UNIVERSITY OF READING

Free-Text Keystroke Dynamics Authentication with a Reduced Need for Training and Language Independency

Submitted by:

Arwa Alsultan

Submitted for the Degree of Doctor of Philosophy

Supervisor(s):

Prof. Kevin Warwick & Dr. Hong Wei

Department of Computer Science

School of Mathematical, Physical and Computational Sciences

September 2016

To my caring parents - Aljoharah & Fahad -

who encouraged me during my educational journey since I was very young. I would not have been able to achieve any of this without their love and prayers.

To my loving husband - Rawad -

who stood by my side during this journey. He has never stopped believing in me and his love and support is the reason for all my success.

Declaration

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Arwa Alsultan

Acknowledgment

I would like to express my deep appreciation to my supervisor Prof. Kevin Warwick for all the support he provided me with during my study. Without his guidance and continuous help, this research would not have been possible.

I would also like to convey my sincerer gratitude to my supervisor Dr. Hong Wei for her guidance and encouragement throughout my research. She never stopped reassuring me and reinforcing me to achieve my goals. She was there when I needed her the most.

I would like to thank Dr. Tristan Fletcher for granting me permission to use one of the figures produced by him in this thesis.

I also would like to express my appreciation to Dr. Luis Gonzalo Sánchez Giraldo for his cooperation with the progressive MANOVA algorithm. He was very helpful and provided me with all the information I needed to use this algorithm.

I wish to extend my thanks to the participants who were involved in this research for the time they took out of their busy schedules to contribute in this study.

Finally, I would like to deeply thank my brothers & sisters, family members and friends for their continuous love and support.

Abstract

This research aims to overcome the drawback of the large amount of training data required for free-text keystroke dynamics authentication. A new key-pairing method, which is based on the keyboard's key-layout, has been suggested to achieve that. The method extracts several timing features from specific key-pairs. The level of similarity between a user's profile data and his or her test data is then used to decide whether the test data was provided by the genuine user. The key-pairing technique was developed to use the smallest amount of training data in the best way possible which reduces the requirement for typing long text in the training stage. In addition, non-conventional features were also defined and extracted from the input stream typed by the user in order to understand more of the users typing behaviours. This helps the system to assemble a better idea about the user's identity from the smallest amount of training data. Non-conventional features compute the average of users performing certain actions when typing a whole piece of text. Results were obtained from the tests conducted on each of the key-pair timing features and the non-conventional features, separately. An FAR of 0.013, 0.0104 and an FRR of 0.384, 0.25 were produced by the timing features and non-conventional features, respectively. Moreover, the fusion of these two feature sets was utilized to enhance the error rates. The feature-level fusion thrived to reduce the error rates to an FAR of 0.00896 and an FRR of 0.215 whilst decision-level fusion succeeded in achieving zero FAR and FRR. In addition, keystroke dynamics research suffers from the fact that almost all text included in the studies is typed in English. Nevertheless, the key-pairing method has the advantage of being language-independent. This allows for it to be applied on text typed in other languages. In this research, the key-pairing method was applied to text in Arabic. The results produced from the test conducted on Arabic text were similar to those produced from English text. This proves the applicability of the key-pairing method on a language other than English even if that language has a completely different alphabet and characteristics. Moreover, experimenting with texts in English and Arabic produced results showing a direct relation between the users' familiarity with the language and the performance of the authentication system.

Glossary

ACO	Ant colony optimisation
ANOVA	Analysis of variance
ATM	Automatic teller machines
DD	Down-down time for a key-pair
Di-graph	Combination of two keys
DSS	Decision support systems
DTs	Decision trees
EER	Equal error rate
FAR	False accept rate
FRR	False reject rate
GA	Genetic algorithm
GUI	Graphical user interface
H_1	Hold time for the first key
H_2	Hold time for the second key
LSA	Percentage of left shifts released after letter
LSB	Percentage of left shifts released before letter
MANOVA	Multivariate analysis of variance
ms	Millisecond
NegUD	Percentage of negative up-down
NegUU	Percentage of negative up-up
N-graph	Combination of n keys
NN	Neural networks
PCA	Principal component analysis

PDA	Personal digital assistants
PSO	Particle swarm optimization
RBF	Radial basis kernel
ROC	Receiver operating characteristic
RSA	Percentage of right shifts released after letter
RSB	Percentage of right shifts released before
SVMs	Support vector machine
TAR	True accept rate
Tri-graph	Combination of three keys
UD	Up-down time for a key-pair
UU	Up-up time for a key-pair
WPM	Word-per-minute

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Over the past few decades, online services have become an essential tool for performing many tasks on a daily basis. Such services usually require a username and password in order to verify users' identities. Unfortunately, the security of passwords is at risk because of social engineering, and they can be easy to crack using various methods, such as the dictionary attack and even a brute force attack [1]. Therefore, users are obliged to take extreme measures to safeguard their passwords, through procedures that include remembering long and complex passwords in addition to being required to change their passwords periodically. This causes users to become frustrated and apprehensive, especially when a single user is responsible for more than just a handful of ID/password pairs spread over multiple systems [2].

Therefore, it has become necessary to find a more user-friendly method for authentication. One alternative to ID/passwords is behavioural biometrics, such as signature recognition, handwriting recognition, gait analysis etc., all of which rely on the user's behavioural patterns, making the process intuitive for the user and difficult for others to imitate [3]. Unfortunately, all of these methods need highly reliable external hardware in order to take the measurements [1]. This is considered to be a critical drawback by the users, who are concerned mainly with the practicality of the system. The cost of these devices also makes them less desirable [1].

Monitoring keystroke dynamics, on the other hand, is considered to be an effortless behaviour-based method for authenticating users by employing their typing patterns to validate their identities [4]. A standard keyboard is all that is needed for measuring this biometric; no additional hardware is involved. More specifically, the use of free-text keystroke systems can be applied in many different settings to assist in real-life situations

because it uses arbitrary text, therefore provides a balance between security and usability [5]. In addition, free-text keystroke dynamics can enhance security through continuous and non-intrusive authentication.

An issue that free-test keystroke dynamics suffers from is that it incorporates extensive amounts of input from the user in order to learn the user's keystroke habits. Whilst such experimentation can be tolerable in laboratory settings, this substantial input can in real-life situations be very irritating for the user. Therefore, this research focuses on developing techniques and capturing features that make use of the least amount of input to extrapolate the user's identity which will allow for more practicality in real-world situation.

This research considers a new method, based on the keyboard's key-layout, to compare the timing features of free-text typing samples. The method classifies the text into different key-pairs, depending on the position of the two keys on the keyboard. Timing features, such as the hold, down-down, up-up and up-down times, are extracted from the key-pairs and stored in the timing features' vector. The timing features' vectors are used to find the level of similarity among the training and testing samples.

The study follows this structured method for extracting features, in order to increase the number of the key-pairs that can be found and compared in both the training and the testing samples. This will then enhance the stability of their means, which are the main components of the user's timing vector. This will aid the key-pairing scheme to reduce the training input to the smallest possible amount. The main goal for that is to provide comfort and ease of use and, as a result, achieve a system that is both effective and practical. It is not enough to relieve users from the need to remember long passwords if they still have to type huge amounts of text when they are logging into the system. Therefore, the key-pairing scheme makes enrolment a simple, relatively rapid process.

More features are investigated in order to pursue the same aim of reducing training input. This is done to increase the amount of information extracted from a user's input and therefore build a better understanding about his or her typing behaviour. Therefore, a number of non-conventional typing features were examined. These features can be extracted collectively during the typing of a piece of text in which the user's typing habits are inspected. Features such as words-per-minute, error rate, and shift key usage are employed to find typing patterns that can be used to differentiate between individuals.

Fusion between the key-pairing features and non-conventional features is also conducted. Both decision-level and feature-level fusion are applied to achieve the best system performance. This allows the system to make use of both types of features to combine the timing characteristics and other input and editing patterns in the process of inferring the identity of the user.

Moreover, the key-pairing method is developed to be language-independent as it can be applied to any language. Therefore, an additional test was done in order to investigate the method's applicability in scenarios where the training samples are in a language different from English. Arabic was chosen due to its distinctive attributes that make it utterly different than English.

1.2 Overview of User Authentication

The number of information systems that people are using on a daily basis to perform regular but highly sensitive actions, such as logging-into bank accounts, online shopping, or even sending an e-mail is rapidly increasing. As simple as that may seem, such activities have to be carried-out in a completely reliable and secure manner. This emphasises the necessity of adapting a dependable authentication technique which is used as an assurance measure of the user's identity with the intention of protecting the system from fraud and impersonation. Nevertheless, the fact that these activities are performed regularly enforces the need for it to be also useable and practical.

Authentication, also called identity verification, is often confused with identification. They are, in fact, completely different concepts [6]. Identification is only concerned with determining who the user is; it does not acquire any extra information to verify the claimed identity [7]. An example of the most common technique for identification is using the login-ID in computer systems, which is normally employed by making sure that this particular ID actually exists in the database. On the other hand, authentication is focused on testing the identity of a particular user to make sure that he or she is not impersonating another person's identity and by doing so, getting access to that person's private resources [1]. This is done by asking the user to provide a special kind of information, or item, that is only known, or possessed, by him or her. The most widely used method for authentication is passwords. That is normally done by comparing the provided password with the one associated with that user in the database. It is worth noting here that identification is often not enough in the majority

of computer systems; some sort of authentication has to be implemented, especially in highly secured systems, such as on-line banking [1]. This is similar to the process in which a bank employee doesn't rely on the customer's name when conducting bank operations, but asks for actual proof such as his or her driving licence or passport before handing any important information or carrying out any operation.

Another scheme that is similar to user authentication and identification is user classification [8]. In user authentication, the system must determine if the sample is produced by the user whose identity has been provided to the system, whilst in user identification, the system must decide if the sample comes from any one of the users known to the system or someone un-known to the system. In user classification, on the other hand, the system must find the identity of the user supplying the sample provided that the sample comes from one of the users known to the system [8].

In this research, user classification is applied yet the term user authentication is used. The decision to use user authentication was to reduce the confusion on the readers' behalf. As most research in the area of information security is concerned with merely authentication and identification [9], it was seen to be more appropriate to use the term user authentication. Moreover, the study implemented here is considered closer to authentication than to identification. This is because some information is known beforehand about the user; that he or she is from the list of users known to the system. Furthermore, as the user classification is harder [8] and its results are conservative compared with authentication, the use of the term, user authentication will produce results that are considered understated. This is because user classification is a 1-to-M process whilst user authentication is a 1-to-1 process [9]. Moreover, in the classification process, it is possible to achieve authentication if needed. Therefore, from the readers' point of view, using the term user authentication is more apt for this study.

There are different critical issues that have to be accounted for when considering the application of any authentication system [10]. These issues include: accuracy, speed, resistance to counterfeiting, reliability, data storage requirements, enrolment time and user acceptance. Not all systems provide satisfactory levels of each of these issues, therefore selecting the best one for the application in-hand is crucial.

A large amount of research has been done to develop various authentication methods over more than half a century. The main classes into which user authentication is categorised are three [11]: knowledge-based, possession-based, and biometric-based. In addition, other less-

dominant methods are in-use in some applications. Moreover, it is quite common to use two or more of these classes to provide safer, multi-factor authentication. The next sections explain, in some detail, each of these authentication classes and how they are used.

1.2.1 Knowledge-based Authentication

This authentication method uses a piece of information which is only known by the user [1]. The simplest information that a user might provide to confirm his identity is personal and historical information such as date of birth, address and national insurance number [12]. Although, this is still used in some circumstances, it does not provide adequate proof of the person's identity since some of this information can be known by his or her friends or family members.

The most used knowledge-based authentication method in electronic systems is the ID/password system. Although, it is the most common method for authentication, it is considered the weakest [13]. Passwords can be guessed easily by using the user's personal details, as it is common for users to use their date of birth or children names as their passwords. Social engineering is a way of tricking a user to reveal his or her password through deceptive phone calls or phishing e-mails[1]. Passwords are also easy to crack using various methods, such as the dictionary attack and brute force attack, where an automated tool tries to discover the user's password by trying every word in the dictionary or all possible combinations of characters, respectively [1]. Spyware and adware can also be used to capture the user's password using programs such as key-loggers that operate transparently to the user to capture all his keyboard strokes and pass it to a harm-intending individual [14].

All these are reasons for the user to use great measures to protect his or her password. Using passwords more than 8 characters long and containing upper case letters, lower case letters, numbers and special characters will significantly reduce the chance of an attacker succeeding in cracking the password [1]. Nevertheless, memorizing such hard passwords causes aggravation and frustration for users, especially when a single user is almost certainly responsible for more than one pair of ID/password, extended over various systems ranging from online-bank accounts to e-mail and social media accounts. In fact, the study conducted in [2] provided evidence that a user owns on average around 25 separate electronic accounts, each of which requires a password. Moreover, a user is forced to use 8 passwords on average every day. Consequently, passwords suffer from insufficiencies caused by the problematic

compromise between security and the ability to remember them i.e. “security-usability dilemma” or “password problem” [15].

1.2.2 Possession-based Authentication

This authentication method is used to verify the user’s identity using an object of which only that user has possession [1]. It is also sometimes referred to as the ownership-based authentication. It is the oldest type of authentication; as early as using door keys to have access into rooms or buildings and using a driving licence to prove a person’s identity.

Tokens are the most well-known objects used for possession-based authentication. Tokens can be one of three types [16]: proximity cards, smartcards, and password generating tokens. First of all, proximity cards are used primarily for accessing doors and work by releasing radio signals which are received by the door’s sensor. This kind of token is frequently left at home which causes the users to ask their managers or co-workers to allow them access, which, of course, defeats the purpose of the proximity cards. Second, smartcards are plastic cards embedded with a small memory chip which is used for accessing machines such as laptops that are equipped with smartcard readers. The problem that most smartcard users face is that they often keep the smart cards in the same place that they use to store their laptops, therefore, whenever the laptop is stolen, the smartcard would also be stolen. This means that whoever got them both can have access to the laptop’s content. Lastly, password generating tokens are small gadgets that generate a random sequence of numbers every minute to be used by the user as a password to log-in to the protected system. This kind of token is unfortunately expensive to administer and manage [17].

All three kinds of tokens are at risk of being lost because of their small size. In addition, although the price of one token is fairly cheap, supplying all users in an organization with tokens and token readers is cumulatively costly [16].

1.2.3 Biometric-based Authentication

This authentication method uses something inherent only to the user as a means of identifying him or her [1]. Biometrics is categorised into three types: physiological, behavioural and cognitive traits. First, physiological biometrics is related to the user’s body, and thus, it varies from person to person [18]. Examples of such biometrics are: fingerprints, face recognition, hand geometry, and iris recognition. Although physiological biometrics are considered one of

the most secure authentication methods, it still can deny legitimate users from accessing the system when a change happens in their appearance like injuring their fingers [14].

Secondly, behavioural biometrics depends on the user's behavioural patterns, which make it instinctive for the user and hard to mimic by others [3]. These include: keystroke dynamics, signature recognition, handwriting recognition, hand gesture recognition, and gait analysis. Unfortunately most of the previous mentioned methods, except keystroke systems, need additional hardware to collect data, which is an un-desired extra cost [19].

Lastly, Cognitive biometrics is related to perception, thought process, and understanding of the user. Cognitive biometrics is based on the user's experiences; therefore, it is considered to be much easier for the user to remember [3]. Examples of this kind include question-based passwords, click-based graphical passwords and Passface graphical passwords. Question-based passwords [20] use either fact based or opinion based questions to test whether the user is providing the same answers at the enrolment and the log-in phases. Click-based graphical passwords [21] allow users to enter their passwords using mouse clicks on specific pixels of an image. Passface graphical passwords [22] oblige the user to identify specific photographs of faces which were presented to him or her during the enrolment phase. According to the study conducted in [3], cognitive biometrics cannot work as a standalone method for identity verification, as it is easy for a skilled attacker to imitate.

1.2.4 Other Authentication Methods

There are some schemes which are considered less secure than the three main methods listed above, yet they are still used as a part of the authentication system in appropriate applications. These methods were included in fewer studies compared to the three major types. Two of these less-dominant methods will be discussed here. They are: mutual acquaintance and location.

Firstly, the mutual acquaintance method [23] works by incorporating human relationships in the authentication process. A peer-level authentication is undertaken, in which a user seeks help from a third party, i.e. someone known and trusted by both parties, to aid in the process of validating his or her identity to a second user. Yet, it only makes sense to use such techniques in a social network environment where users actually know each other.

Secondly, the location of the user can be used for assessing his or her identity [24] especially in ubiquitous computing environments. In other words, if the location of the user corresponds to a place where he or she frequently visits, there is more chance of him being genuine as opposed to another place where he or she has never been. Moreover, if a user just logged-in from a certain location and, then, he or she logs-in again from a different continent within a couple of hours, there is a high chance that those two system accesses weren't performed by the same person.

Apparently, these two methods are normally not enough to be used on their own as an authentication scheme for highly secured systems [25]. Instead, they are normally used in combination with other methods in a multi-factor authentication scheme, which will be discussed next.

1.2.5 Multi-factor Authentication

Using merely one authentication method may not yield the preferable results. Therefore, a simple solution to enhance the security level of the system is to combine two or more methods to work as a multi-factor mechanism, also referred to as multi-modal authentication, for authenticating users. This will aid in supporting the weaknesses of using a single method, which will result in collectively strengthening the overall authentication system. It is often used in highly secured systems, such as the use of the PIN number and the ATM card to access one's bank account [26]. Another example is found in the study conducted to determine the probability that the user is legitimate or not using short-text keystroke dynamics and his or her location [27].

Similarly, any two or more methods can be used to achieve a multi-factor authentication system. Most of these systems utilize authentication methods from the different authentication classes, listed above, in order to create an effective system. Figure 1.1 shows the relation between the three main authentication classes and some examples of how to combine them to create practical multi-factor authentication systems.

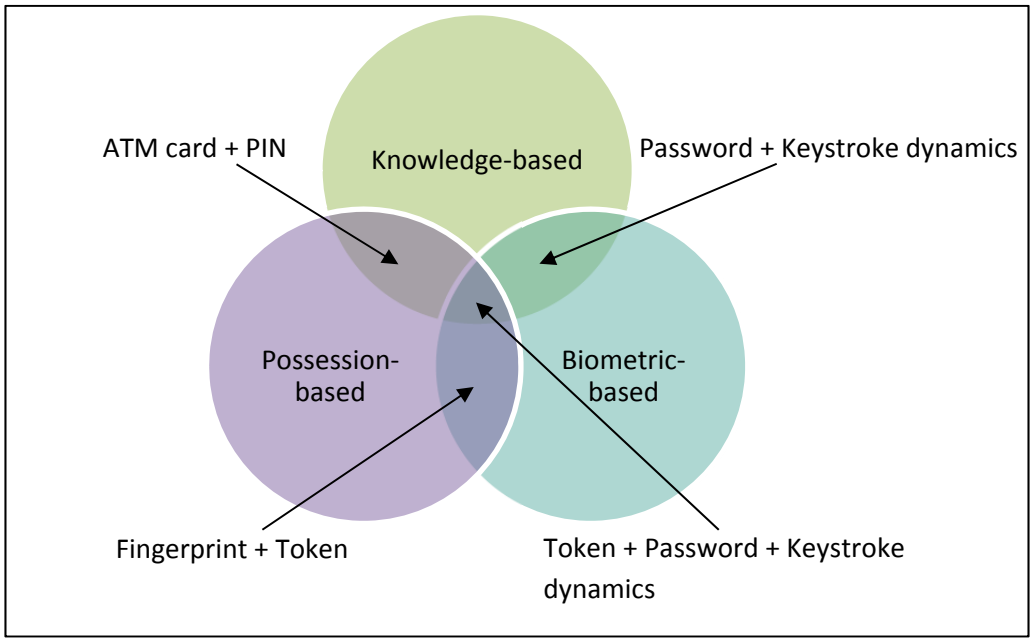

Figure 1.1: Multi-factor authentication system.

1.3 Problem Statement

The mechanism that is most commonly used for authentication in current computer systems is the ID/password method, even though it causes a number of problems with the security/usability aspect of these systems, as described in Section 1.2. There is an obvious trade-off between the security and the memorability of a password. This is known as "the password problem" and can be described briefly as follows: the password becomes harder for the user to remember as its strength against guessing and other malicious attacks increases [28]. The strengthening process includes using a greater number of characters that vary in type between letters, digits and special characters. The process also includes avoiding both well-known words and actual facts about the user. This problem is one of the main reasons why users become irritated with such schemes.

The password problem provided computer researchers with a new challenge of balancing the password's strength and its memorability level through the use of a number of methods, such as password phrases and automated password managers. In spite of the continual attempts to improve passwords, no technique has been found to be an adequate solution for the password problem. Using passphrases does not eliminate the need to memorise a phrase and might even make memorising tasks more difficult for users than single passwords. Also, automated password managers still require the use of a strong password to act as the key password in order to gain access to the password repository [16].

The next logical step, then, was to find an alternative authentication method that could provide the same level of authentication as the password method but without the drawback of the memorability issue and other vulnerabilities. Several methods were tested, including the use of a number of biometric traits that varied from the obvious fingerprint [29] and face recognition [30] to measuring brainwaves [31] and heartbeat sequences [32]. However, all of these techniques suffer from high deployment costs, including those for the additional hardware that must be used, as well as for maintenance and sustenance [1].

Nevertheless, the deployment of the user's keystroke patterns as a method for identity verification provides a less expensive mechanism and, more particularly, the use of free-text keystrokes is considered to be very easy for users. Using free-text keystrokes does not require memorising any text because the text used for enrolment does not have to be the same as the text used for log-in, which fits directly with the objectives of this study.

Over the last four decades, a large amount of research has focused on enhancing the performance of the free-text keystroke dynamics approach. Unfortunately, a crucial drawback, and one from which it still suffers, is the large amount of input that is necessary for the enrolment phase. As a result, these systems are hampered by their reduced usability because enrolment is overly time consuming. Although most researchers overlook training time, it is an important factor in real-life applications.

Therefore, this research aims to explore methods for comparing typing patterns using the smallest amount of training possible. Using only little training data reduces the amount of time the user spends enrolling, which enhances the degree of user-friendliness while maintaining the desired level of security.

Moreover, the vast majority of the research done in keystroke dynamics was conducted on text typed in English language. However, the approach developed in this research is designed to be language-independent. This means that it can be applied to different scenarios and different input languages which should appeal to more users. This is tested using text typed in Arabic language, which has very different characteristics compared to that of the English language.

1.4 Aims and Objectives

This research seeks to achieve the following main goal:

> To investigate and ideally produce a reliable free-text keystroke authentication mechanism that solves the problem of the need for massive amount of training data and can be applied to different languages.

In order to achieve this goal, the following specific objectives have been identified:

- To research the available literature for the best keystroke features applicable for use in this research.
- To study the literature to find the feature subset selection methods most applicable to the system in hand.
- To synthesise the related work to find the best classification methods for application in this problem.
- To explore and design a keyboard-based key-pairing scheme for capturing users' typing behaviours.
- To implement a non-conventional features scheme for understanding the user's typing patterns.
- To investigate the fusion methods found in literature in order to realize how it can be applied to enhance the system performance.
- To evaluate how well the suggested methods (on their own and fused together) benefit from the small amount of training data.
- To test the feasibility of applying the key-pairing scheme on text in the Arabic language.

1.5 Thesis Structure

The rest of this thesis proceeds as follows (a demonstration of the thesis structure is illustrated in Figure 1.2):

Chapter two: presents the keystroke dynamics theory, which includes a description of the biometric characteristics of keystroke dynamics, its phases, its advantages and limitations and its classes, i.e. fixed-text and free-text systems. The techniques that are followed for feature extraction and performance measurement are then considered, followed by a discussion of

state-of-the-art research that has been done in both fixed-text and free-text keystroke systems. This section concludes by outlining the factors affecting the performance of these systems, the various applications that benefit from these systems and the level of protection that these systems provide against some of the common security threats.

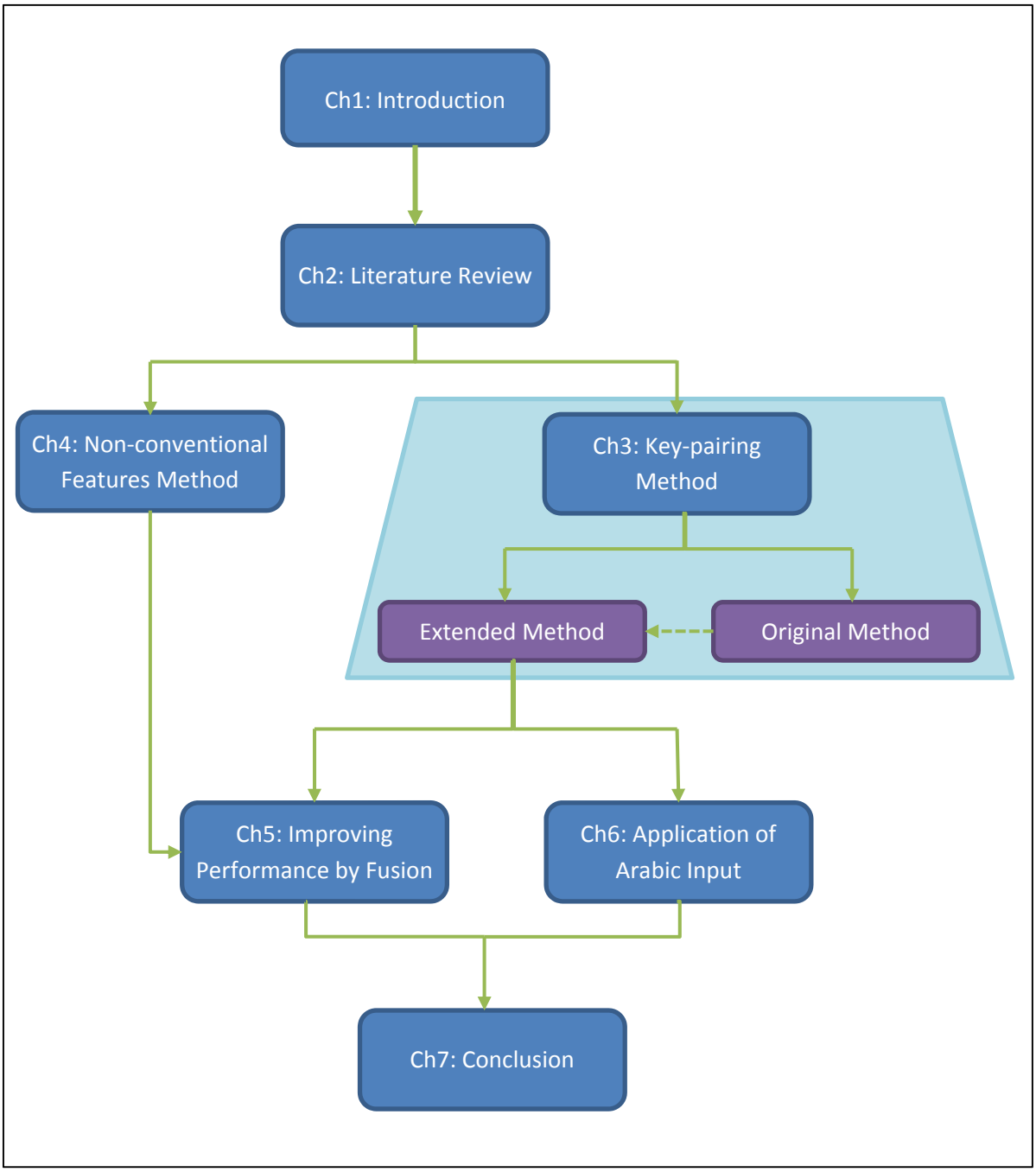

Figure 1.2: Thesis structure.

Chapter three: describes the key-pairing method. The original method, which follows a simple key-pairing mechanism and classification method, is described first. An explanation of the original key-pairing technique is presented along with the timing features that were included in the study. An indication of the data collection technique is then provided, followed by the data space. The distance calculation process is explained next. The results of the experiment and comparisons with other recent studies in the same area are presented.

After that, the extended version of the key-pairing method is presented. The modifications that have been added to the original method in order to improve performance are addressed briefly. And then, an explanation of the followed key-pairing technique is presented along with how it has been applied in the experimentation process. The timing features that were included in the study are discussed too. The feature subset selection and classification method is discussed, respectively. A description of the data collection technique and the data space is provided. The experiment results, in addition to comparisons with the original method and other studies in the area of keystroke dynamics, are also provided.

Chapter four: introduces the non-conventional features method. A description of the features and how they are extracted is explained together with the classification method used. After that, the data collection and data space are described. Experiment details are then presented. Results are presented and comparisons with the key-pairing method are then conducted.

Chapter five: provides an overview of fusion techniques and applies it to the methods described in Section 3 and Section 4. Both feature-level and decision-level fusion are applied. Results from both fusion techniques are produced in addition to a discussion of the level of improvements each has on the previously mentioned methods.

Chapter six: examines the applicability of the key-pairing method on Arabic text. In this chapter, the difference between English and Arabic is presented in addition to the way key-pairing method is applied in the case of Arabic text. The data collection and data space are also touched on in this section. Results from the Arabic study, along with comparisons with the experiment conducted on English input, are presented.

Chapter seven: offers a conclusion for the topic and outlines the contributions that have been made by this research. Any limitations faced during this research are also presented. It also lists the future work that follows the research done in this thesis.

Chapter 2

Overview of Keystroke Dynamics

2.1 Introduction

Keystroke dynamics utilize one's innate typing patterns for validating his or her identity which is deemed to be a natural behavioural-based method for authenticating users. As mentioned in [33], keystroke dynamics is "not what you type, but how you type." In this approach, the user types in text, as usual, without any kind of extra work to be done for authentication. Moreover, it only involves the user's own keyboard and no other external hardware.

The original idea of using keystroke patterns for user identification/authentication purposes originated from the idea of identifying the sender of Morse code on a telegraph machine around 1895, where operators had been able to identify the sender of a message by the rhythm, pace and syncopation of the received taps [19]. Likewise, the uniqueness of a user's typing pattern is noticeable as anyone in close proximity to a certain typist is often able to recognise the typist through his or her typing rhythm [19]. Therefore, keystroke patterns provide promising identity verification abilities [34].

Checking the handwritten signature is standard when a person provides a written document in real life. Similarly, when providing an electronic document, a natural equivalent of the handwritten signature, using the keyboard, is the typing rhythm. This is due to the fact that the same neuro-physiological factors that permit the uniqueness of handwritten signatures also exist in a user's typing pattern [35]. A digital signature is created when a user uses a keyboard, in the shape of time elapses between keystrokes, which can be pretty consistent for familiar and repeatedly typed text [36].

In 1975, Spillane [37] pioneered the suggestion that a person's typing keystrokes can be used to identify him or her, yet the first recorded research in keystroke dynamics was not until

1977 when Forsen et al. [38] first examined the possibility of distinguishing between users based on the way they type their names. Three years later, Gaines et al. [39] investigated the likelihood that users could be identified by the typing patterns that they follow when typing long pieces of text. Since then, researchers began to show interest in proving the hypothesis that typing patterns can be used for user authentication. Experiments were conducted to find typing features that can be used effectively for user authentication. Results from these tests showed that the similarity between typing samples from the same person is high with respect to the time delays it takes the user when typing one key or two successive keys [19]. These studies have also investigated the ability of several techniques to achieve the correct decision about the user's identity. Some of this research will be examined later in this Thesis.

2.2 Biometric Characteristic of Keystroke Dynamics

Specific properties have to be present and valid in the system at all times to ensure that keystroke dynamics is considered a biometric feature that can be used to distinguish between users with high accuracy [25]. These properties are briefly described below:

- Universality: every person should have this feature.
- Distinctiveness: every two users should be adequately different in this feature.
- Permanence: this feature value should not change after a period of time.
- Collectability: this feature can be quantitatively measured.
- Performance: the system using this feature is accurate.
- Acceptability: this feature is ethically accepted by users.
- Circumvention: this feature is hard to deceive.

In the case of keystroke dynamics, each user can easily utilize the keyboard to type-in text, which fulfils the universality property. The timing features extracted from the user's typing sample is considered distinctive enough to recognize a user for verification purposes more than that for identification purposes. Keystroke dynamics is, unfortunately, variable through time due to the physical or mental state of the user at that specific period of time, which reduces the level of the permanence property. Keystrokes are easily collected while users are typing on the keyboard, whether it was a long e-mail of several paragraphs or even a single password. The performance of keystroke dynamics systems varies a great deal depending on the features and classification methods used and on the environment surrounding the experiment. The privacy concerns of users, whom might not accept the idea of their

keystrokes being stored and analysed, may lower the level of the acceptability of the keystroke dynamics, particularly in the situation of typing sensitive information. Keystroke dynamics are very hard to imitate but, nevertheless, replay attacks are still a serious threat. Table 2.1 summarizes the level of fulfilment that keystroke dynamics have for each property as stated in [25].

Table 2.1: Biometric properties of keystroke dynamics.

	Property						
Feature	***Universality***	***Distinctiveness***	***Permanence***	***Collectability***	***Performance***	***Acceptability***	***Circumvention***
Keystroke Dynamics	High	Medium	Low	High	Medium	Medium	Medium

2.3 Phases of Keystroke Dynamics Authentication

There are two main phases that a user has to go through in order to be authorized by keystroke dynamic systems; namely: the enrolment phase and the log-in phase [19]. The first phase has to do with collecting data about the user such as username and password in addition to capturing the user's typing behaviour [40]. The system gathers the keystroke times and extracts the timing features to create a template for each user's typing behaviour. This template, also referred to as a user's profile, is stored in a database that corresponds to other details of the user.

The second phase takes place whenever the user needs to actually use the system. At that time, the system collects the user's keystroke times and, then, extracts the timing features in the same manner as followed in the enrolment phase. After that, the system performs feature matching with the user's template, which is stored in the database [40]. Based on the results of the matching process, one of two actions will take place: granting access to the user if the two sets of data are sufficiently similar or otherwise denying access to the user. Figure 2.1 shows the flow of these two phases.

These two phases were represented in all keystroke dynamics studies by conducting five main experiment parts in the following order: recruiting participants, requesting a typing task to be done by the participants, collecting the keystrokes timing data, obtaining timing features from the raw keystroke data, training the classifier using part of the keystroke data and using the other part for testing the classifier [41]. Section 2.8 will go through the previous

mentioned stages in order to compare and contrast what has been done in this area as reported in the current literature.

