)

DORIN LELEA

Digitally signed by DORIN LELEA
T=Director Scoala Doctorala, C=RO, S=TIMIS, L=Timisoara, O=UNIVERSITATEA POLITEHNICA DIN TIMISOARA, E=dorin.lelea@upt.ro, SERIALNUMBER=201104209DL10, G=DORIN, SN=LELEA, CN=DORIN LELEA
Atest integritatea acestui document
Computerul meu
2021.04.22 11:28

BUPT