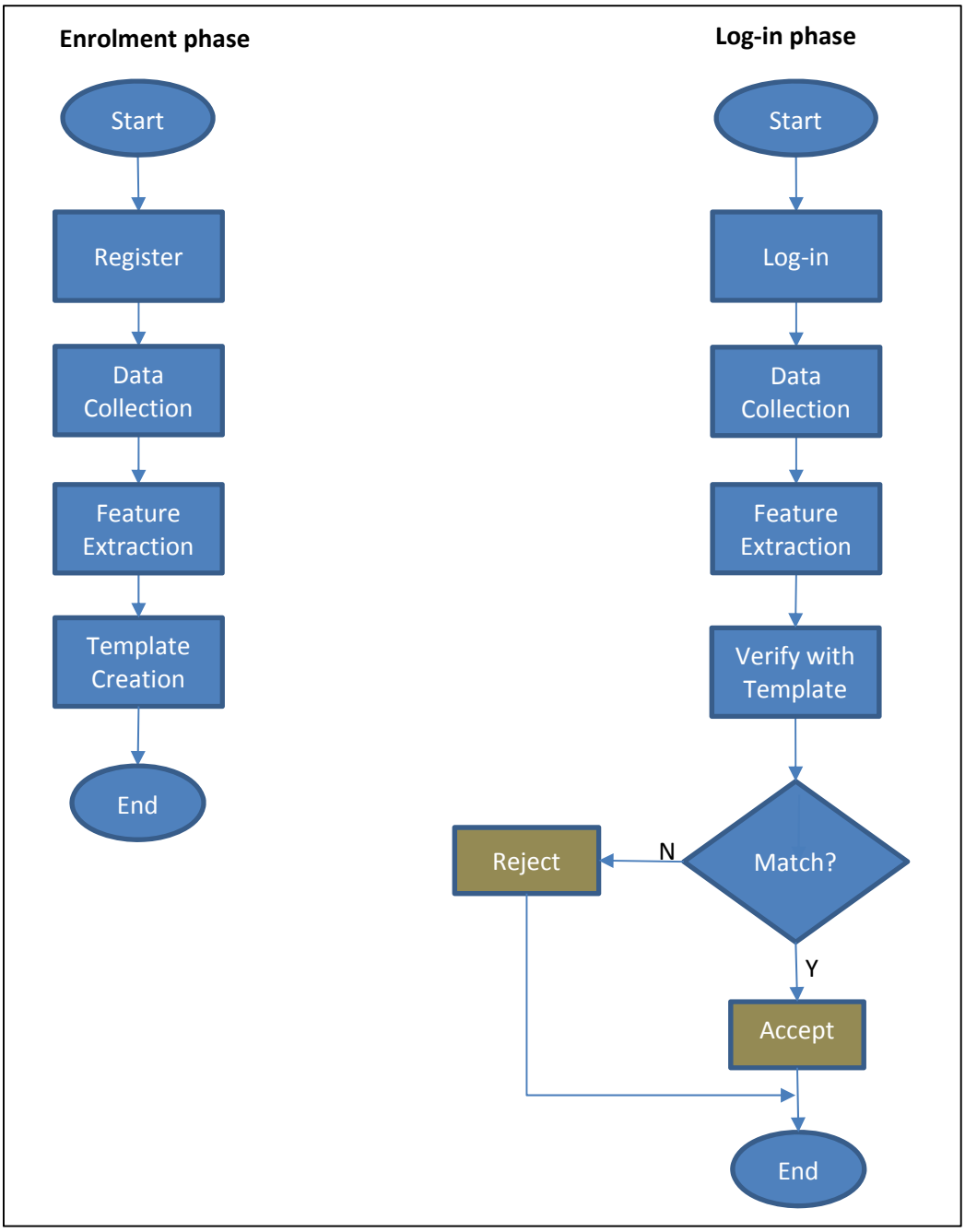

Figure 2.1: Flow of the two keystroke dynamics phases.

2.4 Advantages and Limitations

There are several reasons that make keystroke dynamics a strong contender for replacing the ID/password scheme as it provides good level of security while sustaining the usability level that the system's users are mostly intrigued by. First of all, it is relatively cheap compared to

other biometrics as it requires no additional hardware [42]. Moreover, since keystroke patterns depend on the user's behaviour, it is considerably difficult to copy by others and impossible to be lost, stolen, or loaned [4]. It also provides transparent authentication since it is completely non-intrusive, i.e. the user performs it unknowingly. Therefore, it is considered the most natural method for authentication in computer systems, which depend mainly on the keyboard's input operation [19].

Additionally, keystroke dynamics can be used in multi-factor authentication as a means of strengthening the password [13]. This is done in an unobtrusive manner, since the user already needs to enter his or her password for authentication, during which the keystroke timings are captured. Therefore, it is not likely that the impostor can gain access to the system if he or she had illegally obtained the password. It also provides the potential for performing continuous authentication during the whole time that the system is in use, as opposed to only providing authentication at the beginning of a session [43, 44].

In addition to the physical safety it provides for PC's and laptops, it is also used for online services protection, since keystroke data can be gathered from anywhere in the world that has access to a keyboard and an internet connection, and with the aid of special software such as client-side Java applets [45]. Furthermore, this method is not only limited to computers, all devices equipped with a keyboard can benefit from it. This includes devices like tablets, smartphone, ATM machines etc. [46].

Nonetheless, similar to other biometrics, keystroke dynamics has some limitations that may affect its performance; especially because it is such a delicate measurement which is affected by changing the keyboard used for collecting data. It is also sensitive to the user's mental and physical condition [47]. Certain conditions can change the typing pattern that the user normally follows such as: state of mind, fatigue, injuries etc. [48]. Moreover, other external conditions can also cause the user's typing patterns to vary. Text choice, text length, entry mode etc. are all factors that can cause the user to behave different than normal [49].Typing errors also cause the user to behave in a different way, thus causing the system to react differently [50]. Furthermore, since it depends on the human behaviour, typing patterns are subject to change over time, with no obvious reason [18], which lowers its level of reliability.

Moreover, due to the little amount of information that can be inferred from typing data, large amount of data are required to train the classifiers used for the authentication process [19].

In addition, the amount of false rejects that a genuine user may face is high compared with other biometrics. It was found in [51] that 30% of the users will face an account lockout before their 50th authentication session. Lastly, despite the amount of research that had already been conducted in the area of keystroke dynamics authentication, the performances of these different studies are very difficult to compare. This is due to the huge amount of factors that might vary from one experiment to another [4].

2.5 Classes of Keystroke Dynamics

Two types of keystroke systems are discussed in the literature, for performing authentication in computer systems. These two classes are: fixed-text and free-text keystroke systems, also referred to as static and dynamic [18]. This section discusses the two classes in some detail, yet Section 2.8 will examine some of the significant studies conducted in each of these classes.

Fixed-text forces the users to use only a pre-defined text to produce the typing samples. The pre-defined text varies in the research done in this area in the way that some have utilized the same shared password for all users [52] and others used different short fixed text for each user such as using the user's name [53], log-in IDs [54] or passwords [12]. Other research utilized long phrases of fixed text [55], while others focused on short words [56]. The main function of the fixed-text systems is applying the authentication scheme at log-in time in order to verify the user's identity at the beginning of the session only [19]. This is done by forcing the user to retype their password, or any predefined text, a number of times at the enrolment phase in order to determine the user's typing rhythm for that specific password. This is considered a critical usability issue because of the extra load it adds to the user. In addition, the user still needs to memorize the predefined text in order to use it at each log-in. Generally speaking, fixed-text keystrokes are mainly used for strengthening passwords [13, 57].

When the user is typing his or her password, in fixed-text keystroke dynamics, the system not only checks the accuracy of the password but also the typing manner that the user followed. Although Monrose et al. [13] have demonstrated that keystroke dynamics have the ability to make even weak passwords protected against brute force attacks, Song et al. [58] denoted that if the time between the keystrokes is exposed whilst the user types a certain password, it will considerably reduce the effort and time that it will take an attacker to guess the password

using the brute force attack. A valid solution for this problem is to use encryption to conceal the keystroke latency time, as done in [59].

The issue of the user's increased familiarity with the password, after using it for a considerable amount of time was pointed out in [34]. This causes the user's typing pattern to change and the overall typing speed to increase, which will negatively affect the authentication process. Moreover, the authors of [60] indicated the problem that occurs when a user wishes to change his or her password. In that case, the system will need to re-learn, which requires the user to go through the enrolment process again and re-type the new passwords repeatedly every time the password is changed.

Most of the early research in keystroke dynamics only focused on the keystrokes generated by typing fixed words, starting as early as 1975. In fact, the majority of work in the field of keystroke dynamics was performed using fixed text [18].

Free-text keystroke systems, on the other hand, don't restrict users to a particular text; on the contrary, users are given complete freedom to use any text of any length without any constraints. Free-text authentication can be carried-out either periodically or continuously [4]. Unlike fixed-text, free-text systems can continue to collect the keystrokes, after successfully passing the log-in session, throughout the whole time that the user is logged-in. This can then ensure the identity of the user during the full duration of that session [40]. In continuous authentication, a static authentication is performed first at log-in, and then after, continuous authentication is carried-out during the remaining time of the session. Please refer to Figure 2.2 for further details about the flow of continuous authentication. In free-text systems, the user's typing pattern is typically monitored during several days, in which he or she is performing regular typing tasks such as writing e-mails or typing word documents. While both free-text and fixed-text systems are quite similar in the way that they utilize the key press and release times to build a user's behaviour profile, they clearly differ in the way that the system is trained and applied [42].

In 1980, Gaines et al. [39] first utilized long text in free-text authentication, and in 1995, Shepherd et al. [61] was the first to show interest in continuous free-text authentication. In 1997, the first organized attempt to use a free-text keystroke system was conducted by Monrose and Rubin [33] where both fixed-text and free-text were used. The overall performance was not encouraging for free-text giving only 23% correct classification, while fixed-text produced about 90% correct classification. This shows the complexity of using

free-text systems compared with the fixed-text systems. Nevertheless, free-text systems have gone a long way since that experiment and much better results have been obtained using more sophisticated techniques.

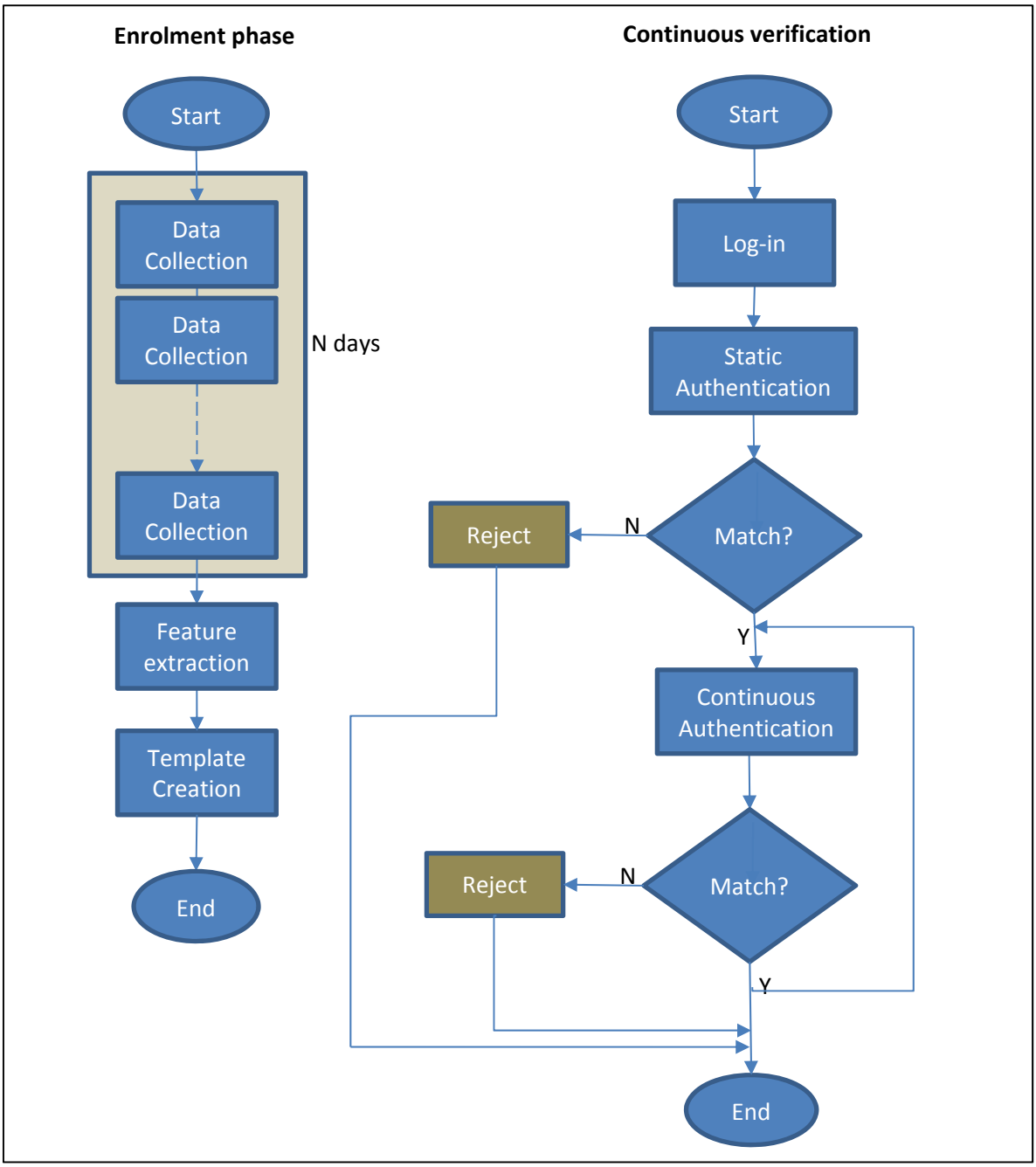

Figure 2.2: Continuous authentication.

Free-text keystrokes suffer from producing extremely long character streams, which has to be profiled in the most generalised manner over the whole system [62]. In addition, it is subject

to more noise, since it is more likely for the user to introduce pauses in longer typing tasks compared with short and familiar passwords in the case of fixed-text keystrokes [62]. Furthermore, the usefulness of free-text keystroke systems had decreased since the development of the Graphical User Interface (GUI), which resulted in reducing the amount of typing that a user goes through when using his or her PC [63]. In contrast to using console systems, such as MS-DOS, in which users type commands to perform all operations, almost all operations can be performed via mouse clicks in the GUI system. This also applies to some services such as online banking, in which there is often no chance for entering text except for the time when the user logs-in [4].

2.6 Keystroke Features

After obtaining the users' raw data, extracting the keystroke features is performed [64]. These features are computed using two main values, specifically: the press time (D_n) and the release time (U_n) of each key (n), in milliseconds.

One keystroke feature is extracted from the timing of a single pressed key as shown in Figure 2.3. It had been used solely as a timing feature in [65] and coupled with other timing features in [45, 66, 67]. This feature is explained below:

- Duration time (also called dwell time or hold time) (H). It is the time a key is pressed until it is released. It can be computed using the following formula:

  $$H_i = U_i - D_i \quad (2.1)$$

Three keystroke features are extracted from the timing of each di-graph, i.e. two successive keys, as shown in Figure 2.3. These features are called keystroke latencies or flight times. The three types of keystroke latencies are listed below:

- Down-down time (DD) (also called press-press time (PP)): It is the interval time between two successive key presses; used in studies such as [68, 8, 69]. It can be computed using the following formula:

  $$DD_i = D_{i+1} - D_i \quad (2.2)$$

- Up-up (UU) time (also called release-release time (RR)): is the interval time between two successive key releases; utilized in [70, 52]. It can be computed using the following formula:

  $$UU_i = U_{i+1} - U_i \quad (2.3)$$

- Up-down (UD) time (also called release-press time (RP) or inter-key time): is the interval time between a key release and the next key press; examples of studies exploiting it are [49, 71, 50] . Moreover, UD time can be a negative value in the case that the next key is pressed before releasing the previous one, which can happen in the case of very fast typists. It can be computed using the following formula:

$$UD_i = D_{i+1} - U_i \quad (2.4)$$

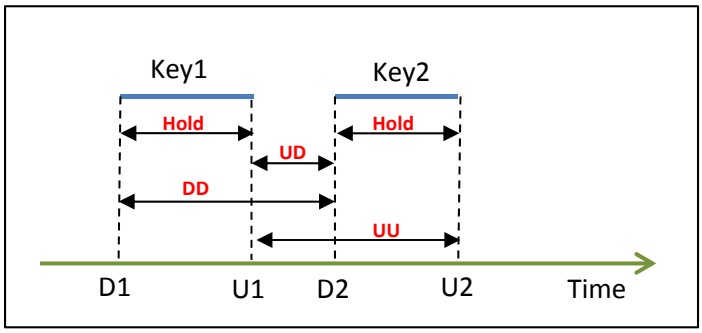

Figure 2.3: Keystroke timing features.

Most studies employed more than one of these latencies at once [72, 52]. Moreover, some studies claim that using hold time yields better results [54], while others argue that latency features produce better results [69]. Nonetheless, many studies deduced that using combinations of both hold and latency times have the best effects on the system performance [66, 72].

Moreover, the down-down (DD) latency was utilized between more than two keys. Bergadano et al. [73] and Bond & Awad [74] employed it on tri-graphs, i.e. three keys; while Gunetti and Picardi [8] applied it to n-graphs, i.e. 4-graph, 5-graph and 6-graph. Using a larger 'n' increased the authentication performance based on the work done in [73].

Although, di-graph and tri-graph time has been used in plenty of research, Sim and Janakiraman [75] concluded from their several experiments that using di-graphs/tri-graphs is not a good discriminator between users when the actual typed words are not taken into consideration. This is because the context of the text that a particular letter is included in regulates the manner in which it is typed [44] i.e. the letters 'te' have different latency times in the words 'sentence' and 'teacher'. Therefore, di-graphs/tri-graphs are more effective for keystroke dynamics when using context-specific n-graphs.

In many keystroke systems, users' profiles consist of the mean and/or standard deviation of each key hold time and/or di-graph latency [33, 76]. Furthermore, only the latencies' means and standard deviations for di-graphs that have occurred a minimum number of times are included in the users' profiles in some studies such as [77, 78]. Moreover, only a fixed set of letters and two letter combinations were used in [44, 79]; these sets were chosen based on each letter's frequency in the English language. Letters including E, A, T etc. and di-graphs including: AT, TH, HE etc. are frequently found in English text, therefore, it is good practice to use the mean and standard deviation of their duration and latencies in the user's template, which will increase its stability.

Other less common features were considered as well. Typing speed or words-per-minute (WPM) was used in [70]; it is the number of words that a user types in a minute. Frequency of error was also used in [80]; it measures the number of times that the user makes use of the backspace key. Shift key usage was used as well in [57]; it calculates the percentage of using the right and left shift-keys. Moreover, the additional keys usage was used in [49]; it computes the frequency of using special keys such as the num-pad and the control keys. Then again, to use these features, a piece of text with considerable length has to be typed in order to correctly capture the user's habits.

Another feature that was used for inferring the typing behaviour for users, was the typing pressure [81, 82]. This feature computes the pressure applied to each key being pressed on the keyboard, whilst the user is typing. This feature suffers from the drawback of having to use a special kind of keyboard, which defeats the purpose of using keystrokes in the first place.

Other features that also suffer from having to use additional hardware include the finger placement feature [83]. This feature analyses the positioning and the angle that the fingers are applying on each key being pressed. Even the finger choice for pressing each key on the keyboard is studied as a feature for keystroke dynamics [84]. Moreover, the shape and position of hands while typing were also used as features to infer users' typing behaviour [85]. Unfortunately, all these features require an external camera to capture the shape, position of the hands and the placement, and choice of the fingers during the time of authentication.

Quite a recent study has investigated the use of keystroke intensity as a feature for user authentication [86]. Keystroke intensity overcomes the problem of typing pressure, hand

shape and position, finger placement and choice as it does not impose the need for an additional camera. It is captured using the PC's built-in microphone which is used to acquire an audio recording of the user's typing. Nevertheless, this feature enforces a semi-controlled environment in which the user has to be alone in a quiet room to carry-out the typing.

Table 2.2 lists all the keystroke features used for user authentication.

Table 2.2: Features of keystroke dynamics.

Category	Features
Conventional Features	Hold time Down-Down time Up-Up time Down-Up time
Non-conventional Features	Typing speed Error rate Shift key usage Special keys usage
Additional Hardware Features	Typing pressure Fingers placement and choice Hands shape and position Keystroke intensity

2.7 Performance

Unfortunately, not only keystroke systems but all biometric authentication systems sometimes suffer from mistakes in the authentication decision. Some reasons that might influence the accuracy of the system are related to the classification method of choice, the features included in the timing vector, and the quantity of training data [4].

It is possible for an imposter to be mistakenly identified as the legitimate user if, by chance, the typing patterns of these two individuals are close enough to the extent that the classification method fails to distinguish between them. Conversely, when one of the legitimate user's fingers slips off the keyboard and causes the typing pattern to change slightly, the user may not be successfully authenticated. Thus, it is important to have some

metrics to exactly measure the error rate; this helps to identify the performance level that can be expected and tolerated by that system's users [87].

There are four possible results in a two-class problem which should be taken into consideration in any pattern classification system such as keystroke dynamics authentication [25]. The classifier will produce either a positive classification, if this is the legitimate user, or a negative one, if this is an imposter. If the actual classification is positive which means that the user under study is the legitimate user, then the result is true accept in the case where the classifier predicted a positive classification and then the result is otherwise a false reject. In the same manner, if the actual classification is negative, which means that the user under study is an imposter, then the result is false accept in the case where the classifier predicted a positive classification and the result is otherwise a true reject. Table 2.3 demonstrates all the possible result combinations in a more concise manner. True accept and true reject demonstrate the quantity of correctly classified users. False accept and false reject, on the other hand, are used to compute the system's error rate [25].

Table 2.3: Two-class decision result's combination.

		Predicted class	
		Positive	***Negative***
Actual class	***Positive***	True accept	False reject
	Negative	False accept	True reject

There are many methods to evaluate the performance of a keystroke dynamics system, thus a variety of error rates can be found in the current literature.

A very simple way to measure the error rate was used in earlier studies; using the accuracy measure, which is the percentage of successfully classified attempts compared to the total number of completed attempts; this technique was adapted in [33, 71, 68]. Quite the opposite, a misclassification error is the percentage of incorrect classifications compared to the total number of attempts; it was applied in [88]. Both rates are computed using the following equations:

$$Accuracy = \frac{Correctly\ classified\ attempts\ (true\ accepts + true\ rejects)}{Total\ number\ of\ attempts} \quad (2.5)$$

$$Misclassification\ rate = \frac{Incorrectly\ classified\ attempts\ (false\ accepts + false\ rejects)}{Total\ number\ of\ attempts} \quad (2.6)$$

The most frequently used error rates for determining the performance of an authentication system are: the False Accept Rate (FAR), also referred to as the Imposter Pass Rate (IPR) and the False Reject Rate (FRR), also called the False Alarm Rate (FAR). The FAR is the percentage of impostors who have successfully gained access to the system, while the FRR is the percentage of legitimate users who have been denied access to the system. These two error rates were used by the majority of free-text keystroke systems, including [8, 50, 80]. These error rates are computed using the following two formulas:

$$FAR = \frac{Number\ of\ false\ accept\ attempts\ (accepted\ imposters)}{Total\ number\ of\ impostor\ attempts} \tag{2.7}$$

$$FRR = \frac{Number\ of\ false\ reject\ attempts\ (rejected\ legitimate\ users)}{Total\ number\ of\ legitimate\ attempts} \tag{2.8}$$

When looking at the numbers produced by both FAR and FRR, the smaller these values are, the more secure the system under study is. As shown in Figure 2.4, there is a trade-off between the FAR and FRR which can be controlled according to the strictness level of security required [4]. FAR is required to be as low as possible in highly secured applications, even though a higher FRR compromise might be experienced. Meanwhile, a higher FAR is somewhat acceptable in systems where security is not the major aim yet the system's usability is of higher priority.

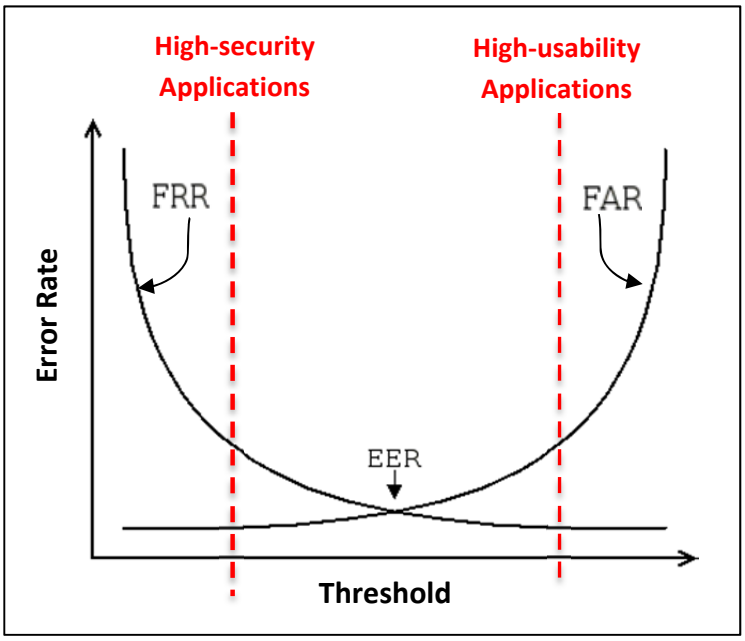

Figure 2.4: An example showing the relationship between FAR, FRR, and EER.

The other commonly used error rate is the Equal Error Rate (EER), also referred to as Cross-over Error Rate (CER), which represents the value where FAR and FRR are equal. It was used in many studies such as [68, 69, 89], where lower EER values indicate a more secure system. Figure 2.4 demonstrates the relationship between FAR, FRR, and EER with respect to different threshold values and how it reflects on the security level of the application in-use.

ZeroFAR and ZeroFRR are two other error rates used in [12, 66]. ZeroFAR corresponds to the FRR value when the FAR is equal to zero and, likewise, ZeroFRR corresponds to the FAR when the FRR is equal to zero. This is performed by setting the threshold so that FAR becomes zero in case of ZeroFAR and, similarly, setting the threshold so that FRR becomes zero in case of ZeroFRR.

The Receiver Operating Characteristic (ROC) curve is another performance measuring method. It finds the relationship between the FAR and the True Accept Rate (TAR), which is the percentage of legitimate users correctly classified. A high accuracy system would have this curve plotted closer to the upper left corner of the diagram, as shown in Figure 2.5. This error rate was used in studies such as the one conducted in [67].

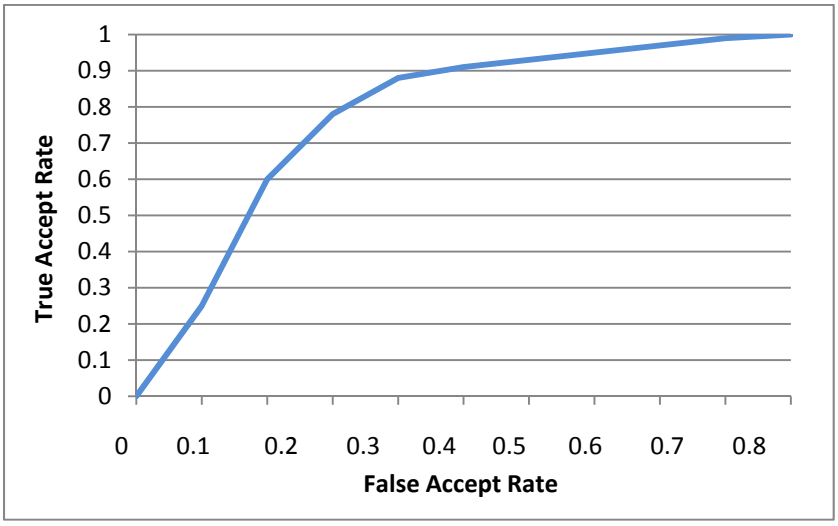

Figure 2.5: An example of ROC curve.

Due to the fact that free-text keystroke authentication can be a continuous process, another metric was proposed in some studies. This metric exactly defines the amount of time, in number of keystrokes, that it takes for the system to discover that an imposter has had access. This aims to detect the impostor as fast as possible, incorporating as few keystrokes as

possible, which implies that an attacker would be detected before he can do more harm to the system. The number of keystrokes performed before the imposter was detected was used in [44, 77].

Giot et al. [90] introduced a Failure To Acquire Rate (FTAR) in their paper, which investigates the acquisition process of keystroke data. This error rate identifies the ratio of acquisition difficulties that face the users. A common example of such difficulties is typing mistakes in fixed-text keystroke dynamics. Such mistakes compel the users to delete the word being typed and start all over again. This might cause some frustration for the users, which deteriorates the usability of the method in-hand.

The fact that there is no major method used for measuring the performance in the studies relating to keystroke dynamics authentication, makes it difficult to compare the results of these studies. This has a critical impact on the overall perception of the methods used and may result in incorrect conclusions about them [25].

2.8 State of the Art in the Area of Keystroke Dynamics Recognition

As mentioned previously, keystroke dynamics have been around, in one form or another, since the late 1800's. Having said that, it wasn't until 1975 that the first preliminary tests were conducted. An enormous amount of work has since been done in the next 40+ years. Before going into details of what has been done in fixed-text and free-text keystrokes, it is worth noting that many of the methods used for classification have been utilized similarly in both these systems. The methods used for classification were categorised in [19] to range from simple statistical methods to more complex pattern recognition and neural network algorithms. Moreover, an even more sophisticated combination of methods was used in many cases.

Simple statistical methods were used as a classification scheme for typing patterns in several keystroke systems studies; in which the mean and/or standard deviation of the hold and/or latency times were used for differentiating between users, as performed in [34, 78]. A variety of distance techniques have been used; such as: Euclidean distance [50], weighted Euclidean distance [76], scaled Manhattan distance [44] and Bhattacharyya distance [75]. Other

statistical methods such as Markov chain model [89] and K-nearest neighbour [12, 68] were also utilized.

Pattern recognition and machine learning methods were also utilized as classification methods for keystroke authentication. For example: Bayes classifier [54], support vector machines (SVMs) [70], decision trees [71] and Random Forests [57] were used.

Moreover, Neural networks were also implemented for the purpose of distinguishing between the typing behaviour of different users, as in [12, 54, 91]. Although using Neural networks yielded good results, it is considerably slow to apply and train.

Furthermore, some research has considered fusion in keystroke dynamics. One of which is the work done in [72] which proposed a decision-level fusion between two methods. The first being the Gaussian similarity score between a reference template and a test data template. The second being the Direction Similarity Measure (DSM) for comparing the typing patterns of the user. The two scores were fused by using a weighted sum rule. Combining the two methods delivered more accurate results compared with using each method on its own [92].

A brief description of some of the research conducted in fixed-text and free-text keystroke dynamics is presented here. A chronological overview of the various algorithms deployed in each class is stated, in addition to the experimental details in the different studies, such as the number of participants and samples and the typing text used for training and testing the systems.

2.8.1 Fixed-text Keystroke Dynamics

It was 1975 when Spillane [37] initially suggested using the keyboard as a means for collecting keystrokes for identification purposes. Two years later, tests were conducted by Forsen et al. [38] to check if keystroke dynamics could be used as a mechanism to differentiate between typists. The participant's name was typed by its owner and by other participants. Statistical comparisons proved that the manner that a user types his or her name was distinguishable from the way another user typed the same name.

Three years later, Card et al. [93] suggested a psychological model for explaining the human interaction with computer programs on a keystroke level. It summarizes a human-machine session as:

$$T_{task} = T_{acquire} + T_{execute} \quad (2.9)$$

Where $T_{acquire}$ is the time needed to assess the task and decide how to solve it. This differs between users based on the difficulty of the task and the user experience with it, therefore it is not considered a good way to distinguish between users. While, on the other hand, $T_{execute}$ is the actual time needed to perform the task. It basically describes a mechanical act, so it can be described, in the case of a typing task, as:

$$T_{execute} = \sum_{i=1}^{n}(T_{mi} + T_{ki}) \quad (2.10)$$

Where T_m is time for mental planning and T_k is the time to perform the key press for character (i). The sum of this amount for all characters (n) denotes the execute time for a subsection of the text. Moreover, the text subsections are on average between 6 and 8 characters long; this is the number of characters that the brain can handle at once [94]. Therefore, keystrokes less than 6-8 are good candidates to show clear keystroke patterns.

Another related theorem is Fitts's law [95]. It defines the connection between the target distance, width, and time needed to perform a target acquisition task. It represents the speed/accuracy trade-off in rapid, aimed movement [96]. The movement time is computed using Fitts's law as follows:

$$MT = a + b\ log_2\ (1 + \frac{D}{W}) \quad (2.11)$$

Where MT is the movement time, a and b are empirically determined constants, that are device dependent, D is the distance of movement from start to target and W is the width of the target.

Fitts's Law can be physically interpreted as: big targets at close distance are obtained faster than small targets at long range [97]. This can be applied to the keyboard layout where the modifying keys that are pressed most often, i.e. space, enter and shift, are larger than other modifying keys. They are also closer to the alphanumeric keys which are frequently used [98]. These two characteristics make these keys, i.e. space, enter and shift, easier to hit. Moreover, Fitts's law aims to reduce the total travel distance between keys. Therefore all alphanumerical keys (frequent keys) are located in the centre of the keyboard. Moreover, frequently connected letters (such as T and H) are located closer to each other compared with less frequently connected letters [98].

Although the width of all alphanumerical keys is equal, the distance between two adjacent keys is smaller than non-adjacent ones. This will cause higher accuracy of movement between adjacent keys compared with non-adjacent ones. This accounts for the non-adjacent key-pairs having more outlier or noisy data compared with adjacent ones. This is because users are more likely to miss when hitting a key that is located farther away. In addition, fitts's law validates the idea behind grouping keys based on adjacency, i.e. distance between each other. As adjacent keys are closer to each other, they are pressed with higher speed compared with 2nd-adjacent ones. The same applies between 2^{nd}-adjacent keys and 3^{rd}-adjacent ones and so on.

A U.S. patent was granted to Garcia [53] for an identity verification method that involved the use of the user's name as a keystroke generator. Garcia stated that the timing latencies produced by the user when typing his or her name had been proven to be stable enough to use for identity verification, therefore, the best data that can be used for authenticating a user is collected when the user types his or her own name. In his study, Garcia, asked the users to type their names a number of times. This data was used to create a vector of mean latencies for this user's profile (electronic signature) after discarding any outlier data, i.e. values larger than 500% of the mean. At log-in time, the user was also required to type his or her name. This data was used to create the keystroke's latency vector of the test data. The two vectors were compared using the Mahalanobis distance function. The user was denied access to the system if this distance was more than 100; and likewise, the user was granted access if the distance was less than 50. In addition, in the case where the value of the distance was between 50 and 100, the user was required to retype his or her name again. Garcia produced an overall performance of a 0.01% FAR and a 50 % FRR.

Similarly, Bleha et al. [99] used latencies between keystrokes as timing features in their experiment, in which two different classification methods were used; the minimum distance classifier and Bayesian classifier, each of which used a different threshold. Both classifiers were used jointly in the classification process; i.e. the user was rejected only if both classifiers agree that the attempter was an imposter, and vice versa in case of accepted users. This, unfortunately, was the reason for some results being undecided. The authors conducted three recognition systems. First of which was an identification system where 10 users retyped a fixed phrase five times. An indecision error of 1.2% was produced. The second recognition system was a verification system, in which 26 users were asked to retype their names 30 times. An FRR of 8.1% and a FAR of 2.8% was generated. The third system was an overall

recognition system, i.e. combining the two systems. In this part of the study, 32 users' data, 10 genuine and 22 imposter, were used to achieve an error of 3.1% FRR and 0.5% FAR.

Joyce and Gupta [36] agreed with Garcia [53] in the sense that the timing data extracted from typing a well-known string provides more consistent results, compared with unfamiliar strings. They used a statistical method that employed the absolute distances between the timing vectors of the user's template and test data. Each of these vectors consisted of the means of di-graph latencies for a fixed-text, i.e. digital signature. This digital signature consisted of the username, password, first name and last name of the user. All outliers, i.e. values as three times standard deviations greater than the mean, were discarded. A suitable threshold was identified for each particular user based on the mean and standard deviation of his or her template data. This threshold was used for deciding if the test data was actually for the user in question. More specifically, the user was authenticated only if the distance between his or her test data and profile data was less than the threshold. Thirty-three users were asked to go through the training session, in which they provided eight digital signatures each. Then, they were allowed to re-enter the system using that same digital signature, five times for testing. In the rest of the testing process, twenty-seven users acted as imposters and targeted the remaining six users by entering the victims' digital signatures five times. The study produced an FAR of 0.25 % and a FRR of 16.36 % over all the tested data.

Although the results produced by Joyce and Gupta [36] were encouraging, they used a replacement methodology in their tests, in which they asked users to provide training data all in one session. Monrose and Rubin [33] suggested the use of data sets that had been recorded at variable times to improve the dataset created by Joyce and Gupta in [36] and therefore make it more reliable. In addition, their study allowed the participants to download the data collecting program on their personal PC's rather than restricting them to use a specific PC as done in [39, 53, 36]. This allowed them to run the experiment at times when it was most convenient for them. The collected users' profiles were represented as n-dimensional feature vectors, which were created using the same method developed by Joyce and Gupta in [36]. Three different experiments were conducted on the collected data. The first utilized the Euclidean distance between the vectors of the test data and the profile data. The second used a non-weighted probability scheme. The last experiment adds a weight to the probability measure; this weight was based on the frequency in which the di-graph occurs in the written language. Over a period of 11 months, typing data was collected from 63 users. The rate that users were correctly identified using the weighted probabilistic classifier was approximately

87.18%, which was better than the performance of the Euclidean distance which was 83.22% and the non-weighted scoring approach being around 85.63%. A better performance was produced using a fourth scheme, the Bayesian classifier, which resulted in around 92.14% correct identification.

One of the first research that used Neural Networks for keystroke dynamics authentication was produced by Obaidat and Macchairolo [100]. In this study, they presented a new multilayer neural network system to be used for identifying users based on their typing patterns. Only the down-down time, between two consecutive keystrokes, was used. The experiment made use of three different types of networks, namely: a multilayer feed-forward network which was trained using the back propagation algorithm, a sum-of-products network which was trained with a modified version of the back propagation algorithm, and a new hybrid network that combines the two previous kinds. In the experiment, six participants were requested to enter the same password. This password consisted of a phrase that had thirty letters, but only the first 15 characters were used because using all 30 did not have any effect on the final results. In total, 40 vectors were collected for each user over a period of 6 months. After collecting the raw data, it was divided into odd patterns and even patterns in order to use either part in training and the other for testing. The neural network used for classification consisted of 15 inputs, corresponding to the length of the tested pattern, and 6 outputs, corresponding to the number of users, and 5 hidden units. Each of the three types of neural networks was trained similarly: training data was fed to the network and the weights were adjusted until the total summed squared error produced by the training set was smaller than 0.05. It was proven that the back propagation algorithm provided better accuracy than the sum-of-product algorithm; the accuracies for the two algorithms were 97.5% and 93.7% respectively. The new multilayer hybrid sum-of-products neural network produced accuracy of 96.2%, yet it required less training time than the other two.

Fuzzy logic was first used by Ru and Eloff [101] for keystroke patterns classification. They used flight time to distinguish between the typing patterns of the users. In this research, five fuzzy rules corresponding to the typing speed were used for classification. Thirty users were asked to type a minimum of eight-character passwords 25 times. A mean accuracy of 87% was produced.

Cho and Hwang [102] looked at the keystroke system from a different perspective. They argued that the quality of the user's patterns is very important in obtaining a high

performance keystroke authentication system and has the same significance as the quality of the classifier. Therefore, their study suggested different ways to improve the typing pattern quality using specific motions performed by the user. The concept of the typing pattern quality has to do with two issues, namely: uniqueness, which emphasises the difference between the users' patterns, and consistency, which focuses on the similarity of a single user's patterns. Several techniques were proposed to improve both factors. First, artificial rhythms were used to improve uniqueness including: pausing, musical rhythm, staccato (minimum duration), legato (maximize duration), and slow tempo. Second, timing cues were used to improve consistency including: auditory (ticking sound), visual (hammer hitting a nail video-clip), and audio-visual (combining both) cues. Five users were involved in the two experiments of this study. The chosen password was "password", which was typed naturally by the user 20 times; the different typing rhythms and cues of this password were also typed by each user 20 times. The uniqueness level was escalated significantly using these rhythms; there was a 200% improvement using short pauses and 500% using slow tempo. The consistency level has also improved using the cues; it improved almost 15 times using the auditory cues. The above strategies have also improved the discriminability of the typing pattern, which is defined as the relation between uniqueness and consistency, to more than twice using only auditory cues.

A major limitation of the scheme used in [102] was applicability to real life situations, since it only gave results from 5 users. Therefore Hwang et al. [103] used data from 25 users to investigate the level of improvement that artificial rhythm and tempo cues can add to the uniqueness and consistency of the typing pattern, in the case of a larger population. The results showed an error of 11% EER using natural rhythms; this was improved to 3% EER using pauses. An even better performance of 1% EER was produced using auditory cues in addition to pauses. This proved the validity of using these rhythms to improve the system performance.

Moreover, Maxion and Killourhy [104] examined using a keystroke dynamics identification system that uses digits only. This system can be adapted in ATMs, mobile phones, building's electronic security keypad etc. The experiment was performed on 28 subjects, each of which was asked to type-in a 10-digit number using only their right-hand index fingers. The 10-digit passcode was the same for all users and it was carefully chosen to satisfy some conditions such as: memorizing factor and digit's location on the keypad. The actual typing of the digits was restricted to using only the number-pad section of the keyboard. Participants were

requested to type this 10-digit passcode 50 times, throughout four sessions, which were held over four different days. This adds up to 200 repetitions per person; half of which were used as training data and the rest were used as test data. The features used in the timing vector were: hold time, down-down time and up-down time. The random forest classifier was used to determine the identity of the test data's owner. An EER of 8.5% was achieved when using this model, which was less than the target that the authors hoped to reach. Therefore, some improvements were added to the original model in order to increase the performance; this included: outlier data handling and incorporating practice time. This improved the overall performance of the system to achieve an EER of 1.00%. Although practice time had improved the system performance rather a lot, it was not a realistic step to include in such a system, as it adds extra load on the users.

Liu et al. [105] used duration and latency as the original features. These original features were then transformed using: Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT) and Gabor wavelets. Moreover, SVMs, Gaussian classifier, nearest neighbour were used in both the single classification model and a fusion of features and classifiers model. In this study, two large-scale databases were collected, each of which were split into two datasets. Database 1 consists of 1902 test samples and 477 training samples from 117 subjects while database 2 consists of 5089 test samples and 478 training samples from 102 subjects. The best accuracy was produced using original features and the Gaussian classifier reaching 3.6846 EER. Even though, this research has introduced many feature selection techniques and classification methods, these schemes did not produce similar results when applied to different datasets.

A summary of some of the keystroke dynamics studies employing fixed-text are listed in Table 2.4.

Table 2.4: A list of fixed-text keystroke dynamics studies.

Study	Features	Method	Subjects	Samples	Performance
Garcia [53]	Latency	Mahalanobis distance	-	-	0.01% FAR, 50 % FRR
Bleha et al. [99]	Latency	Minimum distance classifier, Bayesian classifier	26	780	2.8% FAR, 8.1 % FRR
Joyce & Gupta [36]	Latency	Absolute distances	33	429	0.25% FAR, 16.36 % FRR
Monrose & Rubin [33]	Latency, duration	Euclidean distance, non-weighted probability scheme, weighted probability scheme	63	-	85.63% accuracy
Obaidat & Macchairolo [100]	Latency	Neural network	6	240	96.2% accuracy
Obaidat & Sadoun [54]	Latency, duration	Neural network	15	3375	0.00% FAR, 0.00 % FRR
Ru & Eloff [101]	Latency	Fuzzy logic	30	750	87% accuracy
Lin [106]	Latency	Neural network	90	270	0.00% FAR, 1.1 % FRR
Robinson et al. [107]	Latency, duration	Nearest neighbour, minimum intra-class distance, nonlinear classifier, inductive learning classifier	140	2800	10% FAR, 9% FRR
Lau et al. [45]	Latency, duration, press-release ordering, typing speed	Mean, standard deviation	15	150	1.01% FAR, 1.25% FRR
Araujo et al. [66]	Latency, duration	Absolute distance	30	300	1.98% FAR, 1.45 % FRR
Bartlow & Cukic [57]	Latency, duration, shift key usage	Random forests	53	10000	2.1% FAR, 15.1 % FRR
Modi & Elliott [108]	Latency, duration	Absolute distance	40	6000	0.94% FAR, 94.87% FRR
Jiang et al. [55]	N-graph duration	Hidden Markov model, Gaussian	58	2030	2.54% EER
Teh et al. [72]	Latency, duration	Gaussian similarity, direction similarity measure	50	500	6.36% EER
Jin et al. [109]	Latency, duration	Fuzzy logic	10	1000	20% EER
Hwang et al. [103]	Latency, duration	Gaussian, Parzen window, density estimators, k-nearest neighbor, K-means, one-class SVMs	25	6300	3% EER
Rybnik et al. [56]	Latency, duration	K-nearest neighbours	250	1100	88.5% accuracy
Maxion & Killourhy [104]	Latency, duration	Random forest	28	5600	8.5 EER
Stefan & Yao [70]	Latency	PCA/SVMs	20	700	4.2% FAR
Narainsamy & Soyjaudah [15]	Latency, duration	Neural network	40	400	1.00% FAR, 8% FRR
Giot et al. [52]	Latency	SVMs	100	1200	15.28% EER
Abualgasim & Osman [110]	Latency, duration	Statistical weight	26	130	20% FAR, 0.00 % FRR
Kekre et al. [111]	Latency, duration	Relative Entropy, Euclidian distance	33	330	55%EER
Raghu et al. [12]	Latency	Neural network	21	4347	0.00% FAR, 1.1 % FRR
Zhong et al. [112]	Latency, duration	Nearest neighbour	51	20400	8.7% EER
Rezaei & Mirzakochaki [113]	Latency, duration	Linear discriminate classifier, quadratic discriminant classifier, k-nearest neighbour	24	240	19.20% FAR, 0.81% FRR
Gunathilake et al. [65]	Duration, frequencies	Deviation of duration and frequencies	48	-	0.10% FAR, 0.12% FRR
Hansen & Willmore [86]	Latency, keystroke intensity	Naive Bayes, neural network	14	1317	1.00% FAR, 25% FRR
Liu et al. [105]	Latency, duration	SVMs, Gaussian classifier, nearest neighbour	50	3154	7.66% EER
Bours & Komanpally [114]	Latency, duration	Manhattan distance, scaled Manhattan distance, Euclidean distance, scaled Euclidean distance	28	5600	13.0% EER

Descriptions of latency, duration and n-graph duration are found in Section 2.6. Definitions of accuracy, FAR, FRR, ERR and keystrokes are found in Section 2.7

2.8.2 Free-text Keystroke Dynamics

One of the early works done on free-text keystroke dynamics was that done by Gaines et al. [39] in 1980. In that research, data was collected from six professional typists. The typists were requested to type-in three paragraphs of text at two different times, which were four months apart. The data from the three different passages were then used in combination because it was found that there was only little information when using the three passages individually. The three passages included: ordinary English text, a collection of random words, and a collection of random phrases, respectively. The study relied on the down-down time of di-graphs that had occurred at least 10 or more times. Removing the outliers was then performed for di-graph latencies that exceeded 500 milliseconds. The classification process was done using a two-sample t-test for each user. The FAR for this study was zero and the FRR was about 4%. A further enhancement was applied to this study; using only core digraphs which are these five di-graphs: "in, io, no, on, and ul" in the testing process. This yielded in perfect authentication performance, zero FAR and FRR. Although the results are very encouraging, the number of participants involved in the experiment was significantly small.

Likewise, Umphress and Williams [34] used the mean and standard deviation of all latencies to represent the users' profiles. Samples with errors and standard deviation over 0.75 were discarded. Only the first 6 letters of each word were considered which matched with what has been already discussed by Card et al. [93]. This study utilized two tests, the first of which compared each of the latencies in the test data with the corresponding di-graph in the profile matrix; if it was within 0.5 standard deviation from the mean, it was considered a valid di-graph; the ratio of valid di-graphs governed the acceptance of the test data as being from the original user. The second test applied a two-tailed t-test to the overall means. Seventeen people were asked to take 2 typing tests separated by several days. The first was the user profile, including 1400 characters, and the second was the test sample, consisting of 300 characters. The error rates of the study were found to be 12% FRR and 6% FAR.

The R measure, or degree of disorder, was first introduced by Bergadano et al. [73]. The degree of disorder was calculated by summing the distances between the orders of each element in the two samples. It was used to find the distance between two samples based on the tri-graph duration time of these samples. The tri-graph duration is the elapsed time between the press of the first and of the last keys of a three key sequence. Forty-four

volunteers were requested to re-type a text of 683 characters five times. This produced a total of 220 samples, which were used for genuine testing. In addition, 154 participants produced 71500 samples for imposter testing. An overall of 0% FAR and 2.3% FRR was produced in this study.

One of the most cited free-text studies is that conducted by Gunetti and Picardi [8], in which they aimed to refine the algorithm used by Bergadano et al. [73]. This study depended on two measures, the first of which was the relative measure (R measure), which was developed by Bergadano et al. to find the degree of disorder between the two samples; similar to [73]. Unfortunately, the R measure was not always enough, i.e. if the typing speed of all the di-graphs in one sample are exactly twice as the other sample, the distance between the two samples will be zero and fail to differentiate between samples. Therefore, the absolute measure (A measure) was introduced to calculate the absolute distance between the two samples. In both the R and A measures, only down-down time of n-graphs occurring in both typing samples was considered. These n-graphs include di-graphs, tri-graphs, and other longer n-graphs, as opposed to using only tri-graphs in [73]. Forty people participated in this study providing fifteen typing samples each; over a period of six months. Both A and R measures have to be a minimum value to give good estimation that the two typing samples are fairly similar, therefore, originated by the same person. The authors experimented with several combinations of A and R measures using various n-graphs. The best result they found was an FRR of 5% and an FAR of 0.005%. Even though these results were very good, the computational cost required to identify users was expensive because it was necessary to compare the test sample with all users' templates in the database, which clearly makes it less scalable.

Hu et al. [115] attempted to solve the scalability issue of Gunetti and Picardi's method [8] using the K-nearest Neighbor classifier. In this approach, training samples were divided into clusters such that, every test sample was compared only with the samples of those users in the same cluster. Results from this modification revealed accuracy which compared well with that of [8]. Computation speed, on the other hand, proved to be 66.7% better.

In addition, Davoudi and Kabir carried-out a number of modifications on the method introduced by Gunetti and Picardi [8]. In [116] Davoudi and Kabir combined the R and A measures with a distance-calculating method that used histogram-based density estimation to find the probability density function of each di-graph's duration time. This modification

resulted in five false rejections and nine false acceptances. Moreover, Davoudi and Kabir modified the relative distance, in [117], by choosing the di-graph with the highest difference in duration between the two samples to compute the difference of its positions first. After that, it was removed from the two timing vectors, and then, the new vectors were sorted again. This resulted in 0.08% FAR and 18.8% FRR. Davoudi and Kabir also applied one further modification to Gunetti and Picardi's method, in [118], by adding a weight factor to the digraphs when computing the relative distance. This weight was defined as the ratio of the number of occurrences of this di-graph and its standard deviation. The study resulted in 0.07% FAR and 15.2% FRR. All of the modifications mentioned were applied to a subsection of the data extracted by Bergadano et al. in [8]; this includes 21 participants producing 15 samples each.

Continuous authentication was investigated by Bours and Barghouthi [77]; they used a penalty-reward function for long free-text keystroke authentication. They used the duration of a single key and the latency of two successive keys as the timing features for their experiment. In particular, only the durations and the latencies of keys which occurred more than 50 times and which mean and standard deviation were under a predefined value were added to the user's templates. The penalty-reward function in this research had a zero start-up value and it increased if the distance between the test sample and the user's template was larger than a threshold and decreased otherwise. If the value of this function gets higher than another threshold, the user was denied further access to the system. This experiment used data from 25 volunteers, which was collected over at least six days. At the end of the experiment, the average number of keystrokes typed before an imposter was locked out was found to be between 79 and 348 keystrokes.

Most of the previous work has not fully benefited from the keystroke dynamics concepts, in the sense that they did not consider key-pairs. Therefore, Park et al. [69] divided all keystrokes to four features; left hand side, right hand side, spacebar and backspace bar. Then, they created di-graphs using feature combinations. This resulted in sixteen key-pair features, e.g.: left hand side key & space bar, left hand side key & backspace bar etc. Comparing the two samples was performed using these key-pairs. Only key-pairs with more than ten appearances were used in the comparison process. The Kolmogorov-Smirnov test (KS-test) was then used to compare samples. Thirty-five users participated in this study, in which they were requested to type two page length news articles. An EER of 0.0892% proved that this method had indeed increased the performance of keystroke dynamics authentication.

Similar to Park et al. [69], a key grouping technique was introduced by Sing and Arya [50]. Key grouping was performed by classifying the keys based on their location on the keyboard. The keyboard was divided into 8 sections; two left and right halves and then each half was divided into 4 lines representing the rows of the keyboard. For example “wm” is represented as Left2-Right4. Flight times between these key-pairs were utilized. Euclidean distance was applied to calculate the difference between the training and testing vectors. A threshold was defined to decide if the test sample had originated from the authorised user, however, there were no details about its value or how it was chosen. Data used in this experiment was collected from 20 objects, each of which performed 5 login trails, as a legitimate user, and 5 login trails, as an impostor. The overall performance reached 4.0% FRR and 2.0% FAR.

In addition to the standard duration, the latency of the 20 most frequent di-graphs in English and the total duration time of the 20 most frequent words in English were also used as keystroke features in [119]. SVMs, k-nearest neighbour, Naive Bayes classifier were all used to classify the data collected from 28 individuals. For building the user profile, the first 25 repetitions of each feature were captured, 20 of which were used for training and the remaining 5 for testing. The SVMs succeeded to achieve the best accuracy reaching 90%. Moreover, duration time was found to produce the best performance followed by the word total duration. Using only the frequent words might not have been the best choice, as most of the frequent words used in this study are very small words such as: “the”, “is”, “it”. It would have been interesting to compare the results from such words with the results from other longer words.

Moreover, a list presenting brief details about some of the work done in free-text keystroke dynamics authentication is shown in table 2.5.

Table 2.5: A list of free-text keystroke dynamics studies.

Study	Features	Method	Subjects	Samples	Performance
Gaines et al. [39]	Latency	T-test	6	36	0.00% FAR, 4% FRR
Umphress and Williams [34]	Latency	Standard deviation, t-test	17	34	6% FAR, 12% FRR
Monrose & Rubin [33]	Latency, duration	Euclidian distance, probability score, weighted di-graph probability	31	-	23% accuracy
Gunetti & Ruffo [71]	Latency, executed commands	Decision tree	10	-	90% accuracy
Dowland et al. [78]	Latency	Mean, Standard deviation	4	-	50% accuracy
Gunetti & Picardi [8]	N-graph duration	Relative distance, absolute distance	205	765	0.005% FAR, 5% FRR
Gunetti et al. [88]	N-graph duration	Relative distance	30	124	1.67%FAR, 11.67% FRR
Villani et al. [49]	Latency, duration, typing speed, percentage of special characters, editing patterns	Euclidian distance, k-nearest neighbour	118	2360	99.8% - 44.2 % accuracy
Curtin et al. [79]	Latency, duration, typing speed, percentage of special characters, editing patterns	Euclidian distance, k-nearest neighbour	30	90	100% - 97% accuracy
Filho & Freire [89]	Latency	Simplified Markov chain model	15	150	41.6% - 12.7% EER
Janakiraman & Sim [120]	Latency, duration	Bhattacharyya distance	22	-	100% - 70% accuracy
Buch et al. [121]	Latency, duration, percentage of special characters	Euclidian distance	36	650	100% - 98% accuracy
Hu et al. [115]	N-graph duration	Relative distance, absolute distance, k-nearest neighbour	36	36554	0.045% FAR, 0.005% FRR
Hempstalk et al. [80]	typing speed, error rate, press-release ordering	One-class classification	10	150	11.3% FAR, 20.4% FRR
Ahmed et al. [91]	Latency	Neural network	22	-	0.015% FAR, 4.82% FRR
Davoudi & Kabir [116]	N-graph duration	Relative distance, absolute distance, histogram-based density estimation	21	315	0.015% FAR, 0.0025% FRR
Pilsung et al. [122]	Latency	Kolmogorov-smirnov Test	-	-	0.17% EER
Samura & Nishimura [76]	Latency, duration	Weighted Euclidian distance	112	-	67.5% - 81.2% accuracy
Bours & Barghouthi, [77]	Latency, duration	Distance measure	25	-	79 – 348 keystrokes
Davoudi & Kabir [117]	N-graph duration	Modified relative distance	21	315	0.08% FAR, 18.8% FRR
Davoudi & Kabir [118]	N-graph duration	Weighted relative distance	21	315	0.07% FAR, 15.2% FRR
Park et al. [69]	Latency	Kolmogorov-smirnov Test	35	-	0.089% EER
Messerman et al. [43]	N-graph duration	Normalized relative distance	55	-	2.20% FAR, 1.84% FRR
Sing & Arya [50]	Latency	Euclidian distance	20	-	2.00% FAR, 4.00% FRR
Chantan et al. [27]	Latency	Bayes classifier	-	-	0% EER
Bakelman et al. [68]	Latency	K-nearest neighbour	20	200	4% EER
Bours [44]	Latency, duration	Scaled Manhattan distance	25	1620	182 keystrokes
Monaco et al. [123]	Latency, duration	K-nearest neighbour	30	300	99.96% accuracy
Kang & Cho [124]	Latency	MV test, K–S statistic, C–M criterion, R measures, A measures, Gauss, Parzen, k-NN, SVDD.	35	35	7.87% EER
Matsubara et al. [125]	Latency, duration	Weighted Euclidean distance, Relative distance	250	2500	92% accuracy
Darabseh & Namin [119]	Latency, duration, word total time duration	SVMs, k-nearest neighbour, Naive Bayes classifier	28	700	90% accuracy

Descriptions of latency, duration and n-graph duration are found in Section 2.6. Definitions of accuracy, FAR, FRR, ERR and keystrokes are found in Section 2.7

2.9 Factors Affecting Performance

In addition to the various options of timing features and classification methods, there are many different performance measures used to determine the error rate in keystroke systems; it is therefore often difficult to compare studies in this field [87]. This is also due to not having any form of standardization in the data collection process in these different experiments. Therefore, even though the error rate of study A is lower than that of study B, that does not necessarily mean that the method adapted in A is better than that used in B. Different factors may have positive or negative impact on the authentication process regardless of the actual functionality of the method [126].

Moreover, it has been argued by Killourhy and Maxion [67] that even though performance measuring was done by many researchers for different keystroke authentication systems, it is not fair to compare these performances, due to the fact that there are many aspects that vary between the different conducted experiments. These factors include: classifier, feature set, password length, number of repetitions, number of testing attempts, threshold values, used keyboard etc. Therefore, in order to get an idea which system works best, it was inevitable that a standardized environment had to be established when conducting all experiments [127].

Standardization of such factors will offer an improved comparison mechanism between algorithms [128]. This standardization requires information exchange amongst researchers. Moreover, there are two main techniques for standardizing the factors involved in keystroke dynamics studies. First, using one of the automated programs, available in the market, for collecting the data used for the studies. A broad range of software is available commercially; for example: AdmitOneSecurity and BehavioSec [4]. Another solution involves the use of standardized databases, which has been formerly created and published for the purpose of keystroke dynamics research. A list of some of the databases available online can be found in [128]. Using these solutions could not only standardize the data collection method, it could also decrease the duplication of effort among researchers.

Furthermore, there are a large number of different factors which have proven, by various researches, to affect the performance of the keystroke dynamics systems. These factors have to be considered when attempting to deploy a keystroke system. A detailed list of these different factors is provided in this section.

2.9.1 Environment Controlling

There are two basic categories in which experiments have been conducted in keystroke studies. Experiments have either been conducted in a controlled environment or in an uncontrolled one. In a controlled environment, users are asked to type on a specific machine which has built-in software for recording the keystrokes. Thus, the same external conditions were consistent for all users [87]. The issue with this kind of arrangement is that it may not have the same characteristics as those encountered in realistic situations; therefore, the response may not be representative of a user's typical typing patterns [4].

In uncontrolled environments, on the other hand, users are asked to either download a program, for collecting keystrokes, on their machines [120, 89] or to use an online data collection form [8, 115]. This indicates that the data is collected wherever and whenever it is convenient to the user. Although, this method provides a realistic representation of the normal conditions for the user, each user's surroundings can be very different, which makes the data harder to analyse. It might also be the reason for inconsistencies in the keystroke data provided by the users. Although some of the experiments in the literature have been conducted in a controlled environment [99, 103], the majority of research has, so far, been conducted in uncontrolled environments [8, 116, 27, 54], due to the authors' desire to imitate the life-like conditions of a real authentication system.

2.9.2 Keyboard Type

Using different computer brands has an impact on the user's typing pattern since the keyboard of different brands varies in key size and spacing between keys. This is clearly a reason for users to type differently on different keyboards [4]. Furthermore, different keyboards have different key pressing sensitivity levels which consequently may affect the timing data collected from the users [129]. Using a laptop keyboard adds another variation which can also affect the typing behaviour; as laptops provide the freedom of movement, users may use it in different positions, such as on a bed or on a table [42].

Villani et al. [49] investigated the case of using different keyboards in free-text keystroke systems. One of their experiments was conducted using a desktop keyboard and another was performed on a laptop keyboard. A significant finding was uncovered, which can be summarized as: the system has a good chance of accurately identifying a user as long as he or she uses the same type of keyboard for training and testing. It is therefore important that

researchers keep using the same keyboard in order to maintain the same level of typing consistency throughout the data collecting process [79]. In contrast, Giot et al. [6], conducted a study which led to proving that although using two different keyboards for training and testing may slightly reduce the system performance in the case of fixed-text keystroke dynamics, it did not have a significant negative influence on the system.

2.9.3 Entry Mode

Because free-text keystroke systems are used for long text, it makes more sense to allow the users to enter whatever text they prefer. Having said that, free-text authentication studies have actually used two different methods for text entry, i.e. way of input. The first technique allowed the users to type completely free text as they desired, such as: typing an e-mail or a report for work or an essay for school [8, 75]. The second approach required the users to type a specific long text from an article, in which the users needed to copy specific text into a section specified for text entry [79, 115].

In the research conducted in [49], participants were asked to be a part of several experiments with different conditions. One of these tests incorporated a copy-task in which the participants were asked to copy a predefined long text. Another included a free-text input where users were free to type arbitrary text. It was found that the accuracy of correctly authenticating a user decreased considerably when using different input modes in the training and testing phases. Moreover, it was also shown that the accuracy in free-text typing mode was slightly higher than that in the copying task mode. This can be explained by the frequent pauses that a user has to perform in order to look at the text during the copy-task which might cause the collected data to be inconsistent.

In contrast, the experiments carried-out in [130] produced results that showed that using either free or copied text has no effect on the results of free-text keystroke systems. This conclusion was reached after analysing two keystroke systems, one involving free-text samples while the other included text that participants were asked to copy. The insignificant difference between the two entry modes is maybe due to the fact that the timing differences between free and copied typing are small in comparison with the timing differences between two different typists.

Another experiment was conducted for fixed-text keystroke dynamics in [90] to investigate the relationship between the data acquisition method and the recognition performance.

Different scenarios were, also, followed to acquire data for this study. These scenarios differ in the way that the pre-defined text was presented to the user. It was either displayed on the graphical user interface or the pronunciation of the words was orally provided. The conclusions that can be drawn from this study can be summarised as: oral presentation improves the EER for unknown words, while it stays the same for known words. Generally, there is no significant difference found between displaying and telling a user the text when acquiring the keystroke data.

2.9.4 Text Length

One area in which standard keystroke systems lack is the amount of information that can be obtained from raw data [49]. The only data that can be collected while the user is actually typing is the time each key is pressed and released, from which only little information can be inferred, including the time interval between each two consecutive keys and the duration time for each key pressed. In addition, the data is often not stable since it changes based on the environment surrounding the experiment, or based on the state of mind of the user at the time [87]. To reduce the effect of such instabilities, a large amount of research has shown more interest in using short text [27, 33] in fixed-text keystroke systems.

Furthermore, Montalvao et al. [131] carried-out a comparison between three fixed-text databases. They concluded from experimenting with different length passwords that the EER tends to deteriorate as the length grows for passwords. In contrast, the study in [110] examined the application of keystroke dynamics as a part of a multifactor authentication; in which it was used as an extra security measure for card/PIN schemes used in apparatuses such as Automatic Teller Machines (ATM). Results showed that PINs of longer lengths yielded better performance. However, using long passwords, for example, will worsen the password's security-usability problem [28].

In free-text keystroke systems, however, it is not enough to use short texts to analyse keystrokes since it does not offer an adequate amount of information to distinguish between users. Consequently, using longer text samples is considered a better alternative [78, 8, 39, 124]. Furthermore, more typing features can be collected from longer texts, such as shift key usage and editing habits. Moreover, the study in [79] provided evidence that using long-texts increases the stability of the di-graph's mean and standard deviation significantly.

Nonetheless, the problem with using long free-text keystroke systems is that the training phase, unavoidably, needs more time.

The authors of [79] investigated the accuracy of identifying users when typing long-texts in cases where the training and testing texts were different in length. The accuracy from different-text/same-length experiments was better than that from different-text/different-length experiments. Therefore, improving authentication accuracy can be achieved via standardization of the feature measurements, i.e. the text size in this case.

2.9.5 User's Experience

The user's health and state of mind are not the only crucial part of the authentication process using keystroke dynamics. A user's typing skills and level of comfort while using a keyboard are also characteristics that have a clear impact on the user's typing behaviour [76]. The more skilful the user is, the more stable his or her fingers are located on the keyboard and the more familiar he or she is with the position of each character on the keyboard. This will result in a more consistent typing pattern all through.

Samura and Nishimura [76] conducted a study that examined the effect of the user's experience on long, free-texts keystroke systems. Participants were divided into three groups based on their typing speed, specifically the number of letters typed in a 5 minute period. This study indicated that the best recognition accuracy was obtained from the group which typed the fastest, i.e. the more skilful typists.

2.9.6 Monitoring Mode

A free-text keystroke system can be a continual process of identity verification, which takes place during the whole time a user is using the system. This can be done in either a continuous or a periodic manner. Continuous authentication is done, in real time, every time a key is clicked on the keyboard [44]. Although this method provides strong imposter detection, it is computationally expensive.

Periodic authentication, alternatively, is repeated every time a certain text is entered [120]. This is a less strict method, security wise, yet it is computationally cheaper. Moreover, waiting until a specific text is entered may cause the system to wait for long periods of time if this particular text does not occur frequently enough in the typed text; which will represent a

security threat for the system. A periodic verification scheme that included the use of interruptions was utilized in [68]. In this research, the identity of the user was only verified after text breaks, e.g. user leaving the PC for a coffee break. The system only captured the first burst of input after each pause. This method reduced the frequency of authentication checks, which was a key reason for decreasing the false alarm rate in addition to decreasing the computational cost.

2.9.7 Text Choice

It is clear that choosing a specific piece of text is crucial for training and testing the system. It might be assumed that using familiar English words may produce more consistent typing patterns. However this was proven wrong by Janakiraman and Sim [120]. In this research, a new “goodness” measure was suggested to calculate the universality, accuracy and expectancy of a word used for periodic free-text keystroke authentication. Universality is a measure to identify if a word is one of the words commonly used by users. Accuracy measures how unique a word is. Lastly, expectance is used to calculate the average number of keystrokes typed before that word actually appears in the text. Unexpectedly, using the goodness measure, non-English words, such as: ‘tmr’ which is an abbreviation of ‘tomorrow’ used in online chats, are proven to be better than English words for identification and verification purposes. Similarly, both studies [109, 101] reached similar outcomes, in which normal "English-like" text was discovered to be less capable of discrimination between users.

Moreover, Mondal et al. [132] focused on the relation between the password complexity and the system performance. They produced results that showed a reduction in system performance when passwords increase in complexity. Thus, although random passwords make password guessing by attackers harder, it might not be appropriate for keystroke dynamics systems.

Another aspect that was considered in the literature was the familiarity aspect of the pre-defined text. This aimed to find how the expertise of a user with a certain text affected the system performance. Giot et al. revealed in [6] that there was no noteworthy accuracy difference between an imposed login/password and a login/password that users chose themselves. Alternatively, it was observed that long and familiar strings produces better performance than short and random strings in the experiment conducted by Stefan and Yao in [70]. Similarly, Joyce and Gupta [36] and Garcia [53] both stated that using well-known text

such as the user's name, provided better authentication accuracy, compared to un-familiar text. Even though there is a split of opinion in the available literature, it is clear that the authentication system performance differs depending on the type of strings used.

2.9.8 Number of Training Samples

When considering the training phase in fixed-text keystroke systems, it is hard to ignore the time required for training the system by re-typing the password again and again. This is needed for the users to develop a stable rhythmic signature corresponding to their password [131]. For instance, in [104] participants were requested to type a 10-digit passcode 50 times throughout four sessions, which were held over four different days. This adds up to 200 samples per person; half of which were used as training data and the rest were used as test data.

The same issue arises for free-text keystroke systems where the text collected for training is longer than that collected for fixed-text. In some cases, free-text data can be collected while the user is performing daily typing tasks. For example, 15 samples were collected from the participants over a two weeks period in [68]; each sample was 400 characters long of whatever the user needed to type at the time.

From the results of the experiment conducted in [88], it was found that the accuracy of the system generally escalated when the number of samples in the user's profile increased. In fact, the study in [6] had proved that less than 10 samples per user were not sufficient enough for fixed-text authentication systems, while exceeding 50 samples can also deteriorate the system performance. Moreover, around 40 samples yielded the best results. Meanwhile, an effective mechanism for free-text profile enhancement was suggested in [50] where the user's profile was expanded, during the typing session, by adding new timing data attained from the user after being authenticated. This was done to increase the number of samples in the users' profiles.

2.9.9 Number of Participants

Good performance resulting from a particular system cannot be generalized unless it was proven to distinguish between a reasonable number of users [4]. Some methods work perfectly in smaller populations but fail to scale-up when applied to a realistic number of users. For example, Ke et al. [133] suggested a system that produced a good performance

achieving 3% FRR and 2% FAR; but they used data from only 4 subjects to conduct their study, which is regarded as very small.

Lau et al. [45] intended to make sure that the statistical model proposed by Ke et al. was scalable to a larger number of users. Therefore, they implemented a similar model to that introduced by Ke et al. in [133]. A larger sample size was used in this study; 15 participants in total who produced 10 samples each. The error rates produced by this test were 12% FFR and 10% FAR, which is considerably higher than that of the experiment conducted by Ke et al. [133]. That means that even though this method produced good results, it was not applicable for real-life usage.

2.9.10 Threshold Choice

Many of the schemes used in keystroke dynamics systems rely on a specific value, or threshold, used to judge if the typing sample in-hand was originated by the authorized person. Furthermore, thresholds are used in many studies to act as a barrier between genuine data and impostor data; Figure 2.6 demonstrates the segregation of data performed by a threshold. An imposter's data is mistakenly considered as genuine if it falls below the threshold and, in the same manner, genuine data is incorrectly considered as an imposter if it falls above the threshold.

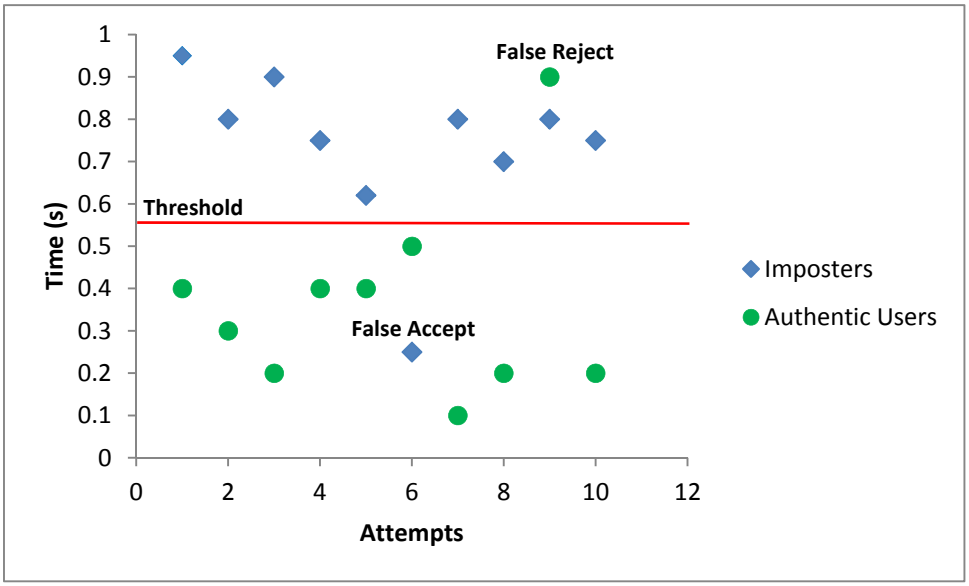

Figure 2.6: Discriminating data using a threshold.

There are two main types of thresholds: namely: global and local. A global threshold is a single value used for all users [134], while a local threshold is different for each user depending on his or her typing behaviour [50]. Both the work done in [6, 135] agreed that using an individual threshold yielded better results compared with a global threshold. On the contrary, the study in [136] made use of both global and local thresholds. The global threshold was set to a constant value for all users; the local threshold was obtained by using the average score of each user's training set and an experimental constant factor. The overall performance of the study using the global threshold was 8% EER while using the local threshold was 8.22% EER. This suggested that using a local threshold does not affect the over-all performance of the system significantly.

2.9.11 Pre-processing and Post-processing

Data collection in free-text keystroke systems involves the collection of a large amount of text, gathered over a period of time. From all the typing data collected during this time, the system infers the typing pattern that the user typically follows, which will be then stored as the user's profile. It is, however, not necessary to include all letters and letter combinations typed during the enrolment phase in the user's template, i.e. there are conditions for including a particular letter or combination of letters in the template. The character or combination of characters have to be typed often enough, during the enrolment phase, to ensure that its mean and standard deviation are statistically sound [44].

Therefore, large amounts of research include a pre-processing stage for removing noise from the data set. Extreme duration or latency values, i.e. very small or very large outliers, were discarded. For example: only the durations and the latencies of keys for which the standard deviation was below a predefined value were added to the user's template in [33, 77], while minimum and maximum values were fixed for the latencies that were used in [78]. Moreover, there were a large number of features to be considered in [70], therefore, Principle Component Analysis (PCA) was used to minimise the dimensions of the feature vectors before attempting to classify the data. Similarly, PCA, Multidimensional scaling (MDS), and probabilistic PCA were used for dimensional reduction in [137] while Linear Discriminant Analysis (LDA) and PCA were used for pre-processing the data used in [138].

In addition to pre-processing, post-processing was often used to enhance the quality of the authentication system. The adaptation process is a post-processing step where data is

collected during the authentication stage to update the users' existing templates. For example, the method used in [6] resulted in around 10.30% EER; using an adaptation mechanism improved the system performance to achieve 6.96% EER. An intelligent adaptation scheme was followed, which added new timing vectors to the users' profile, during log-in time, if the total number of samples was less than 15; and if not, the new vector was swapped with one of the old vectors in the profile. The new vector had to be relatively similar to the ones already stored in the profile in order to avoid distortion of the data in the profile. Additionally, the work in [13, 43, 66, 125, 139] showed that the adaptation mechanism has proved to decrease the error rates since it had the ability to automatically adapt to changes that the user's typing patterns went through, by frequently updating the data in the users' profile.

The 75 timing features selected from the key-pairing scheme, defined later in this research, is chosen to capture the mechanical patterns performed by a user through the time elapses captured when he or she is typing. In addition, the nine non-conventional features, described later, are chosen to capture the user's typing and editing patterns while typing a whole piece of text. These two feature sets are employed in the process of user recognition due to the unique characteristics they extract from the users typing stream which can be used for differentiating between users. In the course of this research, the reasons taken into consideration when selecting these timing features and non-conventional features are further explained in Chapter 3 and Chapter 4.

2.10 Applications

Although more than a third of a century has passed since keystroke authentication was first introduced, it has not yet been applied in the security field to a high degree. Fixed-text keystrokes has been used to add protection for PCs at log-in. Free-text keystrokes, on the other hand, has provided security to the user by locking-up the workstation when an imposter is detected at any point of time, during which the system is in use. In addition to that, a wide variety of other applications can also benefit from the keystrokes scheme. Many of the keystroke dynamics applications are listed in this section.

2.10.1 Identification and Authentication

Keystroke dynamics systems are used mainly for two different purposes. Firstly: identification, which is a way of determining the user's identity when no data is available

about their identity beforehand [7]. In this method, a test sample is matched with all the templates stored in a database. The system assigns the user to the identity of the person whose template is the most similar to the test sample. Authentication, alternatively, is used to verify the identity of the user [1]. The user supplies his or her identity and the system takes-on the responsibility of making sure that the user is who he or she claims to be. The test sample, in this case, is only compared with the user's template in the database.

The complexity of performing identification is clearly higher than that of authentication since it includes comparing the test sample with all available templates, which may be a very large undertaking in large-scale systems. From the definition of both methods, a fixed-text keystrokes system is used mostly for authentication as the system employs a ID/password that is considered a mean for providing the user's identity [102], while it can be used for identification in the case of a shared password [52]. Nevertheless, fixed-text seems more suitable for authentication rather than identification [140]. A free-text keystrokes system, on the other hand, is used for both identification and authentication, as it makes use of long, random text [8].

2.10.2 Different Language Authentication

Most of the work done on keystroke dynamics has concentrated on using the same language for training and testing the system. In fact, the vast majority of studies included only English input. Gunetti et al. [88], however, gave empirical proof that free-text typing patterns could be used to authenticate the user, even when the test samples were written in different languages to that of the samples in the user's profile. Evidently, this only works when the two languages share a significant number of di-graphs. So, languages like English and Italian, which have the same alphabet, can be used for this kind of authentication, but English and Arabic, for example, cannot be used because they have a completely different set of letters.

The data used in this study was provided by Italian speakers, each of whom provided two samples typed in Italian and another two samples typed in English. From experimental results, about 10% mistakes in identification occurred when the test sample was in a language different to that of the user's profile. Better performance was obtained when the user's profile contained samples in both languages. The error rate was even smaller when the test sample was in the same language as that which dominated the samples in the user's profile.

By experimenting with different combinations of template samples and test samples, it was clear that those samples provided by the same person when typed in different languages were more similar than those samples provided by different persons which were typed in the same language. An average performance of 1.61% FRR and 3.23% FAR was achieved in total. Thus, although keystroke authentication for different language texts is possible, it is more difficult than the case where all samples are in the same language.

2.10.3 Old Profile Authentication

Most of the studies conducted in the free-text keystroke authentication field have had only a few months gap between the time the training samples were collected and the time in which the test samples were gathered. Gunetti et al., however, showed in [141] that a typing profile could still be used to identify a user, even though, it has been created a long time before the test samples are provided.

Gunetti et al.'s original experiment involved 30 participants whom were asked to provide 15 samples each. The samples consisted of whatever the users chose to type. One and a half years later, the same 30 volunteers were asked again to provide another two free-text samples. It was discovered that even after such a long period of time, the keystroke dynamics system was still able to identify users with an average accuracy of 1.67% FAR and 11.67% FRR.

2.10.4 Intrusion Detection

The continual authentication scheme that the free-text method can provide is a very effective intrusion detection method [44]. It is mainly used to notice any warnings with regards to irregularities in the typing patterns of a specific user. Moreover, free-text keystroke systems are used for active monitoring of the system which can aid in finding any intrusion quickly and reliably.

One important issue that has to be addressed here is the generation of false alarms in such systems. This might cause frequent and rapid system halts, which cause a great deal of annoyance for the users when they falsely occur. Therefore, Gunetti et al. [88] suggested using it combined with other authentication methods in order to reduce the false alarms, or FRR.

2.10.5 Online Marketing

Free-text keystroke systems can also be utilized for identifying users over the internet. This is done by capturing the users' typing patterns on their first visit to the website, and then it can be used to identify returning users [88].

This data can be used to determine user preferences and interests which can be employed for marketing purposes. This approach, on the other hand, has many privacy issues regarding the amount of information that users are happy to hand-in to the websites they visit. It also can cause the user to experience large amounts of un-wanted marketing advertisements.

2.10.6 Cybercrime Investigating

Tracking users through their typing patterns can also help in cybercrime and investigating illegal electronic movements of anonymous users. Using free-text keystroke schemes was suggested for network forensics in [91], through attacker profiling, which is conducted by collecting the suspect's typing patterns when he is surfing websites on the internet. This profile, collected for each user, can be used as a digital fingerprint gathered from the cybercrime scenes.

This is considered as passive fingerprinting because it can be created without the knowledge of the attacker, which can be extremely beneficial in fraud or identity theft cases [142], in which attackers are completely oblivious that they are being monitored. The issue with such a digital fingerprint is that it must be built progressively, which requires a large number of internet service providers to collaborate and work together to face such threats.

2.10.7 Emotion Detection

Free-text keystroke systems gather great amounts of data from the user during the whole time he or she is using the computer. This data can also be used to infer the emotional state that the user is going through during the typing process. This has been employed in [143] to determine what the user is feeling during every day free typing. Emotions such as: frustration, focus, anger, stress, relaxation, excitement and tiredness were derived from the user's typing behaviour.

A similar study was performed in [144] to identify the level of boredom and engagement during typing in educational systems. Keystroke features were applied separately or coupled with stable traits and/or task appraisals in this research.

Extracting the emotional state that the user is going through during a particular period when the user is using the system has many benefits for intelligent computers. It helps the system make the right decisions regarding the best interaction method to follow with the user [143]. This can also aid in detecting the fatigue level of the employees as suggested by [33], which helps to increase the productivity of the organisation.

Moreover, levels of stress were successfully recognised for the individuals involved in the study published in [145]. The detection of harmful levels of stress can be an easy solution for users to manage their health issues. In this study, typing samples were gathered under stress and no-stress conditions. In addition to keystroke features, linguistic features were also extracted from the samples of each user.

The issue with using keystrokes for user emotion detection is that it can cause an invasion into the user's typing experience. For example, in [143], the user was required to determine his or her emotion every 10 minutes, in order to train the system to identify his or her emotions automatically.

2.10.8 Soft-Biometrics Identification

Keystroke dynamics can be used to infer some of the "soft biometric" information [146] of the typist. Soft biometrics are characteristics not unique to an individual but are directly related to him or her, such as: gender, age, dominant hand, native language ... etc.

Keystroke dynamics was deployed for providing information to the message recipients about the message sender's gender in the social network environment. This was done in the work conducted by Fairhurst and Costa-Abreu [147] in which they used data from 98 male and 35 female participants. All individuals were asked, over a period of three months, to type the password "greyc laboratory" between 5 and 107 times. K-nearest neighbours, decision trees and Naive Bayesian were used for classification. Additionally, fusion techniques were also used to combine the three classification methods; the fusion techniques used were: dynamic classifier selection, majority voting and Sum. Individual classifiers achieved the best

accuracy of 79%. Fused classifiers succeeded to predict the gender with a higher degree of accuracy with the best reaching 97%.

Another study that involved gender recognition using the users' typing pattern was published in [148]. The data set used in this study was the same as the one used by the authors of [147]. Moreover, SVMs were used to achieve gender recognition with 91.63% accuracy.

The research in [149] investigated soft biometrics identification for both fixed-text and free-text keystrokes. Gender, age and dominant hand were the focus of this study. In the fixed-text experiment, the recognition rates of the three soft biometrics were: gender between 70% - 86%, age between 67% - 78% and dominant hand between 78% - 88%. In the free-text experiment, the recognition rates were slightly better: gender between 79% - 84%, age between 72% - 75%, dominant hand between 83% - 88%.

Moreover, keystroke dynamics features were used together with stylometric features and language production features to identify some of the users' demographic attributes. The study carried-out in [150], aimed to recognise the user's age, dominant hand and native language. In this study, free-text sentences were used to achieve the following results: for 55% of the text, all 3 demographics were correctly predicted and for 95% of the text at least 2 of the 3 demographics were correctly predicted.

2.10.9 Key-pad Devices

Devices that include key-pads for user input such as mobile phones, automated teller machines (ATMs), point of sales (POSs) and building access control systems have been the focus of many keystroke dynamics research. The users of these devices don't normally type as often on their devices as they do on computers, which has caused the recognition of unique typing to be more challenging in these devices [151]. Therefore, keystroke dynamics authentication is frequently done on these devices using fixed-text that is used to unlock the device when beginning to use it.

An example of that can be found in the work done in [152] in which a 4-digit numeric PIN was used. In this experiment, 25 users provided five training samples and 30 testing samples. All samples were provided using the SAMSUNG SCH-V740 phone. An average of around 14% EER was achieved.

A similar research was performed on 25 Nokia smart phone users [46] in which keystroke data was collected over a 7 day period. In addition to the standard duration and latency, the authors made use of another four features: they are: error rate, horizontal digraph, vertical digraph, non-adjacent horizontal digraph and non-adjacent vertical digraph. An FAR of 29.2% and FRR of 30.8% were achieved in the end of the experiment.

Moreover, a keystroke dynamics authentication system was applied on a modified ATM keypad in [153]. In that research, both the applied force and the duration and latency timings were used as the keystroke features. A total of 30 subjects were asked to enter a PIN eleven times to be used in the experiment resulting in a 10% EER.

2.10.10 Touchscreen Technology

The introduction of advanced and sensitive multi-touch screen devices such as touchscreen phones and Personal Digital Assistants (PDAs) has improved the abilities of the hardware used for keystroke dynamics research. This development accounts for more accurate measurements of existing timing features in addition to the introduction of new measurements that have the potential of being deployed in keystroke dynamics systems [48].

A notebook touch PDA was used to capture users' typing rhythms in [64]. Duration and latency times were used along with figure pressure as features for the authentication system. A total number of 10 participants were asked to enter a 10-digits-long password 30 times. Probabilistic neural network (PNN) was employed to carry-out the classification stage of this study. Finger pressure produces higher accuracy, compared with duration and latency times, reaching the EER value of 1%.

Moreover, another study published in [154] involved 152 subjects typing a 17-characters-long passphrase for ten times on a Samsung Galaxy Nexus keyboard. A K-Means classifier was used to classify users based on their n-graph (di-graph and tri-graph) times. FAR and FRR resulting from this study were: 4.59% and 4.19%, respectively.

A different study used Android-based smart phones to collect data from 42 individuals [155]. This data consisted of the password ".tie5Roanl" which was typed 30 times by each user over two sessions. Features used in this study were: hold time, latency time, key-hold pressure and finger area. Naive Bayes, K-nearest neighbour, SVMs, random forest and multilayer perceptron were used for classification. The best achieved accuracy was 93.26%.

More recent research was published in [156], which involved a simulation of an Android pattern-lock layout. In such a layout, the keystroke features: DU, DD, UD, and UU are extracted in a slightly different way. In the graphical keystrokes, such as the pattern-lock layout, these features are formed through the press, move-in, move-out, and release events. Furthermore, an authentic pattern-lock should be completed in one draw. Therefore, the pattern-lock construction process consists of only one press and release event. In addition to the four timing features, three original features were used: pressure for move-in, size for move-in and angle for move-in. A total of 113 participants provided 20 samples each to produce empirical results, of which pressure and angle offers the best EER reaching 3.03%.

Even though touchscreen is the future for most devices, the study conducted in [124] produced proof that the usage of the PC's keyboard conveyed the best authentication accuracy, followed by soft keyboard and touch keyboard.

2.11 Security Issues

This section discusses the security level that keystroke authentication systems provide. This is certainly crucial when it comes to actually applying such a scheme to highly secured systems. Moreover, some research in the literature was completely dedicated to test the tolerance of keystroke systems against a variety of attacks; such as the work done in [70, 157]. For example, in the study conducted in [70], a series of tests were undertaken to evaluate the performance of the keystroke dynamics system against human and bots attacks. Two bot simulations, i.e. GaussianBot and NoiseBot, were programed to attempt passing the keystroke authentication system by imitating a specific user's keystroke patterns. They worked by inserting statistically-generated keystroke sequences on the victim's computer.

The keystroke data was collected from 20 users, typing short strings. This was used in three sets of experiments. The first experiment used the data collected from each user to attack other users' profiles. The second and third experiments used the two simulated bots to attack the system. In both experiments, the bots were given access to the keystroke data of all 19 users except the original user's data. The average FAR of the human attacker experiment was 4.2% while the average FAR for the bots attacker experiments was 1.5%. This shows that even though human attackers can slightly exploit the system, it was still adequately protected against automated attacks.

Moreover, a list of the most common security threats is provided below along with the degree of safety that keystroke systems deliver against these dangers [18].

1. Shoulder surfing and user mimicking: is an attack in which the attacker monitors the victim typing, in order to try imitating his or her typing behaviour [1]. Even though there is little possibility of an attacker successfully mimicking a user's typing pattern in fixed-text keystrokes [70], it is even harder to do so in free-text systems. Since it requires the attacker to observe the user's behaviour for a long time, it is very rare that an attacker can actually imitate all the aspects of the user's typing behaviour.

In fact, the study published in [66] investigated this issue and concluded that it is still unlikely to authenticate an imposter even if he or she observed the way the legitimate user types the password.

2. Spyware and replay attack: spyware is software downloaded into the victims' computer without their consent, to record information about them [1]. Spyware is perhaps the biggest threat to keystroke dynamic authentication systems [4] because it can record exactly the time each key is pressed and released. This can be used by the attacker to simulate the legitimate user's typing behaviour and initiate a replay attack such as the one used in [158].

This attack was carried-out through "snooping" the victim's keystrokes, i.e. stealing around 20 to 1200 of his or her keystrokes using a key-logger. Then, another typing sample was forged, using the stolen data, by creating a new sample that contains only di-graphs similar to that existing in the stolen sample. Lastly, the foraged text was injected into the system to make it appear as if the victim was the person using the system. Data from 150 subjects was exploited using four state-of-the-art continuous authentication methods. This attack succeeded in raising the average EER between 69.33% and 2730.55%.

3. Social engineering: is manipulating the user in order to obtain his or her private information [14]. Tricking the victim to reveal his or her typing pattern is not possible using telephone calls or face to face meetings.

Yet, phishing emails [1] can be used to trick the user to type some text which can be used to extract the victim's typing patterns. But even then, the attacker has to get hold of a sufficient amount of keystrokes to be able to actually simulate the victim's typing patterns.

4. Guessing: is trying to guess the way that a victim types. There are simply too many different ways that a user might normally follow when typing. Therefore, guessing the typing behaviour of another person is almost impossible especially for free-text keystroke dynamics [4].

Even if that was possible, it will take a considerably long time to extensively try all conceivable patterns. This exponentially hardens the task of keystroke guessing without any additional knowledge about the user's typing.

2.12 Summary

Keystroke dynamics is a behavioural biometrics solution for identifying a user based on his or her distinctive typing patterns. This is done using the time taken to press a single key and/or the interval-time when pressing two keys sequentially.

Utilizing users' typing behaviour for authentication is performed using either fixed-text or free-text schemes, each of which has advantages and drawbacks. Therefore, the application requirements govern the type of keystroke dynamics system that should be applied.

One concern about keystroke systems is that it tends to be instable, in the sense that it might be influenced by the user's state or by experimental conditions. Nevertheless, keystroke dynamics still has potential in the area of identity verification. This is because it's a low-cost and non-intrusive alternative to the traditional ID/password authentication technology.

Keystroke dynamics offer many security functions, such as providing a more secure ID/password scheme and delivering continuous protection to a PC or an online service. It can also be used in many other real life situations as the same keystroke concept can be also applied to other kinds of keyboards such as virtual keyboards and other keypad operated machines.

From the literature covered in this chapter, a number of gaps have been identified. These gaps have influenced the progress of this research. For instance: free-text keystrokes, although being more applicable in real-life situations, are not examined thoroughly in the current literature (compared with fixed-text). This has driven this research to concentrate on applying free-text in keystroke systems.

Moreover, exploring a wide range of features (including conventional and non-conventional) is of high importance in this research. This is because each one of these features represents the individuality of the user's typing behaviour in a different way. Moreover, feature selection is a vocal point with which the authentication system can make use of the features that have the most influence on the system performance.

In addition, choosing a classifier that best represents the problem in hand is crucial. As found in the literature, the method chosen for classification has a great impact on the system performance. Moreover, testing the proposed technique using more than one classification methods is preferable as well.

It should be noted, as well, that most of the previously reported studies involved substantial input from the user in order to provide the computer system the chance to learn the user's keystroke characteristics. Whilst such experimentation can be acceptable in the laboratory, this extensive initial input can in reality be very off putting for the user, in a practical system. It is more the practical, real-world situation that is focused on here as the least amount of training data is used to infer the user's identity.

It also should be remarked that the vast majority of the studies conducted in this area, only accepts English input from the user. Whilst such experimentation is very important, there clearly is a lack of language variation used in such systems. Therefore, the inclusion of input from another language whose characteristics are completely different to that of English, i.e. Arabic, is a main interest of this research.

Chapter 3

Key-Pairing Method

3.1 Introduction

As mentioned in Section 1.2, keystroke dynamics was chosen for this study to provide an inexpensive, easy alternative to the problematic ID/password scheme [42]. More specifically, free-text keystroke dynamics was found to be closer to real-life situations than fixed-text keystroke dynamics because the identities of participants are verified through natural, everyday typing tasks, and this greatly facilitates ease of use [40]. Nevertheless, keystroke dynamics is hampered by the need to acquire huge amounts of training data [4]. For the purpose of training the system, the users are requested to type large amounts of text in the enrolment session. This research tackles this problem by introducing a novel key-pairing scheme that promotes the most effective use of a minimum amount of training.

This chapter provides an explanation of the original key-pairing method, that has been introduced as part of this PhD research, and then goes on to explain how it has been applied in this study. Some pilot results produced by a very simplified version of this method are also discussed.

Many improvements have been implemented to further enhance the original scheme. In the extended version of the key-pairing technique, more timing features are added in order to capture the significant habits that users follow when typing. The second part of this chapter is dedicated for explaining the modified method. The results of applying the extended method on a group of individuals are presented along with a comparison of the two proposed techniques and comparisons with other state of the art research.

3.2 Original Technique

The original key-pairing technique considers a keyboard-layout based method to compare timing features of free-text typing samples. The method classifies the text to five different key-pairs depending on the position of the two keys on the keyboard. The Euclidian distance

between the timing features vectors of each key-pair is used to find the level of similarity among the samples.

This particular structured key-pair based method for extracting features was explored in this study in order to increase the number of the di-graphs, i.e. combination of two keys, found and compared in both the training and the testing samples. The aim of that is to enhance the stability of the means of each timing feature extracted from these key-pairs which are the main components of the user's timing vector. Thus, the authentication process can be carried-out with the least amount of text possible.

In this approach, an attempt to use the least amount of training possible was carried-out. The main goal for this is the user's comfort and relaxation and hence the realization of a practical and employable system. It is not adequate to relieve users from remembering long passwords if they will still have to type-in a huge amount of text, multiple times when enrolling in the system. Hence, here enrolment is a simple, relatively rapid process.

3.2.1 Feature Definition

The features used in original key-pairing scheme are described in this section. The methods by which key-pair formation and feature extraction were performed are both explained here.

3.2.1.1 Key-pair Formation

This research introduces a novel approach, for free-text keystroke dynamics authentication, which makes use of the keyboard's key-layout. The approach uses the keystroke features extracted between two keys (key-pair) that are pressed consecutively and have a relationship on the keyboard layout. This relationship depends mainly on the key position of each character on the keyboard. The keyboard used in this study was the English QWERTY since it is both the most common keyboard layout and the most popular one [129].

There are five types of key-pairs:

1) *Adjacent:* keys located next to each other on the keyboard.
2) *Second adjacent:* keys that are one key apart from each other.
3) *Third adjacent:* keys that are two keys apart.
4) *Fourth adjacent:* keys that are three keys apart.
5) *None adjacent:* keys that are more than three keys apart.

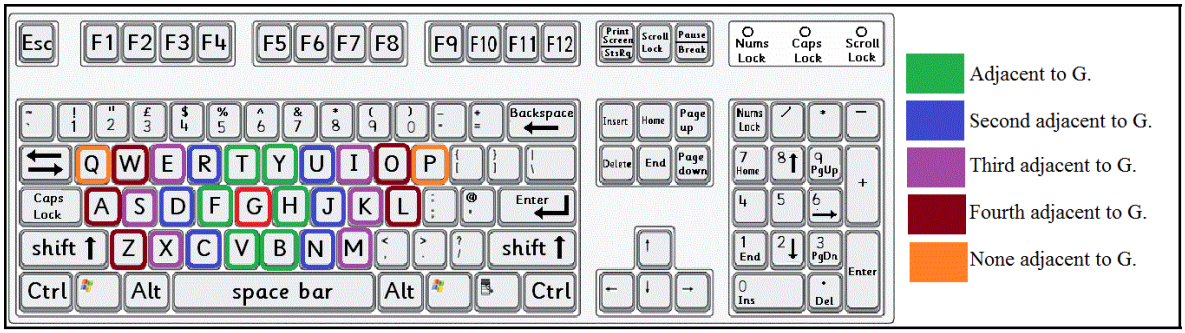

Figure 3.1: Key-pair formation example.

For further explanation, an example is provided in Figure 3.1. If the key G is considered, the related key-pairs are as follows:

- Y, H, B, V, F and T are adjacent keys to G.
- U, J, N, C, D, and R are second adjacent to G.
- I, K, M, X, S, and E are third adjacent to G.
- O, L, Z, A, and W are fourth adjacent to G.
- P and Q are none adjacent to G.

This key classification can be performed in the same way for all the key-pairs in the typed text.

Table 3.1 gives an empirical example for all of the key-pairs in the word "university". This shows the process that is followed to break a text down into di-graphs, or key-pairs, and then classify each di-graph into one of the five types of key-pairs before using its timing data in the corresponding timing vector.

Table 3.1: Key-pairs in the word "university".

Di-graphs	**un**	**ni**	**iv**	**ve**	**er**	**rs**	**si**	**it**	**ty**
Key-pair type	2nd adjacent	2nd adjacent	4th adjacent	3rd adjacent	Adjacent	2nd adjacent	Non Adjacent	3rd adjacent	Adjacent

Given that the key-pairing method significantly boosts the number of key-pairs that can be found and compared in the training and testing samples, it was adapted as a way to increase the soundness of the means of the timing features extracted from these key-pairs. This will help to increase the stability of the timing vectors. This is a clear benefit of the suggested scheme because it enables it to facilitate a small amount of text to compare two samples, i.e.

uses a small amount of typing data in the best possible way. For example, given the following training and testing data:

- Training data: “University of Reading”
- Testing data: “Systems Engineering”

There are only two similar key-pairs (“in” and “ng”) whose typing times can be compared in the authentication process, using regular keystroke schemes such as the one introduced in [8]. However, this is not the case when using the key-pairing method introduced here because there are more instants for each kind of key-pair extracted from both the training and testing data. This is shown in Table 3.2.

Table 3.2: Total number of key-pairs.

Key-pair type	Adjacent	2nd adjacent	3rd adjacent	4th adjacent	Non adjacent
Training data	3	7	2	1	3
Testing data	1	5	2	3	4

Pairs that involve the space key are discarded because of the work conducted by Singh and Arya [50], which provided evidence that a user normally experiences very unusual pauses before and after pressing the space key, which causes inconsistent typing behaviour. Also, typing mistakes have been allowed in this study, as the timing of each key-pair is collected even when it is not correctly spelled. However, key-pairs that include the backspace key are excluded for reasons similar to that of eliminating key-pairs that include spaces [50].

3.2.1.2 Feature Extraction

Once the key-pairs have been obtained from the users’ raw data, the keystroke features are extracted. These features were computed for every key-pair, using four main values: the press time and the release time of the first and the second keys of the key-pair, i.e. D_1, U_1 and D_2, U_2. In this research, five keystroke features were extracted from each key-pair, as shown in Figure 3.2:

- Hold time: each key-pair has two hold times, the hold time for the first key (H_1) and the hold time for the second key (H_2).

- Keystroke latencies: three types of latencies were extracted from each key-pair, down-down (DD) time, up-up (UU) time and up-down (UD) time.

Section 2.6 provides a full description of these timing features.

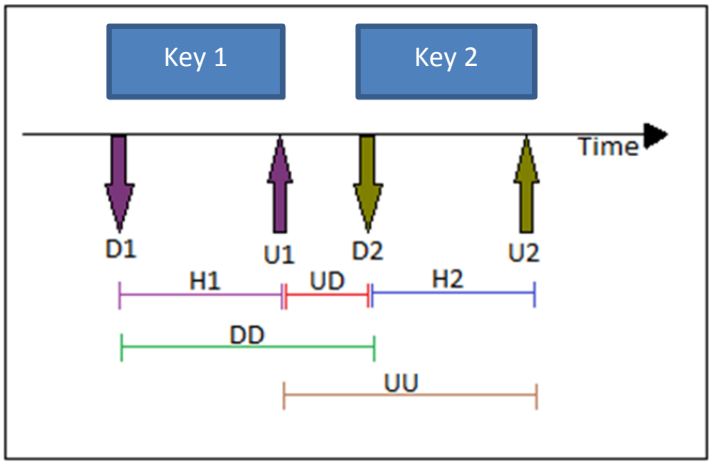

Figure 3.2: Timing features for a key-pair.

In summary, five timing features were defined for each key-pair appearance in the text. This was done for all five types of key-pairs. Therefore, the overall number of timing features is 25 (5 timing features * 5 key-pairs). The means of these features were calculated and stored in the timing vector of each user, i.e. user's template. Table 3.3 lists all the 25 features extracted from all key-pairs. The features abbreviations listed in the table combine the key-pair category and the timing feature, for example: "A-H1" stands for: Adjacent -Hold1 and so on.

Table 3.3: Original approach feature set.

Key-pair Category	Feature Set				
Adjacent	A-H1	A-H2	A-DD	A-UU	A-UD
SecondAdjacent	S-H1	S-H2	S-DD	S-UU	S-UD
ThirdAdjacent	T-H1	T-H2	T-DD	T-UU	T-UD
FourthAdjacent	F-H1	F-H2	F-DD	F-UU	F-UD
NonAdjacent	N-H1	N-H2	N-DD	N-UU	N-UD

3.2.2 Experimental Results and Discussion

In this section, a description of the experiment and the data collection is provided. The data space and the experimental results are also delivered. Moreover, a discussion about the results of this experiment in addition to some comparisons with other studies is produced in this section.

3.2.2.1 Data Collection

For this study, 15 users participated in the experimentation process. Each participant provided merely one training sample and two testing samples. There were participants from both genders and all participants were in the age group between 20 and 60 years. They had different levels of typing skills that varied between moderate and very good. They were not all native speakers of English and included, for example, Greek, Czech and Arabic speakers. Table 3.4 describes some of the demographic characteristics of the participants.

Table 3.4: Characteristics of the participants in the original experiment.

Gender		Age			Native language		Typing skills		
Male	**Female**	**18-37**	**38-57**	**58+**	**English**	**Non-English**	**Good**	**Moderate**	**Poor**
6	9	7	6	2	6	9	10	5	0

During the data collection, or enrolment phase, participants were asked to type a paragraph consisting of five lines of text. The text included five short and well-known English quotes. Most of the words used in the text were short, simple and well-known words. All the words were in lower case, and there were no numbers or punctuation marks.

The 380-character-long text was the only sample used as training for the system; this was considered to be a very short training sample compared to previous studies. This added to the user-friendliness of the system because the only instance of typing that was required as training for the system was of reasonable length. This spared the users from the annoyance of having to spend a long time in the enrolment phase. Users were directed to enter the samples in much the same way that they usually follow when typing. Users were allowed to enter carriage returns and backspaces if needed.

Two shorter testing texts were also collected from the participants, in the same way. These texts were each approximately 75 characters long and were completely different from the text that was used for training.

Volunteers were recruited by the researcher personally approaching them. After receiving the initial approval from prospective participants, they will be provided with the experiment details. All participants were asked to read and sign a consent form that explains the purpose of the research, the nature of the collected data and how the recorded data is treated. This is done in order to fulfil the University of Reading ethical approval process. A copy of the consent form and the information sheet for the ethical approval can be found in Appendix A.1 and Appendix A.2, respectively.

The data collection was performed on a GUI program implemented using Java language. Figure 3.3 illustrates a screenshot of the data collection program. This program was installed in one machine that all participants had to use in order to perform data collection. This is the reason for all the participants being situated in the same location where this research is performed.

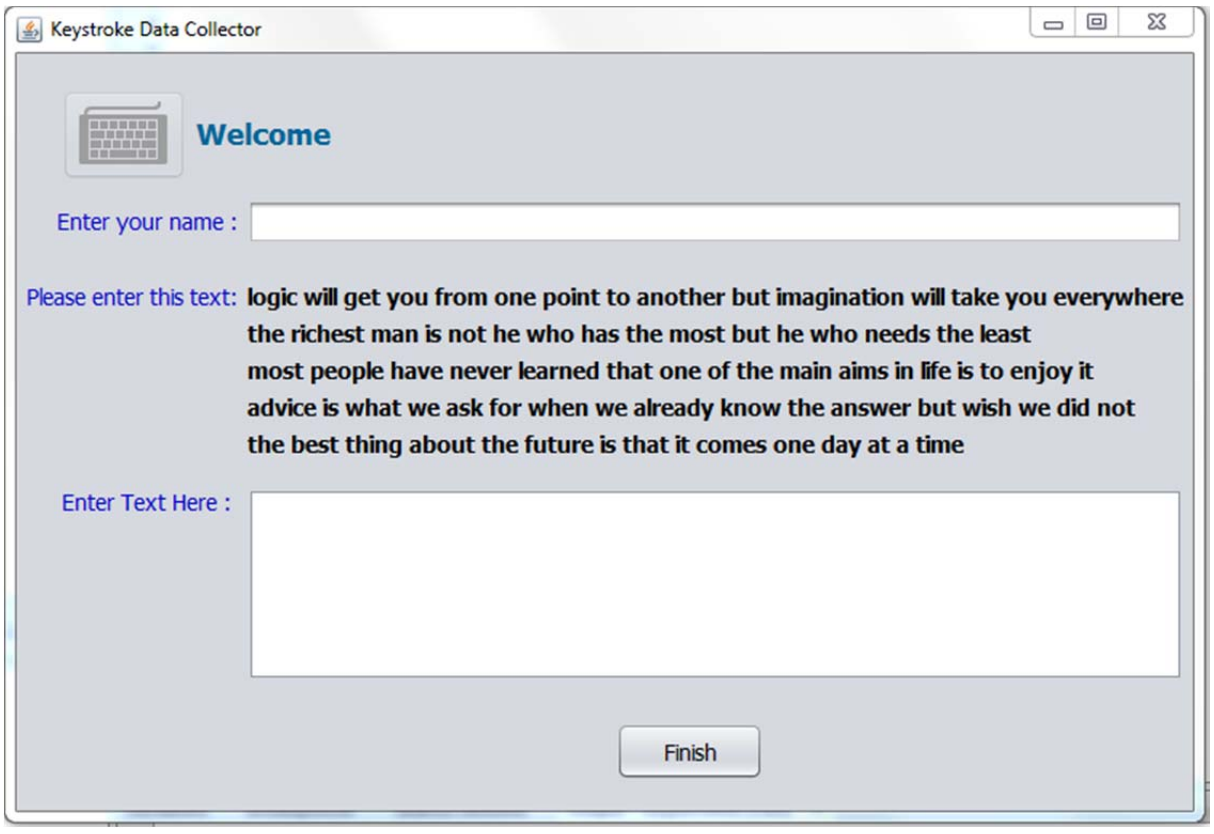

Figure 3.3: A screen-shot of the data collection program for the original scheme experiment.

The decision to collect data as the users were typing on a text area, rather than gathering it while they were using regular applications such as word processors or programming editors, was because of previous work done in [35]. The research done in that study concluded that the keystroke behaviour of a single user varies for different applications. This could add more complexity to the system and interfere with the purpose of this research.

Ethical concerns about key-logging sensitive data were another reason for restricting the data collection process. Data collection was chosen to be done via a separate application, as opposed to a keylogger-like manner, because of the security concerns that follow the use of keyloggers [159]. Users are more likely to reject getting involved in such an experiment, fearing that important information such as IDs/Passwords can be 'accidentally' captured. Moreover, the application used for data collection is similar to word processing applications such as Word and Notepad, therefore the user was able get familiar with the requirements very quickly.

```
$ | 14232027812 | DOWN |
$ | 14232948024 | UP |
R | 14232569880 | DOWN |
R | 14233043428 | UP |
E | 14234914503 | DOWN |
E | 14235313349 | UP |
A | 14235363284 | DOWN |
A | 14235812108 | UP |
D | 14236036757 | DOWN |
D | 14236435939 | UP |
I | 14237384043 | DOWN |
I | 14237807844 | UP |
N | 14238057237 | DOWN |
N | 14238431265 | UP |
G | 14239104939 | DOWN |
G | 14239429151 | UP |
```

Figure 3.4: Data collected from typing "Reading".

The data collection is done by capturing certain attributes for every key action performed on the keyboard. These attributes are: the character of the key (produced from the ASCII code of the key), the time of the action, whether it was an up or down action. These attributes are then used for key-pair identification and features extraction. An example of the data collection

process of the text “Reading” is shown in Figure 3.4. Each raw in the figure illustrates the information captured for each action performed on the keyboard. The first column is the character of the key which the action is performed on, the second column is the exact time the action was performed (this time is the number of ticks since the Unix Epoch, which is a fixed point in time referring to January 1, 1970 00:00:00 UTC [160]), and the last column indicates the action type, i.e. up or down.

Moreover, every two keys pushed successively yield in a key-pair. For each key-pair, the following attributes are identified: key1, key2, H1, H2, DD, UU, UD , the key-pair category and whether one of the keys is a special function key such as: Space, Enter, CapsLock, NumLock, Shift, Control, etc. Key-pairs including any of the special function keys are not classified for reasons discussed in Section 3.2.1.1. An example of the key-pairing process of the text “Reading” is shown in Table 3.5. Each raw in the figure illustrates the information concluded for each key-pair in the entered text. The first and second columns are the first and second characters of the key-pair respectively. The third, fourth, fifth, sixth and seventh columns are correspondingly the H1, H2, DD, UU, UD times for this key-pair (the details for computing these times are explained in Section 3.2.1.2). The final column indicates the key-pair category or whether the key-pair contains any special character.

Table 3.5: Key-pairs formed from typing “Reading”.

First char	Second char	H1	H2	DD	UU	UD	Key-pair type
$	R	295.121735	151.871859	30.597073	173.846949	-121.274786	RightShift
R	E	151.871859	127.914137	727.987708	751.945430	600.073571	Adjacent
E	A	127.914137	143.942611	159.957294	143.928820	16.014683	SecondAdjacent
A	D	143.942611	128.021896	200.069209	215.989925	72.047314	SecondAdjacent
D	I	128.021896	135.917470	439.984465	432.088891	304.066995	NonAdjacent
I	N	135.917470	119.954741	199.937718	215.900446	79.982977	SecondAdjacent
N	G	119.954741	103.978222	320.032610	336.009129	216.054387	SecondAdjacent

3.2.2.2 Data Space

As mentioned earlier, the five timing features were extracted for each key-pair appearance in the text. Then, the mean of each timing feature was calculated for all of the key-pair appearances. This was done for all five types of key-pairs.

Nonetheless, by observing users’ behaviours, it was noticed that some users take small pauses for different reasons, such as moving their eyes to read the provided text; therefore, any

outlier data is discarded. Outlier data is identified as being as much as three standard deviations above or below the mean [36]. These particularly very large or very small data points were discarded from the final data as they represented noise that might affect the overall system performance [48].

Around 0.01% of the overall data was discarded for being more or less than the specified threshold. The amount of discarded data is small enough to not affect the final dataset immensely. Experimenting with smaller thresholds were performed, yet a great deal of the dataset was omitted which affected the final results in a sense that the delicate differences between the individuals' typing patterns were lost, therefore the error rate was higher.

In addition, it was seen as preferable to normalize the data before handing it to the machine learning technology [161]. Therefore, all the data was normalized to be between [0,1] to add a sense of uniformity to the data as attributes in greater numeric ranges might have dominated those in smaller numeric ranges [162].

The final step involves creating the user's timing vector and storing it in the database as the user's template. The timing vector (V) includes five sequences (sub-vectors) (i.e. V_A, V_S, V_T, V_F, V_N) consisting of five means each. The sub-vectors represent the means of the five timing features for each of the five key-pairs. The following equations show a timing vector that describes a user's typing profile:

$$V= \{V_A, V_S, V_T, V_F, V_N\} \quad (3.1)$$

$$V_A= \{\mu_{A\text{-}H1}, \mu_{A\text{-}H2}, \mu_{A\text{-}DD}, \mu_{A\text{-}UU}, \mu_{A\text{-}UD}\} \quad (3.2)$$

⋮

$$V_N = \{\mu_{N\text{-}H1}, \mu_{N\text{-}H2}, \mu_{N\text{-}DD}, \mu_{N\text{-}UU}, \mu_{N\text{-}UD}\} \quad (3.3)$$

Where:

$\mu_{A\text{-}h1}$: denotes the mean of the H_1 timing feature for all adjacent key-pairs.

$\mu_{A\text{-}h2}$: denotes the mean of the H_2 timing feature for all adjacent key-pairs.

… and so on.

3.2.2.3 Finding Distance

At the log-in phase, the user is asked to enter a test text that is different from the text entered at the enrolment phase. The system then prepares the timing vector for the test data, which includes the means of the five timing features extracted from the five types of key-pairs, using exactly the same steps that were followed to create the timing vector for the training data (user profile).

Once the timing vector for the test data has been prepared, the Euclidean distance [163] is calculated between the log-in vector and the vector stored at the user's profile in the database. This distance is calculated using the following formula:

$$dis(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \quad (3.4)$$

Where n is the number of components in the timing vector, p and q are the training and testing vectors.

An example is illustrated in Figure 3.5 in which (A) shows the similarity between the train and test vectors for one user, while (B) shows the difference between the train vector of one user and the test vector of another user.

The Euclidean distance must be small enough to ensure that a reliable decision can be made regarding the identity of the user. However, the distance will never be exactly zero because human behavioural characteristics are not always consistent [50]. Therefore, an acceptable value for the distance is determined and, if it is not exceeded, the user is accepted as genuine; otherwise, the user is denied access. This value, or threshold, is determined based on each user's profile data [36]. After several trials, a threshold described as the mean plus one-and-a-half standard deviations was chosen [36]. This local threshold is calculated individually for each user based solely on the training data used to build the timing vector in that user's profile.

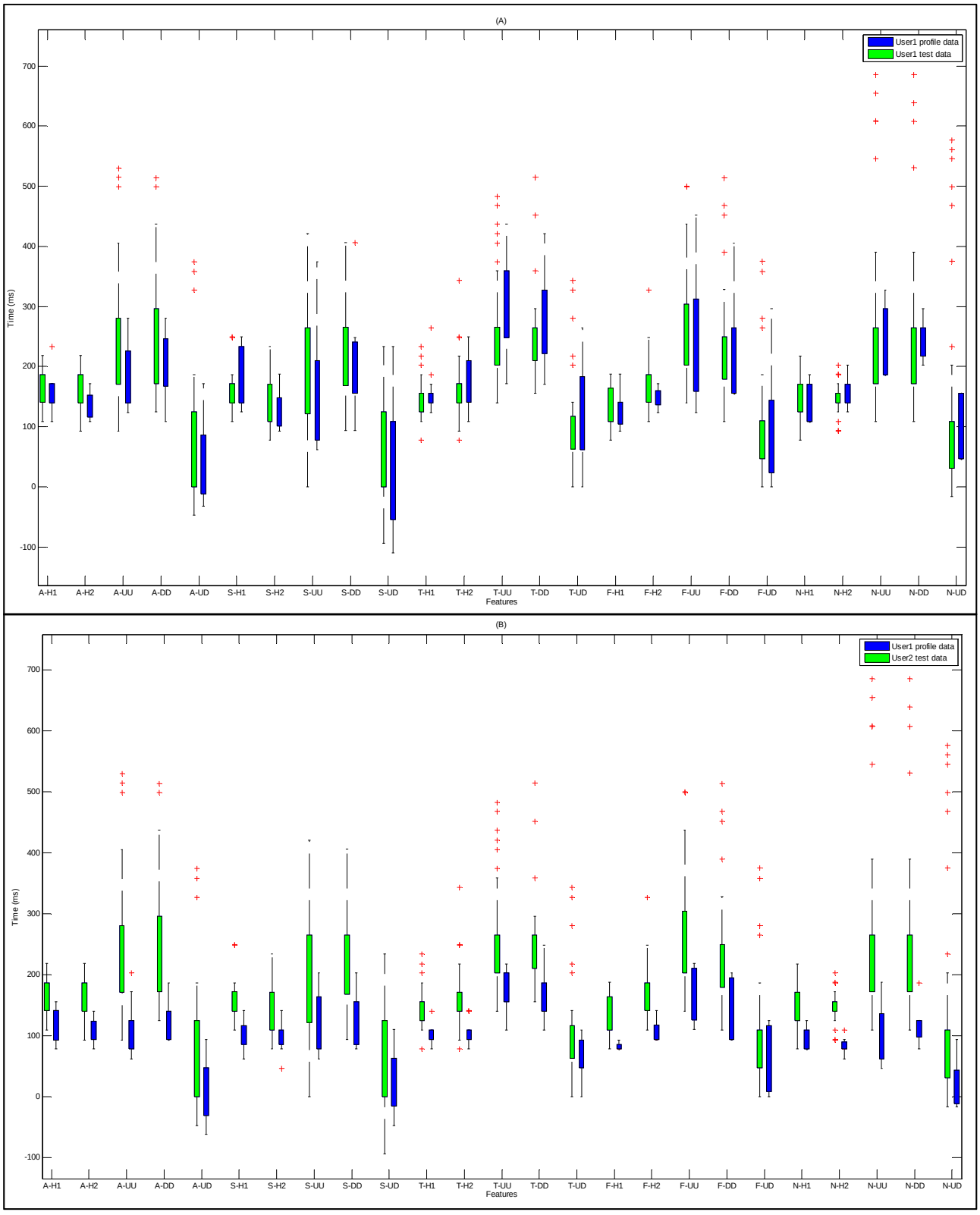

Figure 3.5: (A) The same user's timing vectors (B) Different users' timing vectors.

3.2.2.4 Experiment and Results

Each participant was asked, first, to enrol once in the system in which his or her timing profile was calculated and stored in the database. This is the only sample that the user provides to the system as a training sample. Then, participants were told to provide two testing samples which are different to the training sample.

The system performs key-pair formation and feature extraction which outcomes are stored in the timing vector. This is done for both training and testing samples. Euclidian distance is then computed between training and testing samples. The result of the Euclidian distance is lastly compared against the chosen threshold to decide if the test sample and the training sample are from the same individual. The flow in which the enrolment and log-in phases are carried-out is illustrated in Figure 3.6.

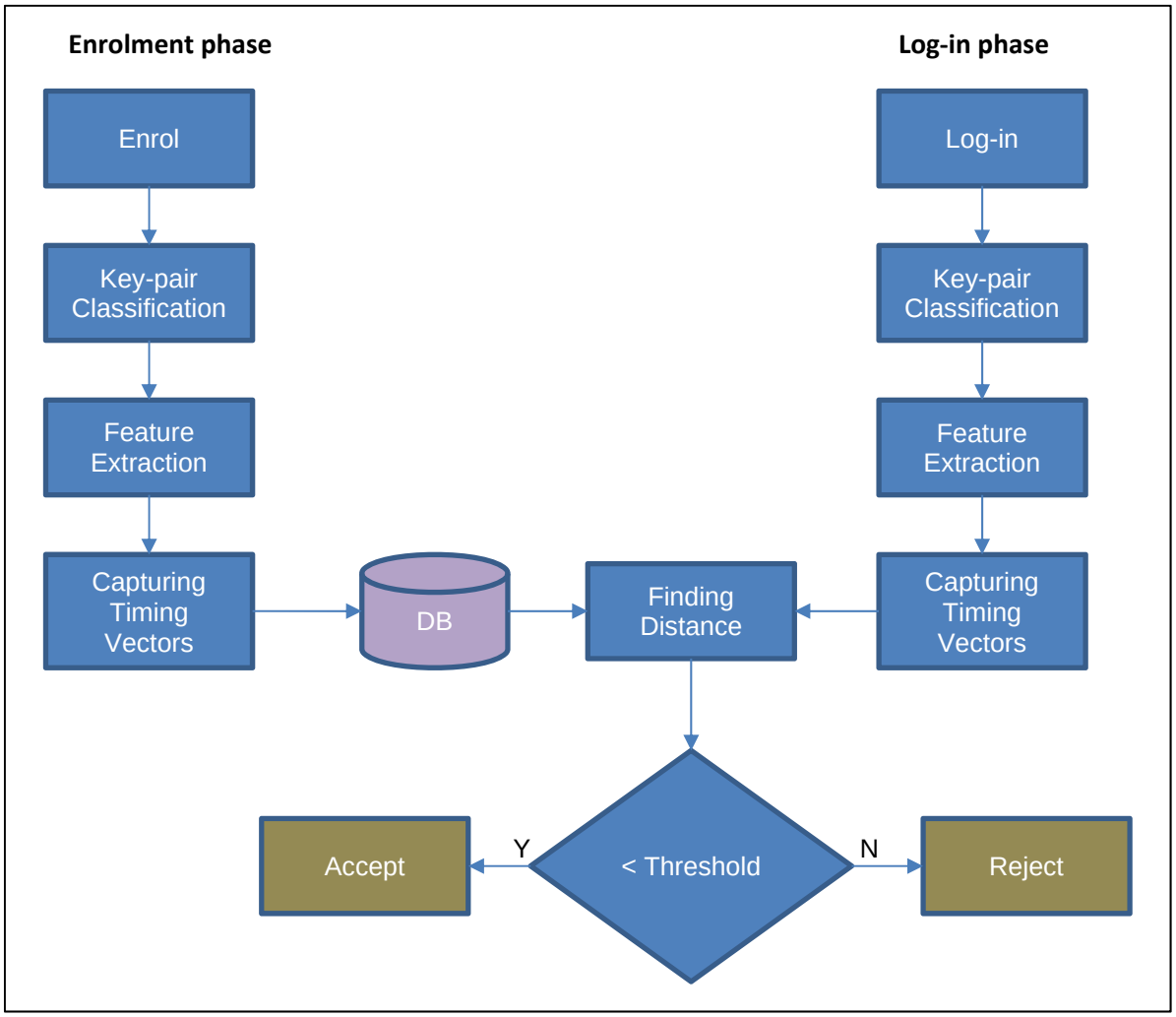

Figure 3.6: Flow of the system.

Two error rates, the FAR and the FRR, were used to determine the performance of the authentication method. These error rates are explained in detail in Section 2.7. As mentioned in Section 2.7, there are various error rates used in the literature to report the system performance. The reason for choosing the FAR and FRR is to provide a better understanding of the security that the system delivers. In biometric systems, operating at a low FAR, where only few imposters get access, and at low FRR, where few genuine users are denied access, is of highest priority. Therefore, FAR/FRR conveys more information about the system's security compared with other error rates such as accuracy or EER [164].

In this study, the testing process was conducted in two ways:

1) Legitimate user testing: each user's timing profile is compared to his or her two sets of testing data.
2) Imposter user testing: each user's timing profile is compared to the two testing data samples of another participant, which was randomly selected.

The results from using the five timing features were analysed separately, and those that used a combination of these features were investigated as well. The details of each individual's results are shown in Appendix C.1

Table 3.6: Error rates for individual features.

Features	FAR	FRR
H1	0.9	0.167
H2	0.867	0.1
UU	0.933	0.1
DD	0.833	0.167
UD	0.867	0.067

Table 3.7: Error rates for combination of features.

Features	FAR	FRR
H1 + H2	0.667	0.267
H1 + UD + H2	0.367	0.5
DD + UD + UU	0.467	0.533
H1 + DD + UD + UU + H2	0.033	0.867

Table 3.6 clearly states that the use of individual features produces better FRR compared with the produced FAR. Nonetheless, latency features and hold features have similar performance outcomes (all results are provided in a 0-1 scale).

The FAR was improved when using a combination of features, as shown in Table 3.7. More so, using a combination of all the features produced the best FAR. Yet, the FRR deteriorated in the combined features. The FAR was lower than the FRR in the combined features including latency features. Merging the two hold times did not greatly change the error rates of each of them individually.

After examining the results, the trade-off between the FARs and the FRRs was clear, as illustrated in Figure 3.7. This trade-off between FAR and FRR is unavoidable. Therefore, combining the features results was significant, since the security of the system is of higher priority, and a low FAR is more important in this study. This indicates that more imposters are detected, yet inescapably genuine users will face more access rejections.

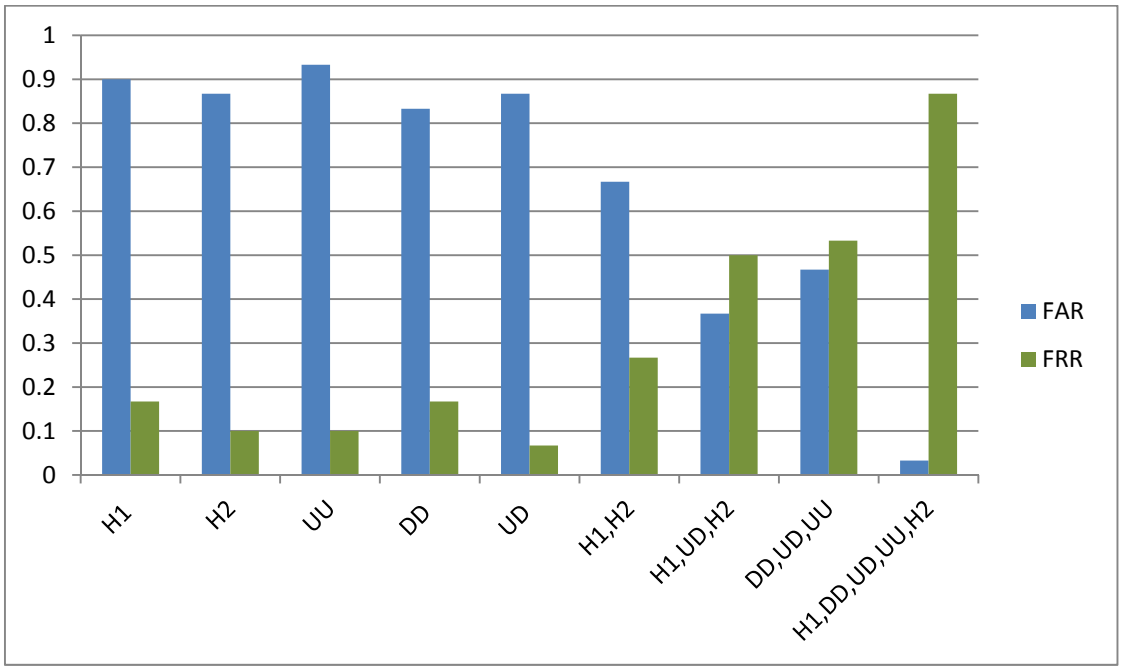

Figure 3.7: Original technique error rates.

3.2.2.5 Discussion

The results of this research, thus far, are comparable to some free-text studies, such as the work done by Hempstalk et al. [80], which resulted in a 0.113 FAR and a 0.204 FRR.

However, it is not at the same level as ground-breaking studies such as that done by Gunetti and Picardi [8], which produced a 0.00005 FAR and a 0.05 FRR. Nevertheless, both studies incorporate a relatively large number of enrolment samples, which are 3,000 samples from 19 participants and 765 samples from 205 participants, respectively. This can be directly compared to this study's one short sample per participant, i.e. 15 samples from 15 participants.

An important factor that enhances the practicality of this system is the text length used for the training. The shortness of the text used for training is highly desirable from the users' perspective. This is not delivered by most methods available in the literature which impose the collection of long and time-consuming input from the users [79]. This requirement interferes with the main goal of this study that focuses on authenticating the user based on the smallest amount of text possible, and therefore, relieving the user from the tedious task of enrolment, i.e. easier training.

This technique clearly needs to be refined in order to improve the overall system performance. Modifications that involve more precise key-pair classification and more advanced classification technology are important to improve this method. This will be introduced in the extended technique discussed in the next section.

3.3 Extended Technique

In the extended technique, the research continues to consider the keyboard-layout based method to compare the timing features of free-text typing samples. A larger feature set is deployed in this research for the purpose of trying to find the best representative features of the typing patterns in human behaviour whilst continuing to provide users with the easy training they long for as the training process involves the user typing only little input. The five different key-pairs introduced in the original technique which depended on the position of the two keys on the keyboard with relation to each other have been expanded to essentially include a sense of hand positioning on the keyboard.

This extended key-pair based method for extracting features was followed, similar to the original scheme, in order to escalate the number of the common key-pairs in the training and the testing samples, i.e. di-graphs found in both samples. This will improve the stability of the means of the timing features extracted from these key-pairs which corresponds to the

main elements of the user's timing vector. Therefore, the utilization of the least amount of text possible in the best way possible is achieved.

The new, more specific features cater to most typists, who use their two hands in typing. Therefore, the speed of which they type two adjacent keys, for example, using their left hands differs to that which they type with their right hands [165]. Nonetheless, this addition is not at the cost of longer training as the extended key-pair approach is still able to utilize the least amount of text in the best way possible.

Extracting the best feature set for this problem was carried-out using two different mechanisms: a wrapper and a filter subset selection techniques. They are: Ant Colony Optimization (ACO) and Multivariate Analysis of Variance (MANOVA). Moreover, One-vs-One and One-vs-Rest Support Vector Machines (SVMs) were used to classify users.

3.3.1 Improvements on the Original Method

While observing the subjects during their typing processes, it was noticed that the original method works well with those who use only one hand during the typing process. Therefore, a more sophisticated key-pair classification, taking into consideration the position of the hand on the keyboard, has been applied. This method distinguishes between the key-pairs located on the left side of the keyboard, those on the right side of the keyboard and those that are scattered between the two sides. This alteration is aimed at producing a better representation of the typing patterns for people who use their right hands for pressing keys on the right side of the keyboard and their left hands for the keys on the left side of the keyboard. This is due to the fact that the typing speed of a particular key-pair may be different across the two hands [165].

Moreover, the clock resolution is a critical part of the process of extracting the keystroke timing features. In fact, according to the study in [166], the clock resolution in keystroke dynamics systems should not, preferably, exceed 1 millisecond (ms). Another study that was conducted in [167] to explore the consequences of the clock resolution on keystroke dynamics discovered that the EER increased by around 4.2% when a 1ms resolution clock was replaced by a 15ms resolution clock. This is a significant reason why the performance produced by the original method was not sufficient. A Java written program was used for extracting time features, in which the clock resolution varies between operating systems [35]. Java programs running on a Windows operating system, which is the case for this study, have

between 10ms and 15ms timing resolution [168]. This made the decision to switch to C++ programing language unavoidable. The clock resolution for C++ is 1 microsecond (μs) when using the function QueryPerformanceFrequency() [169]. Therefore, converting to C++ yields improvement in the timing precision for the timing keystroke features.

In addition, Java applications need the Java Runtime Environment (JRE) to be present on the machine before it can run the application. However, JRE was not available on most of the participants' machines. Therefore, data collection was performed on only one specific machine, in one set. The subjects reported that they found this difficult because they were not accustomed to this new machine. Using C++ solves this problem as programs written in C++ do not have any requirements regarding the machines they run on. This makes it possible to send the application to participants by e-mail so they can download and use it in the convenience of their own computers and at times that are suitable for them. This should result in capturing their realistic typing behaviours and, therefore, increase the level of consistency among the typing samples. This also allows for the data collection to be performed on more than one set, which will help to capture a broader range of the user's typing behaviour.

In addition, the use of advanced technology in both feature subset selection and samples classification is bound to improve the overall system performance. The introduction of feature subset selection using ACO and MANOVA helps to improve the separablity property of the features used for user authentication. Moreover, switching the primitive Euclidian distance for the more reliable and highly advanced SVMs also assists in improving the overall performance level.

3.3.2 Feature Definition

The features used in the extended key-pairing scheme are described in this section. The key-pair formation and feature extraction are both explained here.

3.3.2.1 Key-pair Formation

The extended approach, similar to the original scheme, uses the keystroke features extracted between two keys (key-pair) that are pressed consecutively and have a relationship on the keyboard layout. This relationship depends mainly on the key position of each character on the keyboard with relation to the other character. Moreover, these relationships can vary depending on the location of the two keys with respect to the overall keyboard layout.

Similar to the original technique, there are five categories for key-pair relationships:

1) *Adjacent:* keys located next to each other on the keyboard.
2) *Second adjacent:* keys that are one key apart from each other.
3) *Third adjacent:* keys that are two keys apart.
4) *Fourth adjacent:* keys that are three keys apart.
5) *None adjacent:* keys that are more than three keys apart.

Figure 3.1 (in page 65) illustrates the key relationship concept; while considering the key 'G'. This is exactly the same as the original method introduced in Section 3.2.1.1.

Each of these relationship categories can fall into one of the following overall locations:

- ☐ Both keys are on the right hand side of the keyboard.
- ☐ Both keys are on the left hand side of the keyboard.
- ☐ The two keys are located on different sides of the keyboard, i.e. the first key is located on the right hand side while the second key is on the left or vice versa.

For further explanation, in Figure 3.8, the green section of the keyboard represents the right hand side while the purple keys represent the left hand side. Other keys that are not located in the main part of the keyboard, e.g. the num-pad keys, arrows and the function keys were excluded from the key-pair formation; they are therefore indicated with white colour.

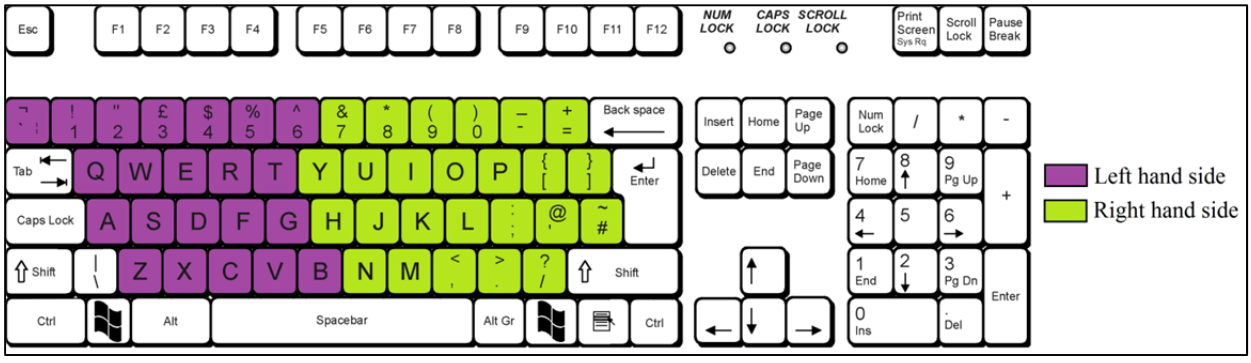

Figure 3.8: Overall key location.

Based on the previously explained technique, the key-pairs formed from the text "kjbr1," are:

- "kj": Adjacent/RightSide.
- "jb": SecondAdjacent/ DifferentSide.
- "br": ThirdAdjacent/LeftSide.

- “r1”: FourthAdjacent/LeftSide.
- “1,”: NonAdjacent/DifferentSide

This process is pursued to break the text down into key-pairs and then classify each key-pair typed on the main part of the keyboard. In total, there are fifteen different combinations of key-pairs that any two keys can be classified into.

The extended key-pairing method was used to increase the soundness of the mean of the timing features that builds the users’ profiles, without compromising the typing characteristics that might differ when a key-pair is typed with the user’s right or left hand.

3.3.2.2 Feature Extraction

The five features in the original technique, i.e. H_1, H_2, UU, DD, UD, described in Section 3.2.1.2, are also used in the extended method.

Table 3.8: Extended approach feature set.

Key-pair Category	Feature Set				
Adjacent/RightSide	AR-H1	AR-H2	AR-DD	AR-UU	AR-UD
Adjacent/LeftSide	AL-H1	AL-H2	AL-DD	AL-UU	AL-UD
Adjacent/DifferentSide	AD-H1	AD-H2	AD-DD	AD-UU	AD-UD
SecondAdjacent/RightSide	SR-H1	SR-H2	SR-DD	SR-UU	SR-UD
SecondAdjacent/LeftSide	SL-H1	SL-H2	SL-DD	SL-UU	SL-UD
SecondAdjacent/DifferentSide	SD-H1	SD-H2	SD-DD	SD-UU	SD-UD
ThirdAdjacent/RightSide	TR-H1	TR-H2	TR-DD	TR-UU	TR-UD
ThirdAdjacent/LeftSide	TL-H1	TL-H2	TL-DD	TL-UU	TL-UD
ThirdAdjacent/DifferentSide	TD-H1	TD-H2	TD-DD	TD-UU	TD-UD
FourthAdjacent/RightSide	FR-H1	FR-H2	FR-DD	FR-UU	FR-UD
FourthAdjacent/LeftSide	FL-H1	FL-H2	FL-DD	FL-UU	FL-UD
FourthAdjacent/DifferentSide	FD-H1	FD-H2	FD-DD	FD-UU	FD-UD
NonAdjacent/RightSide	NR-H1	NR-H2	NR-DD	NR-UU	NR-UD
NonAdjacent/LeftSide	NL-H1	NL-H2	NL-DD	NL-UU	NL-UD
NonAdjacent/DifferentSide	ND-H1	ND-H2	ND-DD	ND-UU	ND-UD

In summary, five timing features were defined for each key-pair appearance in the text. This was done for all fifteen types of key-pairs. Therefore, the overall number of timing features was 75 (5 timing features * 15 key-pairs). The means of these features are calculated and stored in the timing vector of each user, i.e. user's profile. Table 3.8 lists all the 75 features extracted from all key-pairs. The features abbreviations listed in the table combine the key-pair category and the timing feature, for example: "AR-H1" stands for: Adjacent/RightSide-Hold1 and so on.

3.3.3 Feature Subset Selection

Having such a large feature set will add more computational cost in addition to raising the complexity of the classification process [170]. Therefore, it is necessary to incorporate a feature subset selection mechanism to indicate the features that most represent users' typing behaviour.

Feature subset selection is considered as an optimization problem, where the space of all possible features is scrutinised to recognise the feature or set of features that produce optimal or near-optimal performance, i.e. minimize classification error [170]. Feature selection is applied to high dimensional datasets as a pre-processing step before classification is performed [19]. It counterbalances the lack of quality in the data by removing any irrelevant, noisy features. In addition to improving the classification performance and avoiding overfitting, feature selection is used to deliver a faster and more cost-effective method [170]. Nonetheless, feature subset selection causes overload on the system as the search for a subset of significant features adds an extra layer of complexity to the overall process [170].

Feature subset selection algorithms are categorized into two main approaches based on their relation with the classifier. The first is the wrapper approach, in which the features selection depends on the learning algorithm [171]. In this approach, a search algorithm is used to search through all the possible features and evaluate each subset by running a classifier on the subset and then considering the subset with the best performance [171]. Since the space of feature subsets normally grows exponentially with the number of features, heuristic search techniques are introduced to direct the search to the optimal subset.

Two classes of search methods are mainly used: deterministic and randomized search algorithms [170]. Randomized search techniques, such as: Genetic algorithms [172], are less prone to get stuck in a local optimum compared to deterministic algorithms, such as

sequential forward selection (SFS) [173] and sequential backward elimination (SBE) [173], where both are considered greedy search methods.

On the other hand, the second approach is the filter; in this scheme the feature selection is done independently of the learning algorithm. This approach evaluates the significance of features based only on the statistical or probabilistic properties of the data [174]. This is done by computing a relevance score and eliminating features with low-scoring. The remaining subset of features is then used for further steps relating to the classification algorithm. Figure 3.9 shows a brief illustration of both the wrapper and the filter approaches.

The wrapper approach suffers from the computational overhead of evaluating all candidate feature subsets using the selected learning algorithm [175]. It also can face a higher risk of over-fitting [170]. The filter approach, on the other hand, is computationally more effective and easily scalable to higher-dimensional datasets, but, unlike wrapper methods, it overlooks the interaction with the classifier which may cause the feature subset space search to differ from the hypothesis space search [170]. Moreover, many techniques used in the filter approach are univariate, i.e. each feature is evaluated separately, such as t-test [176] and analysis of variance (ANOVA) [177]. This is a reason for ignoring feature dependencies leading to poorer classification performance. Therefore, more multivariate techniques were suggested to integrate feature dependencies to filter approaches, such as correlation-based feature selection (CFS) [178] and the Markov blanket filter (MBF) [179].

In the keystroke dynamics field, a verity of feature subset selection methods was employed over research projects. The Genetic Algorithm (GA) model was used to derive feature subsets in [180, 181]. Moreover, Particle Swarm Optimization (PSO) was used in addition to GA for feature selection in [169]. Ant Colony Optimization (ACO) was used in addition to PSO and GA in the research done in [5, 182]. Based on feature reduction rate and classification accuracy, both studies proved that ACO yields better performance than PSO and GA. Moreover, in [70] a pre-processing step involving Principle Component Analysis (PCA) was used to reduce the dimensions of the feature vectors.

In this research, both a wrapper and a filter method were designated to act as a feature subset selection mechanism. This was decided in order to compare and contrast the two approaches and analyse their effect on the overall performance of the system. The technique used to represent the wrapper approach is the Ant Colony Optimization (ACO), whereas the method chosen to demonstrate the filter approach is the Multivariate Analysis of Variance

(MANOVA). Reasons for choosing these schemes are presented in Section 3.3.5.3. The two methods are described briefly in the following two sections.

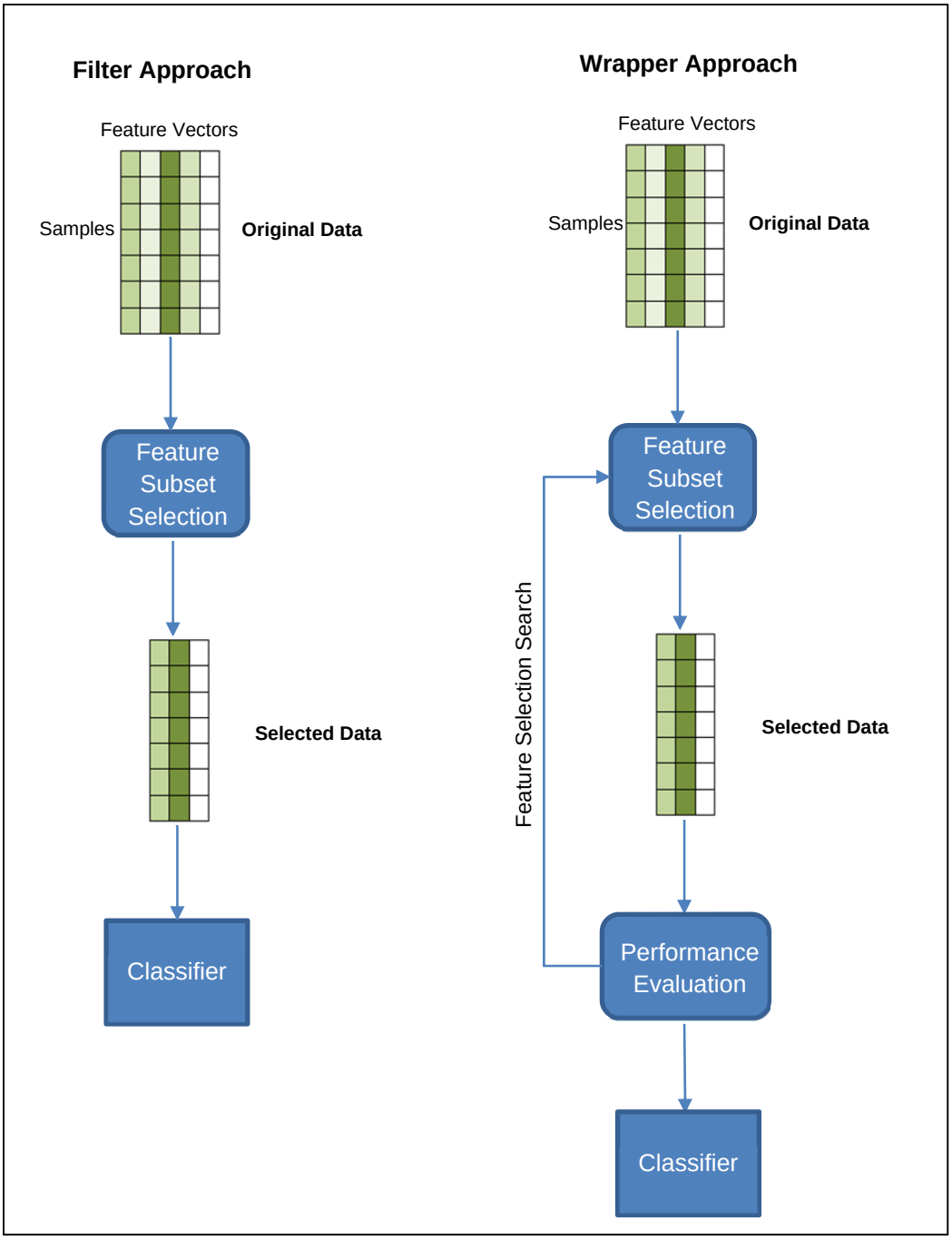

Figure 3.9: Wrapper and filter feature subset selection methods.

3.3.3.1 Ant Colony Optimization (ACO)

Ant colony optimization is an optimization technique that was introduced in the early 1990's by Dorigo and his colleagues [179, 180]. The technique was inspired by the foraging

behaviour of real ants, as shown in Figure 3.10. This behaviour, called stigmergy, was discovered by the French biologist Grasse in the late 1950's [185]. It involves the indirect communication between ants using chemical pheromones that they leave on trails, which permits them to find the shortest path between the nest and the food supply. This behaviour is utilized in Ant Colony Optimization to search for approximate solutions and discrete optimization problems [186].

ACO is one of the most successful mechanisms of swarm intelligence [187]. Swarm intelligence aims to design intelligent multi-agent systems whose inspiring source is the collective behaviour of social animals and insects such as birds, fish, ants, bees and wasps [188].

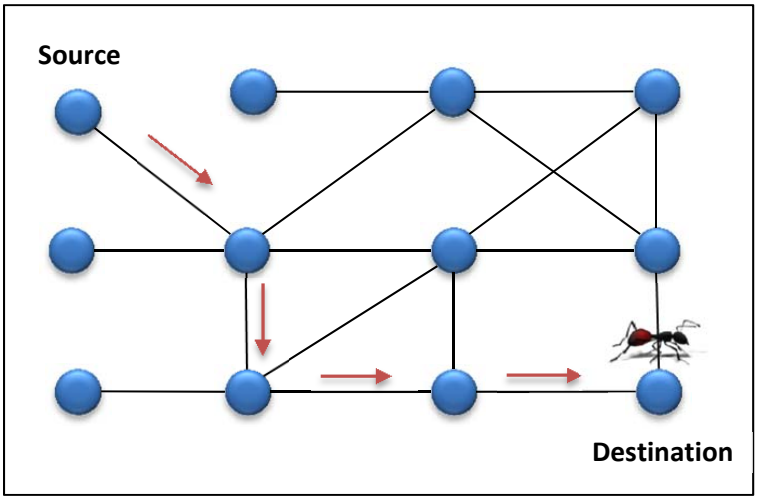

Figure 3.10: Ants' behaviour to find paths from a source node to a destination node.

ACO works at first by ants randomly explore the area surrounding their nest. They leave a chemical pheromone trail on the ground while moving around which can be smelled by other ants. Ants tend to choose paths marked by strong pheromone levels when choosing their way. Whenever an ant finds a food source, it evaluates the quantity and the quality of the food and changes the amount of the pheromone it leaves on the path back to the nest accordingly. Afterwards, these pheromone trails will guide other ants to the food source via the shortest path. The pheromone level left on the ground will decrease with time; therefore, only paths with strong amounts of pheromone will stay to guide ants [188].

One of the problems that the research community has simplified to obtain scientific test cases for ACO is the well-known traveling salesman problem (TSP) [189]. The TSP replicates the

scenario of a travelling salesman who must pass through a number of cities. The travelling salesman intends to navigate between these cities so that the total travelling distance is minimal, while visiting each city exactly once. After that, ACO was successfully applied to a great number of problems such as the quadratic assignment problem (QAP), routing in telecommunication networks, graph colouring problems, scheduling, etc. [190].

The first ACO algorithm developed was the Ant System (AS) [183], which Dorigo created for his masters dissertation. Since then, several improvement of the AS have been developed, many of which were by Dorigo himself such as: Elitist AS [191], Ant-Q [192] and Ant Colony System [193]. Other improvements to the original system were presented by different researchers, including: MAX-MINAS [194] and Hyper-Cube AS [195].

The ACO algorithm can be applied to any optimization problem that the following aspects can be defined for [188]:

- Appropriate problem representation: This insures that the problem can be expressed as a graph consisting of a set of nodes and edges between them.
- Heuristic desirability (η) of edges: It measures the “goodness" of paths from one node to another in the graph.
- Construction of feasible solutions: A mechanism to ensure that only feasible solutions are constructed which needs defining of suitable traversal stopping criteria for stopping path construction whenever a solution is achieved.
- Pheromone updating rule: A technique for updating the pheromone levels on edges which utilizes a corresponding evaporation rule. This involves updating the paths that the n best ants chose.
- Probabilistic transition rule: This is the rule that controls the probability of an ant traversing from one node to another in the graph.

Constructing a solution initially begins with an empty partial solution and then the solution is extended in the following steps by adding a feasible solution component from the set of solution components [190]. The transition rule for any ant 'm' that allows it to decide on including the i^{th} feature at any time ‘t’ in the solution is influenced by two aspects: the heuristic and level of pheromone. Often a classifier performance is used as heuristic information for feature selection [188].

The probabilistic transition rule is calculated as follows:

$$P_i^k(t) = \begin{cases} \frac{\tau_{i(t)}{}^{\alpha} * \eta_i{}^{\beta}}{\sum_{j \epsilon h^k} \tau_{j(t)}{}^{\alpha} * \eta_j{}^{\beta}} & \text{if } i \epsilon h^k \\ 0 & \text{otherwise} \end{cases} \quad (3.5)$$

Where h^k is the set of feasible features that can be added to the partial solution; τ_i is the pheromone value and η_i is the heuristic desirability, they are both associated with feature i. The two parameters α and β are used to control the relative importance of the pheromone value and heuristic information. As mentioned earlier, the value of local heuristic desirability η_i for the i^{th} feature is assessed using classifier classification accuracy used in the problem.

The process of pheromone evaporation on all nodes is activated after all ants have completed their solutions. The goal of pheromone evaporation is to escape the state in which all ants construct the same solution [188]. This is done by dropping larger pheromone amounts on good routes. This is achieved by having ants deposit an amount of pheromones depending on the quality of their solution, i.e. classification accuracy. In addition, to increase the usefulness of dropping pheromones on routes, a little bit of the pheromones is removed at the end of every iteration to emphasize the pheromone reduction on less quality routes. Evaporation rate is shown in Equation 3.6 which shows each ant k depositing a specific quantity of pheromone on each node i that it has navigated.

$$\Delta\tau_i^k(t) = \begin{cases} \varphi * \frac{C\left(S^k(t)\right)}{|S^k(t)|} + (1-\varphi) * \frac{(N - |S^k(t)|)}{N} & \text{if } i \epsilon S^k(t) \\ 0 & \text{otherwise} \end{cases} \quad (3.6)$$

Where $S^k(t)$ is the feature subset found by ant k at iteration t, and $|S^k(t)|$ is its length while $C(S^k(t))$ is the classifier performance for that ant at that iteration. N is the total number of features in the data set. The parameter φ controls the relative weight that controls the importance of the classifier performance and the feature subset length.

At the end of every iteration, the pheromone update is performed on all nodes. This is done by depositing the new pheromone which includes the effect of pheromone evaporation. Pheromone update is computed as:

$$\tau_i(t+1) = (1-\rho) * \tau_i(t) + \sum_{k=1}^{m} \Delta\tau_i^k(t) \quad (3.7)$$

Where ρ is the pheromone trail decay coefficient which ranges from 0 to 1, m is the number of ants and $\Delta\tau_i^k(t)$ is the evaporation rate computed in Equation 3.6.

The stopping criteria for feature selection has been targeted in numerous ways such as using a fixed number of features [196] in which the user defines the minimum and maximum limit for the feature subset length. The ants, consequently, stop choosing the next feature if that maximum number of features is reached. Another often used stopping criterion is accuracy inversions [197] which corresponds to a selection of a feature degrading the performance. It works by the ants stopping the feature selection and returning the subset whenever a max number of inversions is reached.

The process of feature selection using ACO starts with generating a number of ants which are then placed randomly on the graph. Often, the number of ants is chosen to be equal to the number of features; this allows each ant to begin constructing its path at a different feature [188]. Then, ants traverse nodes using the probabilistic rule until the stopping criterion is met. The resulting subsets are produced by all ants is then gathered and evaluated. This process stops if an optimal subset has been obtained or the algorithm has executed a specific number of times. The best feature subset encountered is output as the best solution. If these two conditions have not been satisfied, the pheromone is updated and a new set of ants are created and the process repeats again [190]. The overall process of ACO feature selection is shown in Figure 3.11.

ACO was utilized in keystroke dynamics in several studies. An example of the studies utilizing ACO, together with other feature selection techniques, is the one performed in [5]. In addition to ACO, Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) were applied to the data before feeding it into a back propagation neural network (BPNN) classifier. Based on feature reduction rate and classification accuracy, this study proved that ACO yields better performance than PSO and GA.

Moreover, while ACO, PSO and GA were all used in [182] for feature subset selection, the Extreme Learning Machine (ELM) was chosen to be the learning method. Supportive of the conclusions found in [5], this work demonstrated that ACO results in the best feature subset selection with ELM.

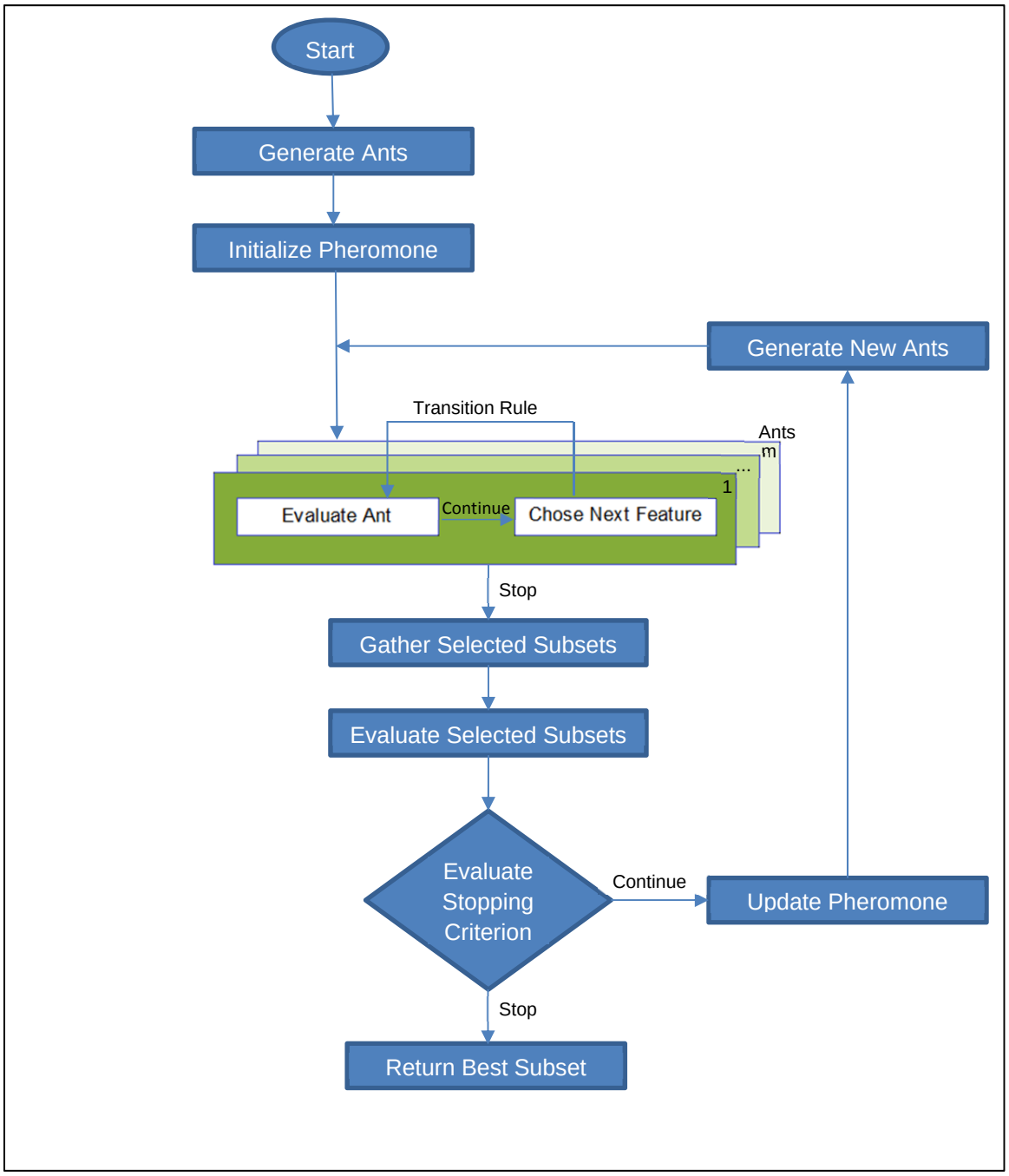

Figure 3.11: ACO feature selection process.

3.3.3.2 Multivariate Analysis of Variance (MANOVA)

MANOVA is considered as an evolution of the analysis of variance (ANOVA), as they are both statistical techniques used to analyse variability in data. ANOVA is used for analysis of variance in which there is only one dependent variable whilst MANOVA can be used for more than one dependent variable. A description of both techniques is provided in this section.

3.3.3.2.1 ANOVA

Analysis of variance (ANOVA), as the name implies, is a statistical technique used to analyse the variability in data in order to deduce the inequality among population means [198]. It was first introduced by Fisher [199] more than 70 years ago. ANOVA is similar to a t-test but it allows the comparison between more than two groups (populations) [200]. Nevertheless, using repeated t-tests to compare more than two means will result in a higher Type I error rate (or FRR) [200]. The Null hypothesis, in this statistic measure, refers to whether or not the different sample means come from the same population:

$$H_0 = \mu_1 = \mu_2 = \cdots = \mu_n \tag{3.8}$$

ANOVA considers the ratio between two types of variances, namely: variance between populations and variance within populations [198]. The variance between populations represents the distance between each sample mean and the overall population mean. The variance within populations refers to the spread of each distribution. The F statistic is calculated by:

$$\mathrm{F} = \frac{\text{Between groups variance (MSC)}}{\text{Within groups variance (MSE)}} \tag{3.9}$$

This means that if the variance between groups, nominator, is relatively larger than the variance within groups, denominator, then the ratio will be much larger than one. Therefore, the samples most likely do not come from the same population, i.e. reject the Null hypothesis.

There are two main types of ANOVA: one-way ANOVA, also called one-factor ANOVA, and two-way ANOVA, also called two-factor ANOVA [201]. One-way ANOVA is considered the simplest form and it involves only a single factor (variable) in the experiment. When two variables are included, two-way ANOVA is used. This study only considers one-way ANOVA and then MANOVA is used for more than one variable.

In one-way ANOVA, the overall sum of squares is computed using the following equation:

$$\text{Total sum of squares (SST)} = \text{Between sum of squares (SSC)} + \text{Within sum of squares (SSE)} \tag{3.10}$$

To calculate the sum of squares, the mean of all responses, regardless of the group is found by:

$$\bar{X} = \frac{\sum_{ij} x_{ij}}{n} \tag{3.11}$$

Moreover, the sample mean of responses from the i^{th} group is found by:

$$\bar{X}_i = \frac{\sum_{j=1}^{n_i} x_{ij}}{n_i} \quad (3.12)$$

The three sums of squares (in Equation 3.10) are computed as follows:

$$SST = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \bar{X})^2 \quad (3.13)$$

$$SSC = \sum_{i=1}^{k} n_i \, (\bar{X}_i - \bar{X})^2 \quad (3.14)$$

$$SSE = SST - SSC \quad (3.15)$$

Where k is the number of groups or populations, n is the number of samples, n_i is the sample size of group i, x_{ij} is the j^{th} sample from the i^{th} group or population and $\bar{X}$, $\bar{X}_i$ are defined in Equations 3.11 and 3.12, respectively.

The mean square for each of the above is the division between the sum of squares and the degree of freedom as follows:

Total mean square (MST) = $\frac{SST}{n-1}$ (3.16)

Between mean square (MSC) = $\frac{SSC}{k-1}$ (3.17)

Within mean square (MSE) = $\frac{SSE}{n-k}$ (3.18)

Where k is the number of groups or populations, n is the number of samples.

Equations 3.17 and 3.18 are used to calculate the F-ration as illustrated in Equation 3.9. The F-ratio is used together with degree of freedoms to determine the p-value and, thus, either reject or fail to reject the Null hypothesis.

ANOVA has three main assumptions: independence, normality and homoscedasticity [201]. First, the observations are obtained independently and randomly from the population. Second, the population at each group is approximately normally distributed. Lastly, the different groups have common variance.

3.3.3.2.2 MANOVA

MANOVA, as previously mentioned, is an analysis of variance in which there is more than one dependent variable. The logic of MANOVA follows that of ANOVA but the calculations include matrix algebra. The independent variables are analysed in combination to provide composite variables to test for the effect of the dependent variables [202]. Moreover, an

assumption of homogeneity of variance and covariance matrices is also applied in MANOVA test statistics in addition to all ANOVA assumptions [203].

In the MANOVA, the between-groups sum of squares is generalized to an H matrix that contains explained covariance related to the hypothesis in hand [202]. The within-group sum of squares is generalized to the E matrix that contains unexplained covariance credited to the error. The sum of these two matrices is the T matrix which is the generalization of the total sum of squares.

To calculate these matrices: there is m random vectors $X_1, \ldots, X_m$ (representing groups). Each X_j is a k × 1 column vector of form $\begin{bmatrix} x_{j1} \\ \cdots \\ x_{jk} \end{bmatrix}$ where each x_{jp} is a random variable.

For each random vector X_j a sample $\{X_{ij}, \ldots X_{njj}\}$ of size n_j is collected. N is defined as $n = \sum_{j=1}^{m} n_j$. Each sample X_{ij} is a k × 1 vector of form $\begin{bmatrix} x_{ij1} \\ \cdots \\ x_{ijk} \end{bmatrix}$ where each x_{ijp} is a data element not a random variable, where index i refers to the subject in the experiment ($1 \leq i \leq n_j$), index j refers to the group ($1 \leq j \leq m$) and index p refers to the position within the random vector ($1 \leq p \leq k$)

The definition of the various means is similar to that in the univariate case, except that these means become k × 1 vectors. The total (or grand) mean vector is the column vector and it is computed as:

$$\overline{X}_T = \begin{bmatrix} \bar{x}_1 \\ \ldots \\ \bar{x}_k \end{bmatrix} \text{ Where } \quad \bar{x}_p = \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n_j} x_{ijp} \tag{3.19}$$

The group mean vector for group j is a column vector and it is calculated as:

$$\overline{X}_j = \begin{bmatrix} \bar{x}_{j1} \\ \ldots \\ \bar{x}_{jk} \end{bmatrix} \text{ Where } \quad \bar{x}_{jp} = \frac{1}{n_j} \sum_{i=1}^{n_j} x_{ijp} \tag{3.20}$$

The H matrix is calculated as:

$$H = \sum_{j=1}^{m} n_j (\overline{X}_j - \overline{X}_T)(\overline{X}_j - \overline{X}_T)^T \tag{3.21}$$

Where: $\overline{X}_T$, $\overline{X}_j$ are defined in Equation 3.19 and 3.20, respectively. N is the number of samples and j is the group.

The E matrix is computed as:

$$E = \sum_{j=1}^{m} \sum_{i=1}^{n_j} (X_{ij} - \overline{X}_j)(X_{ij} - \overline{X}_j)^T \quad (3.22)$$

Where: $\overline{X}_j$ is defined in Equation 3.20, i is the subject , j is the group.

The matrix T is computed as:

$$T = H + E \quad (3.23)$$

Many statistics have been used to compute MANOVA, such as Wilks' Lambda, Hotelling's Trace, Roy's Largest Root and Pillai's Trace [202].

Wilks' Λ (W) is the most commonly used test for MANOVA [204], and therefore used in this study. It is the ratio of determinants for the E and T matrices and it is computed as:

$$W = \frac{|E|}{|T|} = \frac{|E|}{|H+E|} \quad (3.24)$$

Where |E|, |H| and |T| are the determinant of matrixes E, H and T, respectively.

From the above equation, as the explained covariation (H) increases, the error variation (E) reduces and the Wilks' Lambda statistic becomes smaller. Therefore, the samples most likely come from the same population, i.e. accept the Null hypothesis.

Pillai's Trace test (V) is also commonly used for MANOVA [204]. It is computed from the first(s) (non-zero) eigenvalues of the H matrix multiplied by the inverse of the T matrix, as follows:

$$V = H T^{-1} \quad (3.25)$$

Some of the early work done in keystroke dynamics used MANOVA to prove that there is significant variability with which typists produce digraphs [205, 206]. Moreover, MANOVA is used in recent keystroke dynamics studies to find a subset of an original feature set that has the biggest separation between individuals. For example, in the study conducted in [207], MANOVA was used to select a feature subset from the data that consisted of the participants

entering a fixed sentence several times. The nearest neighbour algorithm is, then, used to classify the data.

3.3.4 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) were introduced as a machine learning method in 1995 by Cortes and Vapnik [208]. SVMs's main goal is to project the data points of a two-class training set in a higher dimensional space (if needed) and try to find a maximum-margin separating hyperplane between the data points of two classes. This hyperplane is considered optimal since it generalizes well to unseen data [209]. The optimal separating hyperplane is found by maximizing the margin between the closest points in the two classes. The points lying on the boundaries of the margin are called support vectors and the middle of the margin is the optimal separating hyperplane, as seen in Figure 3.12.

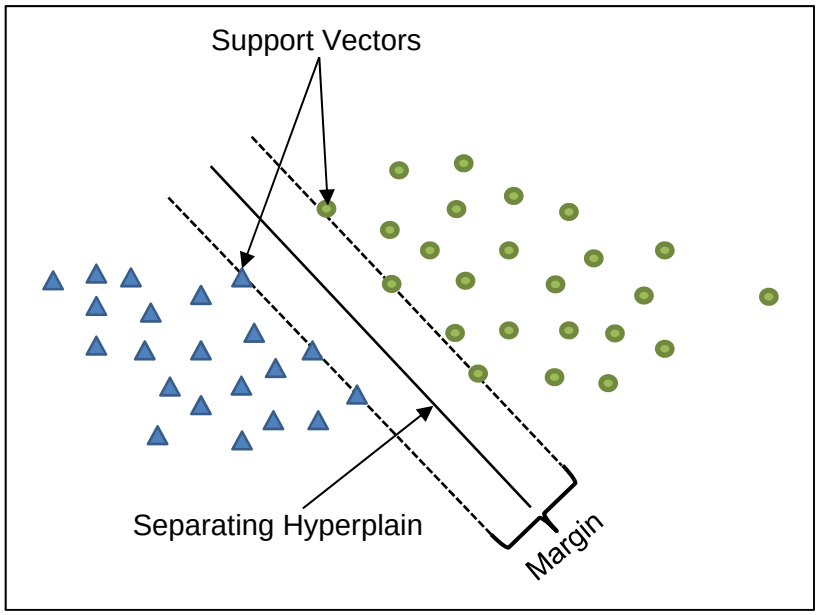

Figure 3.12: Optimal hyperplane between two classes.

This is true for linearly separable training sets for which there exists a linear discriminant function whose sign matches the class of all training examples [209]. There is usually an infinite number of separating hyperplanes when a training set is linearly separable. Choosing the separating hyperplane is done by selecting the one that maximizes the margin by leaving as much space as possible between the hyperplane and the closest data points.

To find the optimum hyperplane for L training points, in which x_i are the inputs of dimensionality D and each point is from one of two classes y_i = -1 or +1. The training data is formed as:

$$\{x_i, y_i\} \quad \text{where} \quad i = 1 \dots L, y_i \in \{-1,1\}, x \in R^D \tag{3.26}$$

The line that can be drawn on the graph to separate the classes, i.e. the hyperplane is defined by:

$$w \cdot x + b = 0 \tag{3.27}$$

Where w is normal to the hyperplane and $\frac{b}{||w||}$ is the perpendicular distance from the hyperplane to the origin.

SVM selects the variables w and b so that the training data is defined by:

$$x_i \cdot w + b \geq +1 \quad \text{for } y_i = +1 \tag{3.28}$$

$$x_i \cdot w + b \leq -1 \quad \text{for } y_i = -1 \tag{3.29}$$

These two equations can be combined into:

$$y_i (x_i \cdot w + b) - 1 \geq 0 \tag{3.30}$$

Finding the optimum hyperplane can be done by optimizing the following:

$$\min \frac{1}{2} ||w||^2 \quad \text{such that} \quad y_i(x_i \cdot w + b) - 1 \geq 0 \; \forall_i \tag{3.31}$$

A full description how to find the value of w and b needed to find the optimal separating hyperplane is found in [210].

The same principle of the SVMs is then stretched to not-fully-linearly-separable data. In the case that there are data points on the wrong side of the discriminant margin, it is called a soft margin. This soft margin SVMs introduces the idea of relaxed variables and the trade-off between maximizing the margin and minimizing the number of misclassified variables [210].

In the case of not-fully-linearly-separable data, the relaxed hyperplane can be found by optimizing the following:

$$\min \frac{1}{2} ||w||^2 + C \sum_{i=1}^{L} \xi_i \quad \text{such that} \quad y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0 \; \forall_i \tag{3.32}$$

Where ξ_i is a positive slack variable and C is a parameter for controlling the trade-off between the slack variable penalty and the size of the margin [210].

Moreover, SVMs are also applied to non-linearly separable data by the use of a family of functions called kernel functions. Kernel functions are based on calculating inner products of two vectors [211]. As a function can be modified to a higher dimensionality space by possibly some non-linear feature mapping function $x \rightarrow \emptyset(x)$, only inner products of the mapped inputs in the feature space have to be identified without requiring to explicitly calculate $\emptyset$.

In order to use an SVM to solve a linearly separable, binary classification problem, H is created to find the values of w and b, H is generated as follows:

$$H_{ij} = y_i y_j \, x_i \cdot x_j \tag{3.33}$$

This can be done in case of non-linearly separable data by applying a Kernel function and thus the mapping $x \rightarrow \emptyset(x)$ as follows:

$$H_{ij} = y_i y_j \, \emptyset(x_i) \cdot \emptyset(x_j) = y_i y_j k(x_i, x_j) \tag{3.34}$$

Where $k(x_i, x_j)$ is the Kernel function.

The use of Kernel functions is very important as many classification problems have non-linearly separable data points in the space of the inputs, however, it can be separable in a higher dimensionality feature space using a suitable mapping [211].

For example, looking at the data set in Figure 3.13 (A), it can be clearly noted that the data set is not linearly separable in the two dimensional data space. On the other hand, in Figure 3.13 (B) the data set is easily separable in the high dimensional feature space defined implicitly by a kernel function called Radial Basis Kernel (RBF), which is calculated using the following formula:

$$k(x_i, x_j) = e^{-\left(\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)} \tag{3.35}$$

Where $\|x_i - x_j\|^2$ is the squared Euclidean distance between the two feature vectors x_i and x_j and σ is a free parameter.

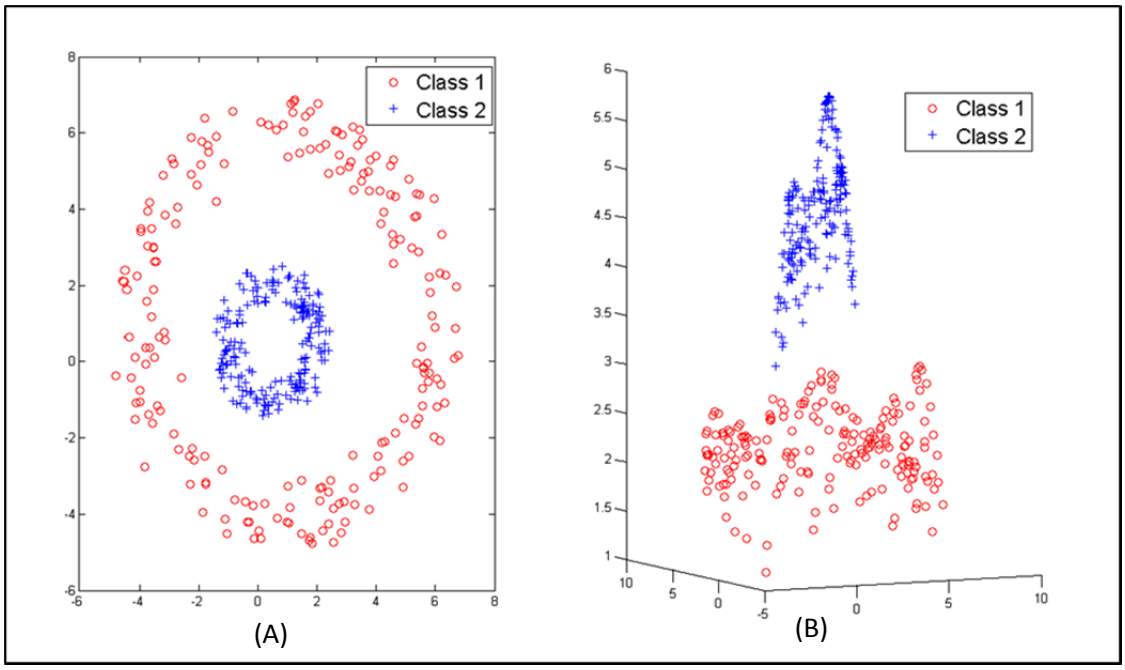

Figure 3.13: Non-linear data re-mapped using Radial Basis Kernel (source [210]).

Some other kernels often used for classification are the Linear Kernel, Polynomial Kernel and the Sigmoidal Kernel, they are defined in Equations 3.36, 3.37 and 3.38 respectively.

$$k(x_i, x_j) = x_i \cdot x_j \tag{3.36}$$

$$k(x_i, x_j) = (x_i \cdot x_j + a)^b \tag{3.37}$$

$$k(x_i, x_j) = \tanh(a\, x_i \cdot x_j - b) \tag{3.38}$$

Where x_i and x_j are vectors and a, b are kernel's behaviour-defining parameters.

Interestingly, these kernels produce different separating boundaries. In fact, the Radial Basis Kernel Function succeeds in separating more complicated datasets, while the Polynomial kernel fails to [212]. Therefore, the choice of kernel for a certain problem is very crucial. However, using more complex hypotheses usually comes with the inevitable cost of increased training time, as these complex kernels need to perform more calculations [211].

Although SVMs are binary classifiers, their success in real-world applications motivated researchers to investigate extending them to multiclass problems [212]. Based on the research conducted by Hsu and Lin [213], the methods for extending SVMs to solve multiclass problems have been divided into two categories. The first category considers the whole

dataset with all the classes at once and solves the multiclass problem directly (direct method), while the second category partitions the problem by constructing several binary classifiers and combining their outputs (indirect method).

In the direct method, an attempt to find the separating boundaries for all the classes is undertaken in one step. For example, Vapnik [214] attempts to define a decision rule similar to that of the binary SVMs for each class. And then, during testing, the data point is assigned the label of the decision rule that yielded the highest (positive) margin.

Nonetheless, in direct methods, a large number of variables need to be optimized which poses serious numerical difficulties. Due to that, approaches that decompose the problem into binary classification have been more preferable to researchers, i.e. indirect methods [213]. The first method for this category is the one-vs-rest method [214], also referred to as one-against-all, in which a binary SVMs classifier is constructed for each class. This is performed by segregating the data points of that class from the rest. Testing is very similar to that of the direct approaches explained previously. Furthermore, each classifier produces a decision value for the test data point and the classifier with the highest positive decision value assigns its label to the data point.

The other method of this category for extending SVMs to solve multiclass problems is the one-vs-one method, also referred to as one-against-one. In this method, there are $N(N+1)/2$ classifiers built for N classes, one for each pair of classes [210]. There are two schemes suggested and tested to obtain a classification from such a group of classifiers: the Max Wins algorithm proposed by Krebel [215] and DAGSVM introduced by Platt et al. [216]. The Max Wins algorithm implements voting among the classifiers and selects the label which is the one with the most votes among the classifiers, while the DAGSVM algorithm places the one-vs-one classifiers constructed on a Directed Acyclic Graph and derives the decision from it.

One-vs-one is proven to be faster than one-vs-rest in training [212]. This is due to the fact that while it builds more classifiers, each classifier is constructed faster because it includes only a portion of the dataset. In addition, comparing the direct methods and the indirect methods performance in [213] revealed that the direct methods are far slower to train.

A great deal of research has incorporated the use of SVMs in the field of keystroke dynamics. Two-class (i.e. binary) SVMs were used in research such as [52]. The binary classification used in two-class SVMs works by analysing the training samples belonging to the two

classes, and then assigning test samples into one class or the other [210]. Moreover, One-class SVMs were also used in [67] where the support vectors model is trained on data from one class only. The model can then identify the properties of that class and predict which test samples are not from that class [217].

Multi-class SVMs were also applied to keystroke data in a variety of means. One of which is the one-vs-rest approach, in which the researchers in [218] built eight different one-vs-rest SVMs classifiers, one for each volunteer. Each classifier was created by treating the data from one individual as positive and all data from other participants are considered negative.

Considering the nature of the keystroke dynamics as an authentication problem in this study, non-linear multiclass SVMs are used. Moreover, both one-vs-one and one-vs-rest multiclass classification are applied for comparative purposes.

3.3.5 Experimental Results and Discussion

This section points to the experiment results and discussion, in which the data collection, data space and the experimental results are indicated. A discussion about the results from this experiment and some comparisons with previous studies is performed in this section.

3.3.5.1 Data Collection

A total of twenty-five users participated in this study's experimentation, eight of which were involved in the original approach experiment. There were participants from both genders and all the participants were in the age group between 18 and 66 years old. They had different levels of typing skills and they were not all native speakers of English. Included were, for example, native Arabic, Czech and Malay speakers. Nonetheless, all participants were very familiar with English and were used to typing in English regularly. Only five users were located outside of the country where the research was conducted. Table 3.9 describes some of the demographic characteristics of the participants.

Table 3.9: Characteristics of the participants in the extended method experiment.

Gender		Age			Native language		Typing skills		
Male	**Female**	**18-37**	**38-57**	**58+**	**English**	**Non-English**	**Good**	**Moderate**	**Poor**
6	19	17	5	3	8	17	14	11	0

Similar to the original approach experiment, all participants were recruited by the researcher personally approaching them. Then, they are asked to read and sign a consent form that explains the aim of the research, the nature of the collected data and how it is treated, a copy of the consent form and the information sheet are found in Appendix A.1 and Appendix A.2, respectively. In addition, in this experiment, the users were also handed an instruction sheet that explains how to install, use and un-install the application on their machines. A copy of the instructions sheet can be found in Appendix A.3.

After three weeks a follow-up e-mail is sent to each participant to ensure that the participants are not facing any difficulties during data collection. Only two individuals agreed to participate in the experiment but later decided to withdraw. No reward was offered to the participants which might be the reason for the drop outs.

During data collection, the participants were asked to perform eight typing tasks in eight different sessions. Six of the tasks involved copying given text (copied-text) that consisted of around 900 characters each. The text included excerpts from the Guardian newspaper and it combined both short and well-known English words with other complex and difficult words. The text included both upper and lower case letters in addition to numbers and punctuation marks.

In addition, the last two typing tasks allowed users to free type around 900 characters of any text that they wished to type (un-copied or free-typing text), for example: previous holiday experience, future plans, opinion about a current issue … etc. Although the first six tasks included text that was chosen for the users to type, it is still considered free-text as the text used for training was different from that used for testing. Therefore, based on the definition of free-text [8], all text used in this study was free-text but with a different method for sourcing the text. In fact, the results produced by the experiments carried-out in [130] illustrates that using either free or copied text has no effect on the results of free-text keystroke systems. However, copied text was provided for the participants here to ease the process of data collection.

Moreover, copy-text was chosen to be used in this experiment to make the text consistent among users. This makes the feature extraction process among participants uniform as they all typed the same text. Yet, un-copied text was also collected in order to compare between the two entry modes, i.e. copied and un-copied text (as discussed in the future work section).

This collected text was the only data used for training and testing the system. Authenticating users based on as little as possible information, i.e. as little training data as possible, is a very important feature in this study in order to achieve the maximum level of user comfort.

Users were directed to enter the samples in the most natural way possible, i.e. the same way they usually follow when typing. As participants exhibited different typing skills, the typing time for each task varied among volunteers between 5 minutes and 10 minutes per task. Moreover, users were allowed to enter carriage returns and backspaces if needed. Furthermore, the data was acquired in different sessions as the users were requested to complete each of the eight tasks in a separate session. These sessions were spread over around four weeks.

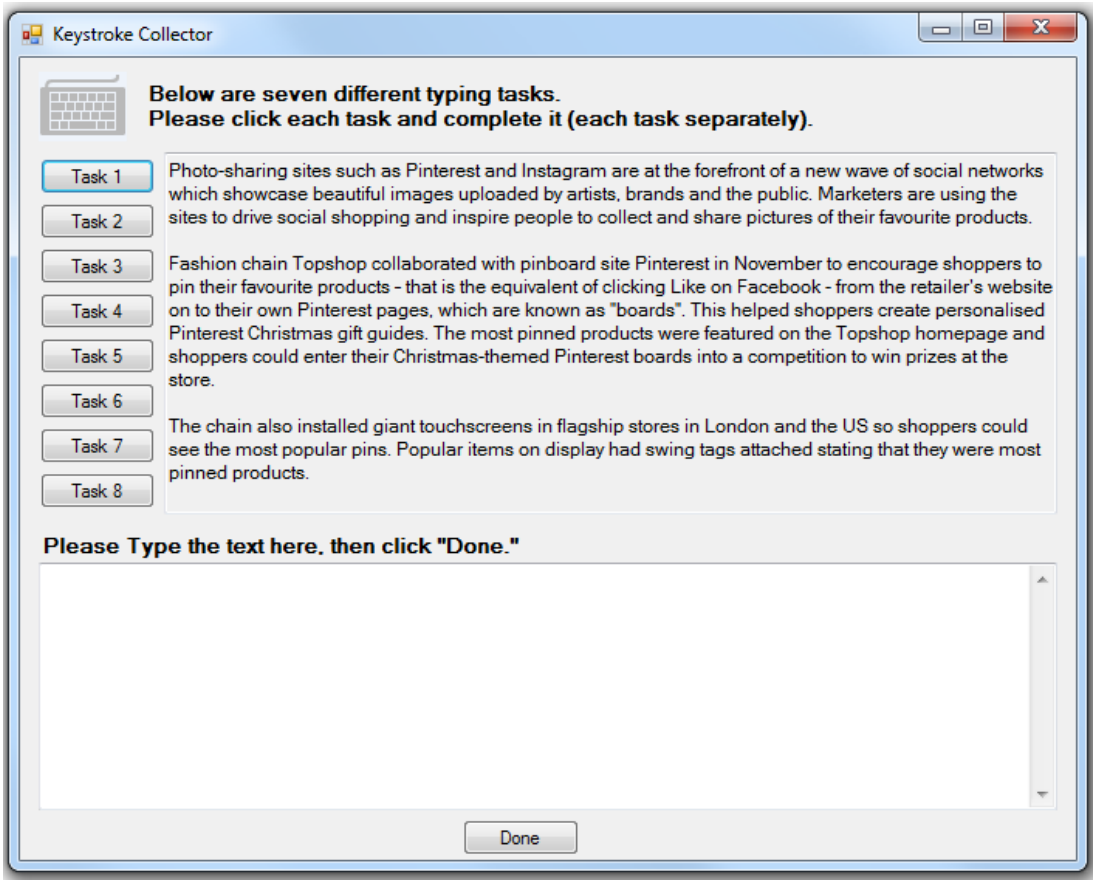

Figure 3.14: A screen-shot of the data collection program for the extended approach.

The data collection was performed on a GUI program implemented using the C++ language, the code for the data collection program is found in Appendix B.1. The application was downloaded on the users' personal machines to maximize their comfort as they are more

familiar with their own machine and its surroundings. Therefore, they were able to feel more at ease, and thus, they will perform the typing tasks in a manner closer to that of their real typing behaviour. Figure 3.14 is a screenshot of the data collection program showing the text appearing when the first typing task is clicked.

Moreover, the data acquisition process was not controlled due to the feedback from the volunteers who were involved in the original approach experiment, in which the users expressed their unease about having to type on a machine different to their own. Different machines have keyboards that vary in key size and spacing between keys. Moreover, keyboards in different machines have different key pressing sensitivity levels [4]. In addition, volunteers have also conveyed their discomfort about having to perform the typing tasks in a specific time and place as their anxiety levels elevated when they were asked to type on a machine located in a laboratory that they were not familiar with.

Similar to the original method, the data collection is done by capturing specific attributes for every key action performed on the keyboard. A complete description of the data collected for every action performed on the keyboard is provided in Section 3.2.2.1.

3.3.5.2 Data Space

Although there were 75 timing features captured from each user's typing stream, there were not enough instances that appeared in the used text for some of the key-pairs which made it not feasible to include them in the final feature set. The key-pairs with less than 10 instances are: Adjacent/DifferentSide, FourthAdjacent/RightSide, NonAdjacent/RightSide and NonAdjacent/LeftSide.

Indeed, this was the reason for excluding four of the key-pair categories from the study. Therefore 20 timing features were excluded from the final feature set. Table 3.10 lists the final 55 feature sets extracted from all key-pairs.

The pre-processing step involved creating the timing vector and storing it in the database as the user's profile. This process was carried-out by dividing the data into 20 equal sections each of which represented a different typing sample.

This was done by extracting the feature vector (which included all the instances of that feature) for each of the 55 features from each typing task separately (there were eight typing tasks as mentioned before). Then, each feature vector from all the tasks are concatenated to

produce 55 large vectors, one for each feature. These large vectors were stored in a feature matrix for each user.

Table 3.10: Final feature set for the extended approach.

Key-pair Category	Feature Set				
Adjacent/RightSide	AR-H1	AR-H2	AR-DD	AR-UU	AR-UD
Adjacent/LeftSide	AL-H1	AL-H2	AL-DD	AL-UU	AL-UD
SecondAdjacent/RightSide	SR-H1	SR-H2	SR-DD	SR-UU	SR-UD
SecondAdjacent/LeftSide	SL-H1	SL-H2	SL-DD	SL-UU	SL-UD
SecondAdjacent/DifferentSide	SD-H1	SD-H2	SD-DD	SD-UU	SD-UD
ThirdAdjacent/RightSide	TR-H1	TR-H2	TR-DD	TR-UU	TR-UD
ThirdAdjacent/LeftSide	TL-H1	TL-H2	TL-DD	TL-UU	TL-UD
ThirdAdjacent/DifferentSide	TD-H1	TD-H2	TD-DD	TD-UU	TD-UD
FourthAdjacent/LeftSide	FL-H1	FL-H2	FL-DD	FL-UU	FL-UD
FourthAdjacent/DifferentSide	FD-H1	FD-H2	FD-DD	FD-UU	FD-UD
NonAdjacent/DifferentSide	ND-H1	ND-H2	ND-DD	ND-UU	ND-UD

Similar to the original approach, outlier elimination and data scaling was performed exactly the same as described in Section 3.2.2.2. Around 0.05% of the overall data was considered outliers and therefore discarded from the dataset used for the experiment; this small amount of discarded data does not vastly affect the final dataset.

After that, each of the large vectors was divided equally into 20 parts. The size of each part, among different features, varied depending on the number of times the key-pair associated with that timing feature appeared in the text. Although the number of each key-pair appearance may vary, it is fairly similar between participants as they all typed similar text. The mean of each feature among these 20 divisions is computed and then stored in the corresponding user's timing vector (V). There are twenty timing vectors (Vs) for each user which were employed as the user's typing samples.

Creating the timing vector (V) for a single sample and storing it in the database of a user is the same as the original method. Yet, the vectors in the extended technique consist of eleven

sub-vectors instead of five sub-vectors in the case of the original scheme. It is eleven sub-vectors rather than fifteen due to four key-pairs not being included in the final feature set as they had insufficient number of appearances in the text as discussed earlier in this section. Thus, the following equations show a timing vector that describes a user's single typing sample:

$$V= \{V_{AR}, V_{AL}, V_{SR}, V_{SL}, V_{SD}, V_{TR}, V_{TL}, V_{TD}, V_{FL}, V_{FD}, V_{ND}\} \quad (3.39)$$

$$V_{AR}= \{\mu_{AR\text{-}H1}, \mu_{AR\text{-}H2}, \mu_{AR\text{-}DD}, \mu_{AR\text{-}UU}, \mu_{AR\text{-}UD}\} \quad (3.40)$$

⋮

$$V_{ND} = \{\mu_{ND\text{-}H1}, \mu_{ND\text{-}H2}, \mu_{ND\text{-}DD}, \mu_{ND\text{-}UU}, \mu_{ND\text{-}UD}\} \quad (3.41)$$

Where:

$\mu_{AR\text{-}H1}$: denotes the mean of the H_1 timing feature for all adjacent key-pairs on the right side of the keyboard in that sample.

$\mu_{AR\text{-}H2}$: denotes the mean of the H_2 timing feature for all adjacent key-pairs on the right side of the keyboard in that sample.

… and so on.

The code for performing outlier discarding, data scaling and timing vectors formation is presented in Appendix B.2.

The data base consisted of twenty-five users' profiles. Each user's profile included twenty vectors similar to the one in Equation 3.39 representing each of the twenty samples for each individual.

The 20 typing samples extracted from each participant were partitioned into 15 samples for training the system and 5 for testing. There was no particular rule for choosing these samples, thus, the first 15 samples were designated to be the training samples and the last five were selected to be the testing samples. As mentioned earlier, the data used for training was acquired in sessions different from that used for testing.

3.3.5.3 *Experiment and Results*

After the key-pair based feature extraction is complete, feature selection is performed using ACO and MANOVA. The selected feature set is then fed into the SVMs classifier where the training and testing procedures are done. Figure 3.15 illustrates the framework followed in this study.

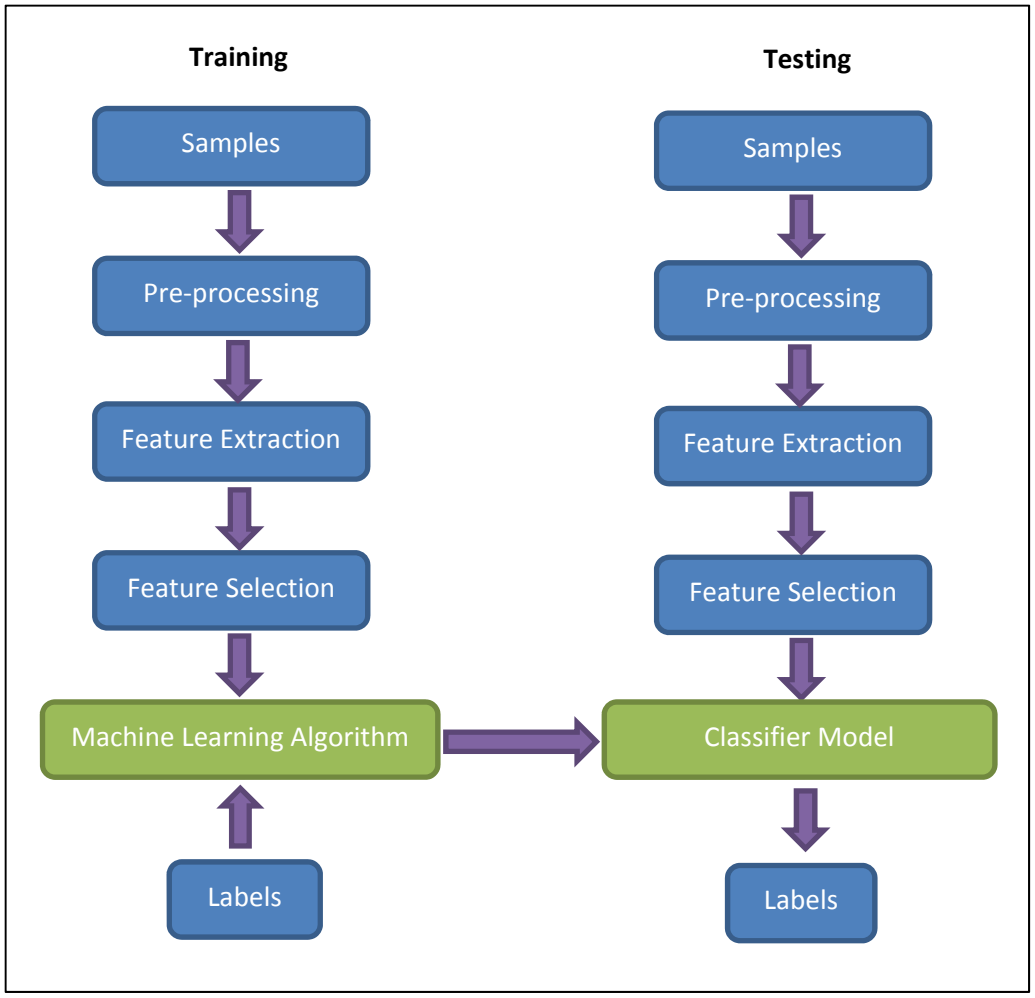

Figure 3.15: Framework for the keystroke system.

The data was analysed in two different ways to select the best features representing human typing behaviour: a wrapper and a filter. First of which, the ACO technique was chosen over other wrapper feature selection techniques such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) based on the study performed in [182]. This study concluded that using ACO with the Extreme Learning Machine (ELM) classifier yielded the best results compared with the two other previously mentioned techniques. A similar conclusion was produced in another study where a Back Propagation Neural Network (BPNN) was used as a

classifier [5]. ACO again proved its superiority over the other feature selection techniques in this study. Furthermore, the technology behind ACO fits into this research as the classification performance of each of the keystroke features is used to evaluate the quality of that feature and then influence the ant's choice of which feature to use for constructing its final features subset [219].

Moreover, a total of 55 ants, which is equal to the number of features used, were generated in each of the 100 iterations performed in the ACO system [197]. In addition, an initial pheromone of 1.0 was used and as the importance of the heuristic information, i.e. classification rate, was more than that of the pheromone level in the transition rule, it has been decided to used α=1.0 and β=0.1 [197]. All the programing for the ACO procedure, illustrated in Figure 3.11, was performed on MATLAB, the code for ACO is found in Appendix B.3.

Nonetheless, MANOVA was also chosen as a statistical based filter feature subset selection method over other similar methods. Although dimensional reduction techniques such as Principal Component Analysis (PCA) permit finding a lower dimension transformed space based on data variance, they do not take into account any information about the separability between classes. The direction of highest variance does not always correspond to the direction of highest separability [220]. This is because PCA doesn't consider the grouping class variable [221]. Moreover, MANOVA studies the interactions among independent variables more closely, thus it allows for better understanding of how different features can work together to produce better discrimination between users [203].

The algorithm followed to select a feature subset using MANOVA was similar to the progressive algorithm used in the study conducted in [220]. MANOVA is utilized to find a feature subset from the original feature set that has the biggest separation between individuals. This is done gradually by adding the feature with the highest separation to the subset first, and then, finding the combination of features that includes that feature and produce the highest separation, and so on.

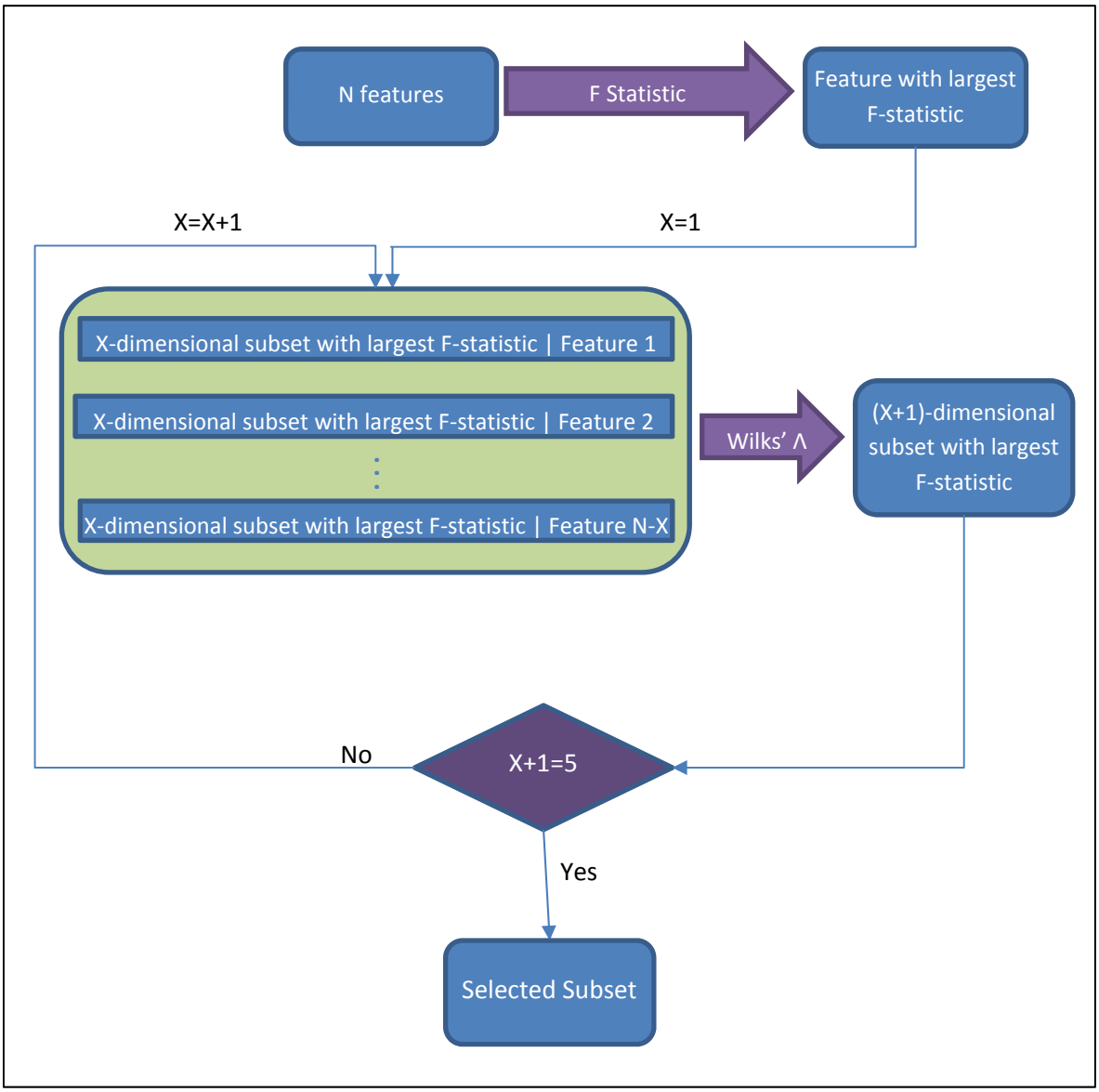

Figure 3.16: Algorithm for applying MANOVA.

The algorithm used to apply MANOVA progresses as the following (as illustrated in Figure 3.16):

1. The F-statistic is calculated, i.e. ANOVA test, for one-feature subsets and the feature with the largest F-statistic is selected.
2. Two-dimensional subsets, combining the feature selected in the first step with the remaining features, are constructed.
3. Each of these subsets is evaluated using the MANOVA's Wilks' Λ test, and its corresponding F-statistic is calculated.
4. The 2-dimensional subset with the largest F-statistic is chosen.

5. Three-dimensional feature subsets are constructed by combining the remaining features with the previously selected subset.
6. This is repeated until the preferred number of features was reached.

The F-statistic and Wilks' Λ test were calculated using the SPSS Statistics software. Moreover, some features did not satisfy the test of homogeneity of variances and covariance [201]. Therefore, they were excluded from the search process. The excluded features are: AR-DD, AR-UD, AL-H1, SL-DD, SL-UD, TD-DD, TD-UD, FL-H1, FL-UU, FL-DD, FL-UD, FD-UD.

The number of selected features was set to be five in both ACO and MANOVA in order to compare and contrast between the selected feature sets without worrying about the feature set length. Moreover, the Curse of Dimensionality corresponds to the problem that the amount of training needed grows exponentially with the number of features [222]. Since the experiment only had 20 samples per person, it was necessary to cut down the features to the least amount possible while conserving the maximum benefit provided to the classification process. Therefore, selecting five features will eliminate the drawbacks of the Curse of Dimensionality whilst preserving the preferred small amount of data required for training.

Table 3.11: Selected features subset.

	Selected Features				
ACO	**SR-H2**	FL-H2	FD-H2	**ND-H1**	**ND-UD**
MANOVA	AL-H1	**SR-H2**	**ND-H1**	ND-DD	**ND-UD**

Table 3.11 lists the selected features subsets that were derived using both methods. It is clear that the hold time appears slightly more important compared with the other timing features as four out of the five selected features in the case of ACO and three of the five selected by MANOVA are hold times extracted from different key-pairs. This agrees with the conclusions produced by a number of studies conducted in the keystroke dynamics area such as the ones published in [107, 119].

Moreover, a variety of key-pairs was included in the selected features subsets. Nonetheless the NonAdjacent/DifferentSide (ND) key-pair was incorporated in two of the ACO selected features and three of the MANOVA selected features. This can be interestingly explained by

the brain's capability, that allows for unconscious typing consistency when typing a key pair that involves the use of two hands [223].

Another interesting aspect of the two feature subsets is that they are similar in three of the five selected features which corresponds to the unexpected similarity between the ACO and MANOVA methods, although they follow completely different procedures to select features. These features are: SecondAdjacent/RightSide-Hold2 (SR-H2), NonAdjacent/DifferentSide-Hold1 (ND-H1) and NonAdjacent/DifferentSide-UD (ND-UD); they appear in bold green in Table 3.11.

Support Vector Machines (SVMs) were chosen to be used in this experiment as it is one of the most successful classification techniques [209] in addition to its capability of combining low computational costs with adequate performance [224]. Specifically, it works with non-linear problems, such as the one presented by the keystrokes dynamics problem, by capturing much more complex relationships between the data points. This allows for less difficult transformations to be performed to achieve the desired classification [225]. Furthermore, due to the nature of this study and the aim of using the least amount of training possible, SVMs is used as it tends to resist over-fitting when its parameters are well-tuned [226].

Furthermore, a Radial Basis Function (RBF) kernel was used in this study as it succeeds in separating more complicated datasets compared with other kernels [212]. In addition, a large number of studies suggests the use of the RBF as it nonlinearly maps samples into a higher dimensional space when there is a complicated relationship between the class labels and its attributes which fits to the keystroke dynamics problem [161].

Moreover, as mentioned earlier, two multiclass classification SVMs techniques were used, namely: one-vs-one and one-vs-rest. Multiclass classification is required as SVMs are used to decide which test samples belong to which of the twenty-five individuals involved in the study. Performance comparison was carried-out between these techniques as shown later in this section.

The classification process was implemented on MATLAB with the aid of the LIBSVM library which is a popular open source machine learning library developed by Chang and Lin in 2000 [227]. The library was programed using both C and C++ languages for the purpose of helping users to easily apply SVMs to their applications. LIBSVM is considered

one of the most used SVMs software in machine learning and many other areas as there were more than 250,000 downloads of the package between the years 2000 and 2010 [227].

Prior to the use of LIBSVM for training and testing, the keystroke data has to be transformed to the LIBSVM package format. LIBSVM uses a format called "sparse" format; in which the zero values are not stored. Therefore, for example, the data 1 0 2 0 is represented as 1:1 3:2 [161]. The utilization of LIBSVM in user classification is shown in the code presented at Appendix B.4.

There are two parameters affecting RBF SVMs; namely: C and gamma [161]. The C parameter represents the trades off between the misclassification of training samples and the simplicity of the decision surface. Thus, choosing a small C allows for a smooth decision surface, whilst choosing a large C yields in correctly classifying all training samples by permitting the model to select more samples as support vectors. This will consequently result in a rough decision surface and cause over-fitting [228].

Additionally, gamma is the RBF freedom parameter. It signifies the influence of a single training example. Choosing a large gamma value causes the radius of the area of influence of the support vectors to only include the support vector itself whilst choosing a small gamma causes the region of influence of the support vector to incorporate the whole training set, resulting in a model behaving like a linear model [161].

Parameters C and gamma were chosen using grid search and cross-validation [161]. A range from 10^{-3} to 10^{3} was set for both C and gamma values as recommended by the research done in [161]. All combinations of C and gamma were tested using a 10-fold cross-validation. The main reason for using cross-validation was avoiding the over-fitting problem [161].

In 10-fold cross-validation, the training set is split into 10 equal-size-subsets. Each subset is tested using the classifier trained on the remaining nine subsets. The cross-validation accuracy of a C and gamma pair is denoted by the average accuracy of these 10 tests. The best performing value pair for C and gamma was chosen. The C & gamma pair chosen for this study was: C= 10^{1}, gamma=10^{0}.

It is worth mentioning here that LIBSVM uses the "one-vs-one" method for multi-class classification. The one-vs-one technique constructs one classifier per pair of classes and, then, the class which received the most votes is selected at prediction time. Therefore, it requires training of n * (n - 1) / 2 classifiers, where n is the number of classes. In addition to

that, the "one-vs-rest" method was also implemented on MATLAB. Twenty-five one-vs-rest SVMs classifiers were created, one for each volunteer. It involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives.

Two error rates were used to analyse the performance of the one-vs-one and one-vs-rest SVMs, namely: FAR and FRR [4]. These two error rates were considered in order to precisely measure the imposter pass rate using FAR and the genuine users' rejection rate using FRR. This will grant more understanding of the security provided by the system. The complete explanation of these error rates is found in Section 2.7. The FAR and FRR derived from the one-vs-one and one-vs-rest SVMs classification process are listed in Table 3.12. The complete results for each participant can be found in Appendix C.2.

Table 3.12: Classification performance for SVMs

	One-vs-One		***One-vs-Rest***	
	FAR	***FRR***	***FAR***	***FRR***
ACO	0.013	0.384	0.021	0.512
MANOVA	0.017	0.408	0.026	0.624

It can be established from these results that the features subset selected using ACO yields slightly better error rates. The ACO being a wrapper feature selection method looks into the interaction with the classifier which causes the feature subset space search to be closer to the hypothesis space search [170].

Moreover, the one-vs-one method produced, in this study, lower error rates when compared with the one-vs-rest method. The reason for one-vs-one performing better here might be due to the fact that it employs voting between all binary classifiers prediction. This is not the case in one-vs-rest in which all binary classifiers must have correct predictions to produce an overall correct decision [212]. However, the performance of the two multiclass classification approaches is case-based as the two methods perform differently for different problems [229, 230]. Yet the superiority of the one-vs-one SVMs classification was also recognized in [231]. In addition, one-vs-one is used to speed up the decision making process as it requires less time for classifying samples compared with one-vs-rest [229].

It can also be remarked that the FAR is considerably lower, i.e. better than the FRR. This corresponds to the system being largely immune against imposters' access. On the other

hand, the FRR is not considered an accepted value because the system denies access to a considerable amount of legitimate users tries.

Biased results are noticeable here due to the fact that the positive samples are much less than the negative samples. This phenomena normally happens in unbalanced datasets similar to the one used in this study [232]. This occurs as the number of support vectors in the positive class is less than in the negative class, which causes samples from the positive class to reside farther from the real decision boundary compared with those from the negative class. Consequently, the predicted decision boundary is pushed towards the infrequent class [233] which causes the large difference between the FAR and FRR figures as new data points are classified more frequently as the class with the larger number of samples, i.e. negative class. Solutions for this bias problem are introduced in the "future work" section.

3.3.5.4 Discussion

The extended version of the previously introduced original key-pairing scheme has clearly improved the authentication performance. Using five timing features in the original key-pairing system yielded in a 0.033 FAR and 0.867 FRR. This has been reduced in the extended version as the FAR was found to be 0.013 in the case of ACO's selected features subset and 0.017 in MANOVA's. Moreover, the FRR was found to be 0.512 in the case of ACO's selected features subset and 0.624 in MANOVA's. Even though, these are not satisfactory figures, the extended technique succeeded in achieving improved FAR and FRR. The advancement in both FAR and FRR has been achieved due to various reasons.

First, the timing features were extended to include the positioning of the hands on the keyboard which contributed to capturing the user's typing behaviour in a better fashion. Second, using an advanced feature selection such as ACO and MANOVA aided in the process of selecting the best features that best differentiate between individuals. Third, the use of a classification technique, i.e. SVMs, which is well known for its capabilities of differentiating between classes even in complex situations, have considerably improved the overall performance level; as opposed to the simple Euclidian distance approach employed in the original scheme study.

Fourth, the test and training samples in the extended technique are of the same length while the test samples are much smaller than the training samples in the original approach. The variation of text lengths between training and testing samples were proven to affect the error

rates negatively in the work done in [79]. Therefore, standardizing the text length in the extended technique assisted in improving the training and testing process. Lastly, the experimentation surroundings changed in the extended approach study to accommodate each user's preferences. As the data collection program was downloaded into the user's own machine, they were more familiar with their own keyboards and the environmental surroundings as opposed to being asked to type on a different machine which might have a different layout or different key resistance in addition to being located in an unusual environment.

Nonetheless, the produced FRR was not as significant as the FAR. This is expected in such a system where there is an unavoidable trade-off between the FAR and FRR. The critical part of such a highly secure system is boosting the number of detected imposters; which (unfortunately) might cause some legitimate users to be denied access too. Even though this might cause some inconvenience to the users, it will help to ensure the privacy and security of the system.

Furthermore, Figure 3.18 illustrates the error rate produced by the original method and the extended method in two cases: using MANOVA feature selection and using ACO feature selection. It is clear that the results from the original technique improved considerably using the extended approach. Furthermore, Error rates produced by ACO were slightly lower than those produced by MANOVA.

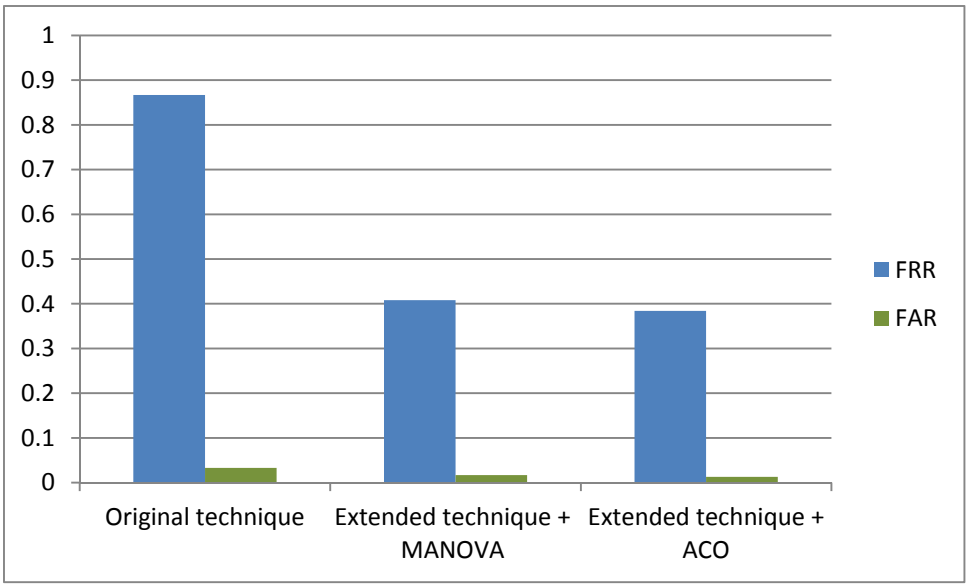

Figure 3.18: Comparison between error rates produced by different methods.

Comparing the results of this study with those from previous free-text studies showed that the FAR rates in this research are pretty similar, indeed in some cases even better, despite the requirement for far less training and the much more practical nature of the study. However, it is acknowledged that FRR rates are better in some other studies. Moreover, the same trade-off accrued between very good FAR rates and somewhat moderate FRR's.

An example of such studies is that conducted by Davoudi and Kabir in [117], where they attained a 0.0008 FAR and a 0.188 FRR using around 60 samples per person. This example shows the same trade-off between FAR and FRR. Furthermore, the FAR and FRR produced by this study is higher than that produced by Davoudi and Kabir, which does not involve any key-pairs. It must be stressed, however, that the method cited used much more training and therefore caused more imposing on the user. Davoudi and Kabir used input of around 11700-13500 characters while the key-pairing method made use of only 7200 characters long input.

Moreover, a key-pairing technique was introduced in the research conducted by Sing and Arya [50], in which key-pairs were classified based on their location on the keyboard. The keyboard was divided into 8 sections; two left and right halves and then each half was divided into 4 lines representing the rows of the keyboard. For example "wm" is represented as Left2-Right4. The overall performance reached 0.02 FAR and 0.04 FRR. The fact that it is a fixed-text study, in which a short password was used for login, have helped significantly in lowering the FRR compared with the study conducted in this research where free-text is used for training and testing.

The only research that was found in literature that employed a key-pairing technique slightly close to the one used in this research is the one conducted by Zahid et al. [46]. Four key-pairs were extracted in that study: horizontal digraph (i.e. adjacent keys in the same horizontal line), vertical digraph (i.e. adjacent keys in the same vertical line), non-adjacent horizontal digraph (i.e. non-adjacent keys in the same horizontal line), and non-adjacent vertical digraph (i.e. non-adjacent keys in the same vertical line). The best achieved FAR and FRR were 0.292 and 0.308, respectively. This study fails to capture the user's distinctive typing characteristics when typing key-pairs other than the ones utilized in the study. For example second-adjacent and third-adjacent key-pairs might have significant properties that can differ from the general non-adjacent keys. Furthermore, the key-pair classification introduced in that study ignores key-pairs that are diagonally adjacent. This study was conducted on smart phones which makes it difficult to compare with the technique applied in this study.

In addition, the studies conducted by Sing & Arya and Zahid et al., mentioned above, suffer from the fact that they only considered one timing feature in their experiments. The up-down time was the only feature used in the first study while the second study merely analysed the down-down timing feature. Extracting only one timing feature from each key-pair seems to cause a loss of some important information from the key-pair as it may ignore some of the significant aspects of the key-pair. Therefore, five different timing features varying between duration and latency times are used in this study in order to capture most of the substantial characteristics that each key-pair holds.

Although the key-pairing method was introduced in this research in order to reduce the amount of text used for training, it has not achieved a performance similar to that of some studies that don't incorporate key-pairs. The milestone study conducted by Gunetti and Picardi [8], for example, involved comparing two samples based on the duration of the n-graphs (i.e. di-graphs, tri-graphs and n-graphs) shared between samples. These n-graphs are specific two characters typed after each other with no concern about the key-pair group they belong to, i.e. example of some di-graphs: "em", "et", "as"… etc. A ground-breaking FAR of 0.00005 and FRR of 0.05 was produced. Yet using such n-graphs in free-text input is challenged by the need to collect the same n-graphs from training and testing samples to carry-out the comparison task. Therefore, it is hampered by using a large amount of text. In addition, the algorithm followed by Gunetti and Picardi is tailored for the English language (or languages with the same alphabet as English) and cannot be applied to languages with a different alphabet. This is due to the fact that it computes the similarity between two samples based on the timing of certain letter pairs in the language. In contrast, the key-pairing method, introduced here, computes the similarity between two samples based on the timing of key-pairs that were constructed based on the position of each of the two characters' keys not on the actual characters forming the key-pair.

Nonetheless, the key-pairing algorithm introduced here needs more polishing to achieve a system performance close to that of some non-key-pairing methods. This will be carried-out in the next section by introducing more non-conventional features that can be extracted from free-text keystrokes.

3.4 Summary

This chapter examines the effectiveness of using a novel method, based on the keyboard key-layout, for free-text keystroke dynamics authentication. The main reason for using key-pairs is to reduce the amount of training data delivered by the user at the enrolment phase because the need for huge training data is a crucial drawback of keystroke systems.

The initial experimentation examined an original version of the key-pairing method. The original method extracted typing features from key-pairs that have a relation on the keyboard's layout. Euclidian distance was used for classification in the experiment involving the original technique. It produced moderate results, yet considering the fact that it uses free-text for authentication, it achieved a good balance between the system's security and the user's comfort. Single features did not perform well on their own and better performance was obtained using a combination of more than one feature.

It is hardly surprising that the authentication performance is not perfect, given the nature of this experiment and the fact that its priority is the user's comfort. However, this study, which used only one short training sample, resulted in an error rate (in case of using a combination all five features) that was quite close to other studies that required many more training samples. Nevertheless, more sophisticated techniques were needed to further improve these error rates.

A number of modifications were applied to the original keyboard's key-layout based method. These modifications included more sophisticated key-pair formation which takes into consideration the position of the hand on the keyboard. The extended key-pairing approach works in each case by classifying every two characters typed consecutively based on their relation to each other and their overall location on the keyboard. For each key-pair, five timing features were extracted to be used in the user's features vector.

In an attempt to select features that best represent the user's typing behaviour, ACO proved to aid the features subset selection slightly more than the MANOVA mechanism. Moreover one-vs-one multiclass SVMs helped to reduce the number of misclassified samples more than one-vs-rest. The FAR rates were satisfactory as the number of imposters who were accepted as legitimate users were very few. On the other hand, the compromise between the FAR and FRR rates in the system was a clear reason for the moderate FRR rates.

The extended approach succeeded in improving the results produced by the original scheme. Yet it did not come at the cost of more training. User relaxation was still achieved as the least amount of text is used for authentication. Furthermore, additional features will be used in the next section in order to further improve the system performance.

Moreover, the key-pairing method is language-independent. This is due to the features being extracted not based on the actual characters of the typed di-graph but on the key-location of the each character in the typed di-graph on the keyboard's layout. It can compute the timing features extracted from the different key-pairs regardless of the language of the text. Therefore, it can be used to authenticate users typing in any language. This will be tested in a later section of this thesis.

Chapter 4

Non-Conventional Keystroke Features

4.1 Introduction

This chapter introduces an approach for user authentication using free-text keystroke dynamics which incorporates the use of non-conventional keystroke features. These features include semi-timing features along with editing features that were extracted from the users' typing stream.

Keystroke dynamics is conventionally based on timing features that compute time lapses between two actions on the keyboard such as key press and key release [4]. In this part of the thesis, however, the use of non-conventional keystroke features in the authentication of users is investigated. Features such as typing speed, error rate, and shift key usage are utilized to find typing patterns that can be used to distinguish between individuals.

Non-conventional features are considered during free-text input in a way that they are extracted from the whole piece of text. They represent the percentage of specific actions that are committed when typing a piece of text. This will not compromise the main objective which this thesis is attempting to achieve. This is because the text used to collect non-conventional features is of reasonable length and will not cause any more burden on the user compared with the key-pairing method discussed earlier. In fact, the same experiment was conducted to gather the data for both the key-pair method and the non-conventional features from the participants at the same time, as will be explained in Section 4.4.1.

The non-conventional features are important due to the limited measurements that conventional keystroke dynamics present. Conventional keystroke data, in a very different way to other biometrics (e.g. image processing), captures very little information [49]. This information consists of time lapses between actions performed on two keys (latency), and on one key (duration) [33].

To enlarge the amount of information that can be extracted from a user input and therefore assemble better indications about his or her typing behaviour, this part of the thesis focuses on non-conventional typing features. Non-conventional features can be extracted collectively during the whole text input, in which more information is available.

Most of the work done in the field of keystroke dynamics authentication focuses primarily on timing features while ignoring other typing behaviour such as input and editing patterns. Even previous studies that have included some non-timing features have not delivered the significance of these features in the way that they still focused on the importance of the conventional timing features, and fail to realize the potential of non-conventional features in the authentication process [70, 45]. For that reason, a high motivation is sustained in this research to explore the area of non-conventional typing features in order to concentrate on their distinctive ability to distinguish between individuals using little training. A comparison between conventional and non-conventional features is in place as well.

A total of nine non-conventional features were extracted from the users' typing stream. These features vary between semi-timing features and editing features. Moreover, an in depth study on the effect of using various non-conventional feature subset sizes has also been conducted.

Decision trees (DTs) were exploited to classify each of the users' data. In parallel for comparison, Support Vector Machines (SVMs) were also used for classification in association with the Ant Colony Optimization (ACO) feature selection technique.

4.2 Feature Definition

A great deal of the research done in the keystroke dynamics field has been focused mainly on the timing features extracted from the user's typing stream. These features compute the time lapses between performing two actions on the keyboard such as calculating the time it takes a person to press a certain key, i.e. the hold time [4]. Latency time is computed in a similar way but the two actions are performed on two different keys pressed successively rather than both actions being performed on one key as in the case of the hold time [19].

In this section, new features are explored. These non-conventional features avoid the conventional way of computing the time lapses between two actions on the keyboard. Instead, non-conventional features focus on the overall typing patterns that a user follows during input that extends over a period of time. It considers the percentage of performing

certain actions (in relation to the total number of actions), i.e. general typing actions or editing actions, which leads to understanding more aspects of the user's typing behaviour. A better perception of human typing patterns is particularly easier to capture while typing a whole piece of free-text in which more information can be extracted [49].

From the nine features exploited in this section, three were used in earlier research. The features used in previous research are WPM [70], error rate [80], capslock usage [49]. The remaining six features are original to this research as there is no prior use for them, in the same way reported in this research, in the available keystroke dynamics literature (at the time of writing this thesis). The decision to use non-conventional features (both novel features and ones used in previous research) was based on the fact that assembling a good indication about the user typing behaviour is enabled by the large amount of information available collectively during the whole text input. This is obtainable by using non-conventional features which are able to capture a variety of attributes that defines the users typing behaviour during the entire period of the typing process.

Furthermore, the power of the non-conventional features was experienced in the experiments conducted in Chapter 3 in which these features appeared to be having distinctive qualities that can be used to differentiate between users. Thus, the work done in this chapter is performed to verify these initial observations and provide evidence that non-conventional features can be used as a good verifier of a person's identity.

Two types of non-conventional typing features are considered: namely: semi-timing features and editing features. A brief description of each category is presented in this section as follows:

4.2.1 Semi-Timing Features

Features that have been extracted using some form of time calculation are incorporated in this research which is different from the standard timing features used in most of the literature. The time calculation followed in this category however, is slightly different from that of the regular timing features. Semi-timing features have a collective property to them, as they are calculated during the typing of a whole piece of text, i.e. paragraph(s).

The first feature is the word-per-minute (WPM) feature which, as the name suggests, measures the user's average typing speed [80]. The total typing time is calculated from the

very first key press until the very last key release and this is used in the final calculation of the WPM. The number of words are totalled and then divided by the total typing time in minutes; this is shown in Equation 4.1. Of course, this feature will easily distinguish between slow and fast typists. Nonetheless, it is not enough to find the difference between individuals who are close in typing speed [49].

$$\text{WPM} = \frac{\text{Number of words}}{\text{Total typing time in minutes}} \tag{4.1}$$

An interesting characteristic that can be found in some user's typing behaviour is the number of negative up-down (negUD) actions detected in their typing stream. The negative up-down is due to an overlap happening between two successive keys being typed. This particular typing behaviour is found in the typing stream of users who have the tendency to press the second key before releasing the first one. While most timing features are always positive because they represent the sequence determining the keyboard output, the up-down feature can be negative in some cases that might involve fast typists [49].

Figure 4.1 illustrates two different two-key sequences showing the up-down time in a non-overlapping situation and in an overlapping one. A keystroke is represented as a horizontal line with the down arrow marking the press and the up arrow indicating the release time. In part (A), a positive up-down time was produced from non-overlapping keystroke events and in part (B), a negative up-down time was produced from overlapping keystroke events where the first key was released after the second was pressed.

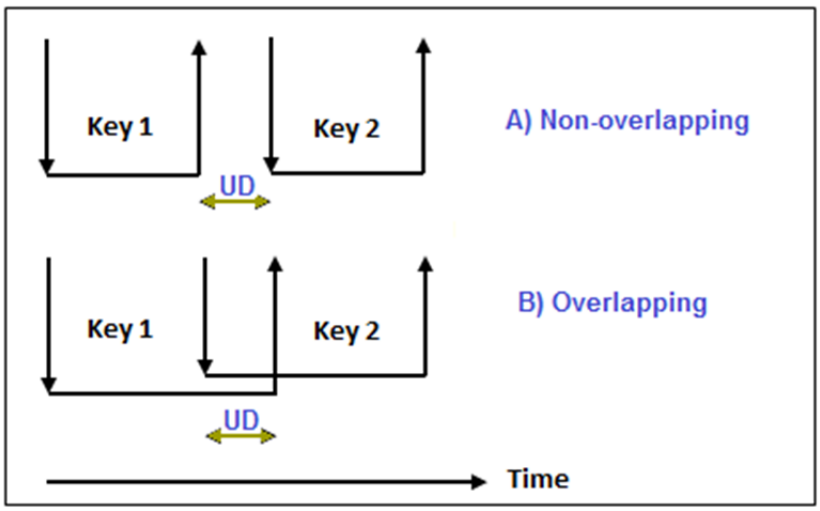

Figure 4.1: Negative UD caused by overlapping keystroke events.

Some studies found it challenging to deal with negative UD time [234]. Here it is used to the advantage of the research by finding the percentage of negative up-down instances for each user. As mentioned in [136], a negative value of UD implies time reduction or faster pressing while positive values imply time addition or slower pressing between two sequences of keystrokes. It has been found in the experimentation undertaken in this section that some users have absolutely no negative UDs whilst others have a fair amount, which was consistent in all the typing tasks they produced. This gives a good indication that comparing the percentage of negative UDs can be a good method to assist in user recognition. NegUD is computed as the percentage of the number of negative UD appearances and the total number of key-pairs, i.e. two keys typed consecutively. This is shown in the following equation:

$$\text{NegUD} = \frac{\text{Number of negative UDs}}{\text{Total number of keypairs}} \tag{4.2}$$

A very similar typing behaviour that has rarely been referred to in the literature up-to the time of writing this thesis, is the negative up-up (negUU) time, which occurs when the typist tends to release the second key before releasing the first key. This characteristic happened with a few of the volunteers who participated in the data collection. Moreover, a negative UU only happens when there is a negative UD between the two successive keys. However, if there happens to be a negative UD, this does not mean that there is definitely a negative UU as shown in Figure 4.2.

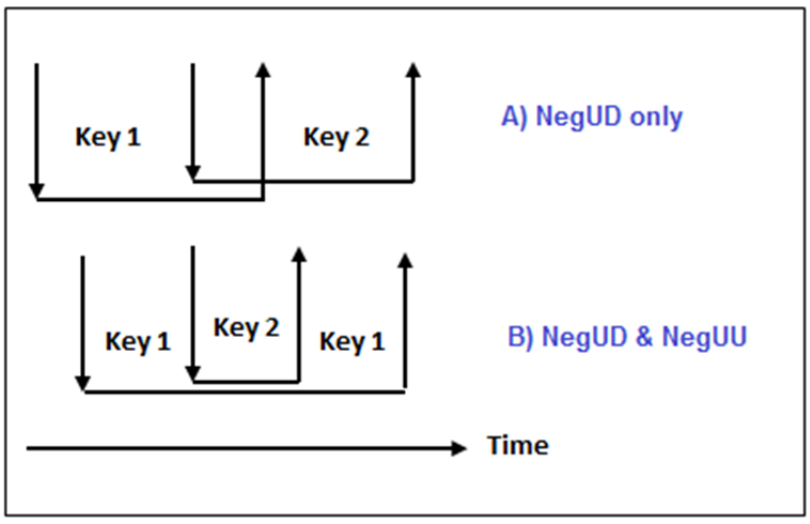

Figure 4.2: Cases of negative UD only and negative UD and negative UU.

Having said that, a negative UU has the property of occurring less frequently, but if it does, there is a high possibility that it is a specific characteristic that an individual possesses intuitively. Thus, there is a very good chance that it can be a good measure to employ in order to recognize that particular typist. Similar to the previous feature, negUU is calculated as:

$$\text{NegUU} = \frac{\text{Number of negative UUs}}{\text{Total number of keypairs}} \tag{4.3}$$

4.2.2 Editing Features

The second category of features does not give any attention to the time a user spends typing, rather it considers the way a user goes about the process of typing. Characteristics such as how frequently a user commits typing errors and how he or she edits text are studied here.

The error rate is the first feature in this category and it captures the percentage of times a user performs a typing error and corrects it [80]. This is simply calculated by dividing the number of times that a user commits an error, i.e. presses the backspace button, by the total number of characters typed, as follows:

$$\text{Error rate} = \frac{\text{Number of errors}}{\text{Total number of characters}} \tag{4.4}$$

The next five features are closely related as they are all associated with the way a user incorporates capital letters and special characters in typing. Including a capital letter is done either by using the CapsLock key on the keyboard or by using a shift key together with the letter intended to be capitalized. It has been noted that if a user normally uses the CapsLock key, then he or she will hardly ever use the shift key for capitalizing letters, and vice versa. Therefore, using these two attributes simultaneously might be a good clue to understand the user's editing habits.

The first measure is CapsLock key usage which calculates the percentage of the CapsLock keys being used to produce capital letters in a given typing task [49]. This is simply computed using the following equation:

$$\text{CapsLock usage} = \frac{\text{Number of CapsLocks}}{\text{Total number of keys}} \tag{4.5}$$

The shift key usage is a bit more complicated than it might appear to be as there are two different aspects where users differ when it comes to shift key usage. The first shift key usage attribute is the right/left shift key choice. Some users use strictly the right shift or strictly the

left shift whilst others alternate between the two [45]. The second attribute is the order in which the shift/letter keys are released. The shift key is always pressed before the letter key if the user is intending to produce a capital version of that letter. However, there are two orders that users go about when releasing those keys. They either release the letter key before releasing the shift key or they release the letter key after releasing the shift key [80]. This behaviour proved to be quite consistent throughout the different typing tasks for most users.

Based on the previous observations, four different features that combine the two aspects of shift key usage were suggested. The percentage of each of the following was utilized: for the right shift key: right shift released after letter (RSA), right shift released before letter (RSB); and for the left shift key: left shift released after letter (LSA), left shift released before letter (LSB). They are calculated using Equation 4.6.

$$S = \frac{\text{Number of x}}{\text{Total number of shifts}} \tag{4.6}$$

Where: x= right shifts released after letter, in case S=RSA;
x= right shifts released before letter, in case S= RSB;
x = left shifts released after letter, in case S= LSA;
x = left shifts released before letter, in case S= LSB.

Table 4.1 gives an overview of all the nine typing features used in the non-conventional features study.

Table 4.1: Overview of the non-conventional typing features.

Category	Features
Semi-Timing Features	WPM
	NegUD
	NegUU
Editing Features	Error Rate
	CapsLock Usage
	RSA
	RSB
	LSA
	LSB

4.3 Decision Trees (DTs)

There are several types of "trees" that can be found in the literature. Of these, two types are perhaps the most significant. The first type is a classification tree, also referred to as a decision tree by default. The other commonly used basic decision tree is the regression tree. Classification trees, as the name implies, are used to divide the dataset into classes belonging to the response variable [235]. The response variable usually has two classes: Yes or No (1 or 0). The standard CART procedure [236] is used for these binary splits. However, if the response variable has more than 2 categories, then the algorithm C4.5 is used [236]. Thus, classification trees are used when the response or target variable is categorical in nature [237].

Regression trees, on the other hand, are used when the response variable is numeric or continuous. Predicting the price of a consumer good based on several input factors is an example of this [237]. Thus, regression trees are applicable for prediction type of problems as opposed to classification problems [235]. This section of the thesis is only concerned with the first type which is simply referred to as decision trees (DTs).

A decision tree is a classifier expressed as a recursive partition of the instance space [2]. The decision tree consists of nodes that form a rooted tree. This means that it is a directed tree with a “root” node that has no incoming edges. All other nodes have exactly one incoming edge. A node with outgoing edges is called an internal or test node. All other nodes are called leaves, and are also known as terminal or decision nodes. In a decision tree, each internal node splits the instance space into two or more sub-spaces according to a certain discrete function of the input attributes values. Commonly, each test considers a single attribute, such that the instance space is partitioned according to that attribute’s value. In the case of numeric attributes, the condition refers to a range of numbers [237].

Each leaf is assigned to one class representing the most appropriate target value. Instead, the leaf may hold a probability vector indicating the probability of the target attribute having a certain value. Instances are classified by navigating them from the root of the tree down to a leaf, according to the outcome of the tests along the path [237]. Figure 4.3 describes a decision tree and the layout of its nodes.

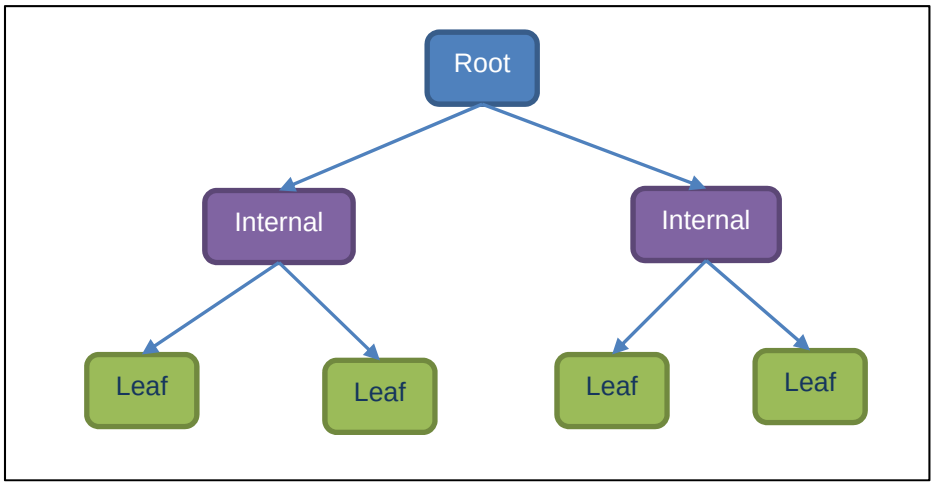

Figure 4.3: Decision tree layout.

In decision trees, each node is labelled with the attribute it tests, and its branches are labelled with its corresponding values. Each path from the root of a decision tree to one of its leaves can be transformed into a rule simply by joining the tests along the path and taking the leaf's class prediction as the class value [238].

Decision trees can be geometrically interpreted as a collection of hyperplanes which are orthogonal to one of the axes, in case of numeric attributes [239]. Consequently, decision-makers prefer less complex decision trees as they are more comprehensible. The tree complexity, which has a crucial effect on its accuracy, is explicitly controlled by the stopping criteria used and the pruning method employed [239]. Tree complexity is measured by one of the following metrics: the total number of nodes, tree depth and number of attributes used [239].

Thus, decision trees are algorithms that build a decision tree from a given dataset for the purpose of finding the optimal decision tree which can be achieved by minimizing the generalization error. Nevertheless, other goals can be also defined, such as minimizing the number of nodes or the average depth [238].

The growing phase of the tree continues until a stopping criterion is activated. Examples of common stopping rules are: all instances in the training set belong to a single target value, the maximum tree depth has been reached and the best splitting criteria is not greater than a certain threshold [238].

The various heuristic techniques for constructing decision trees can be divided into two categories: the bottom-up approach and the top-down approach [239]. In a bottom-up approach, a binary tree is constructed using the training set. A distance measure, such as Mahalanobis-distance, is used to perform pair-wise distances between prior defined classes and in each step the two classes with the smaller distance are merged to form a new group. The process is repeated until one class is left with one group at the root [239].

In the top-down approach, sets of classes are successively decomposed into smaller subsets of classes. There are various top-down decision trees inducers such as: ID3, C4.5 and CART. Some involve two conceptual stages: growing and pruning while other inducers perform only the growing phase [238].

In tree building, the top-down decision tree model (i.e. the most common) is built by recursively dividing the training data set based on a locally optimal criterion until all or most of the records belonging to each of the partitions carry the same class label. Tree pruning is used to enhance the generalization of a decision tree by pruning the leaves and branches responsible for classification of single or very few data vectors. Both phases will be briefly described here.

First, building a decision tree depends on choosing which attribute to test at each node in the tree. A measure called information gain is used to decide which attribute to test at each node. Information gain is itself calculated using a measure called entropy.

Entropy is an impurity measure, thus it is a measure of the homogeneity of the set of examples [238]. It is computed as:

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_- \quad (4.7)$$

Where C is a given binary categorisation and S is a set of examples for which the proportion of examples categorised as positive by C is p_+ and the proportion of examples categorised as negative by C is p_-.

The entropy is equal to 0 if the outcome is "certain" or completely homogeneous while the entropy is maximum, i.e. equal to 1, if there is no knowledge of the system or any outcome is equally possible. The entropy function for binary classification is shown in Figure 4.4 as the proportion of positive examples p_+ varies between 0 and 1.

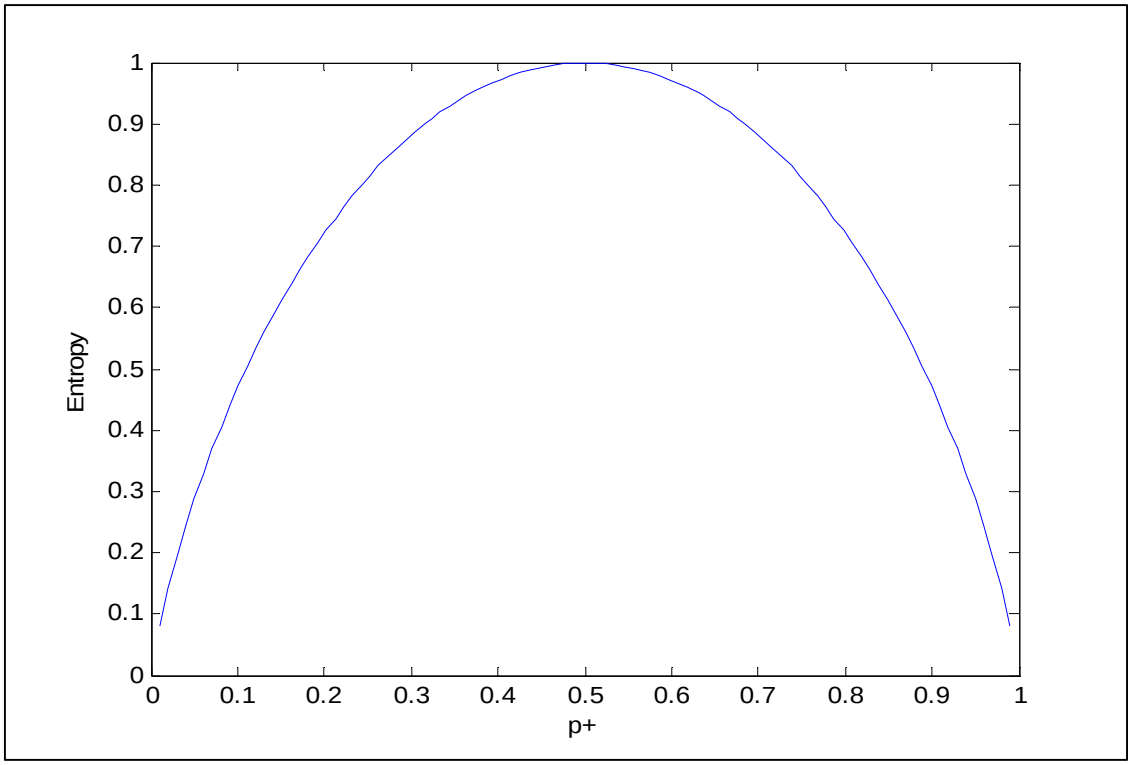

Figure 4.4: Entropy function for binary classification.

The information gain of an attribute can, on the other hand, be defined as the expected reduction in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding an attribute that returns the highest information gain i.e., the most homogeneous branches [238]. It is calculated as:

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \quad (4.8)$$

The above measure calculates a numerical value for a given attribute, A, with respect to a set of examples, S. Values of attribute A range over a set of possibilities which is called Values(A). Moreover, for a particular value from that set, v, there is S_v for the set of examples which have value v for attribute A.

Applying a tight stopping criterion has a tendency to create small and under-fitted decision trees, while utilizing a loose stopping criterion leans towards generating large decision trees that are over-fitted to the training set. Pruning methods, however, were created for solving this problem by applying loose stopping and allowing the decision tree to over-fit the training set and then the over-fitted tree is reduced into a smaller tree by eliminating sub-branches that are not contributing to the generalization accuracy [240].

Pruning is described as "trading accuracy for simplicity" as stated in [241]. It is used to produce a sufficiently accurate compact tree, where the initial decision tree is perceived as a completely accurate one. Therefore, the accuracy of a pruned decision tree shows how close it is to the initial tree.

There are various techniques for pruning decision trees. Cost-complexity pruning is one of them and it goes through two stages [240]. In the first phase, a series of trees T_0, T_1, . . . , T_k are created using the training data where T_0 is the original tree before pruning and T_k is the root tree. In the second phase, one of the trees created in the first stage is chosen as the pruned tree, based on its generalization error estimation. The tree T_{i+1} is acquired by substituting one or more of the sub-trees in the predecessor tree T_i with suitable leaves. The sub-trees that are pruned are those that obtain the lowest increase in apparent error rate per pruned leaf.

Another pruning technique is the reduced error pruning which works by traversing over the internal nodes from the bottom to the top and checking that the tree's accuracy is not affected negatively if each internal node is swapped with the most frequent class. If the accuracy is not compromised, then the node is pruned. This continues until any further pruning would cause the tree's accuracy to deteriorate [242].

These methods, among others, produce different performance levels. For example, cost-complexity pruning and reduced error pruning both tend to cause over-pruning, i.e. create smaller but less accurate decision trees. Other methods, such as: error-based pruning, pessimistic error pruning and minimum error pruning (which haven't been described here) all bias toward under-pruning [238].

4.4 Experimental Results and Discussion

This section presents the experiment results and discussion, in which the data collection, data space and the experimental results are indicated. A discussion about the experiment results and some comparisons with previous studies is performed in this section as well.

4.4.1 Data Collection

The data used in this experiment was collected from the same twenty-five participants involved in the extended key-pairing experiment discussed in Section 3.3.5.1. Moreover, the

data was extracted at the same time that the data for the extended key-pairing algorithm was retrieved, i.e. whilst the participants were typing the same eight tasks, as described in Section 3.3.5.1.

In addition to extracting the time of each key being clicked and released (to be used for the key-pairing method experiment), a number of additional typing attributes were gathered in each task. These additional attributes included: number of words per minute, number of errors, number of Caps-Lock, number of negative DU, number of negative DU, number of right and left shifts released before and after letters. An example of the data extracted from one typing task for this experiment is shown in Figure 4.5.

Words per Minute: 43.528
Total number of errors: 19, Percentage: 1.78236%
Number of Caps Lock used: 0, Percentage: 0.0%
Total number of minus DU: 394, Percentage: 36.9953%
Total number of minus UU: 8, Percentage: 0.751174%
Total number of times shift was released BEFORE letter (right shift): 0, Percentage: 0.0%
Total number of times shift was released AFTER letter (right shift): 0, Percentage: 0.0%
Total number of times shift was released BEFORE letter (left shift): 7, Percentage: 0.636%
Total number of times shift was released AFTER letter (left shift): 4, Percentage: 0.363%

Figure 4.5: Example of the non-conventional data collected from a typing task.

4.4.2 Data Space

A feature vector containing the nine features used in this study was created and stored in the database as the user's profile. This process was carried-out by considering each one of the eight typing tasks as a single typing sample, the features from which were extracted separately. Therefore, eight samples per subject were included in the analysis phase for classifier training and testing.

The following equations show a feature vector (V) that describes a user's single typing sample:

$$V = \{WPM, NegUD, NegUU, Error\ Rate, CapsLock\ Usage, RSA, RSB, LSA, LSB\} \quad (4.9)$$

Where:

WPM: is the number of words per minute, NegUD: is the percentage of negative DUs, NegUU: is the percentage of negative UUs, RSA: is the percentage of right shifts released after letter, RSB: is the percentage of right shifts released before letter, LSA: is the percentage of left shifts released after letter, LSB: is the percentage of left shifts released before letter.

Moreover, there was no need for discarding outliers as the non-conventional features did not rely on a time factor that might add noise in the form of too large or too small time lags. In addition, no scaling was needed as the non-conventional features are all quantities that represent percentages which vary between 0 and 1.

4.4.3 Experiment and Results

Cross-validation was used for classification with the consideration of the limited sample size in this study [243]. Cross-validation is a statistical sampling technique that aims to ensure that every example from the original dataset has the same chance of appearing in the training and testing set [244]. The leave-one-out cross-validation protocol, which is a special case of the well-known n-fold cross-validation, was followed [243].

N-fold cross-validation divides the data up into n chunks and trains n times, treating a different chunk as the test sample each time; such that for each of n experiments, it uses n-1 folds for training and the remaining one for testing. Leave-one-out cross-validation is exactly the same except that all chunks contain only a single sample [245].

In this experiment, eight samples were used to perform eight cross-validation experiments. Seven of the samples were treated as the training sample set and the remaining sample was regarded as the testing sample. In each experiment, a different sample was selected to act as the test data.

Decision trees were chosen as a classifier in this part of the research as they are strictly nonparametric and do not require assumptions regarding the distributions of the input data [246]. Moreover, it is considered a fast and scalable classification technique [247]. Furthermore, decision trees handle nonlinear relations between features and classes [235] which applies to the keystroke data.

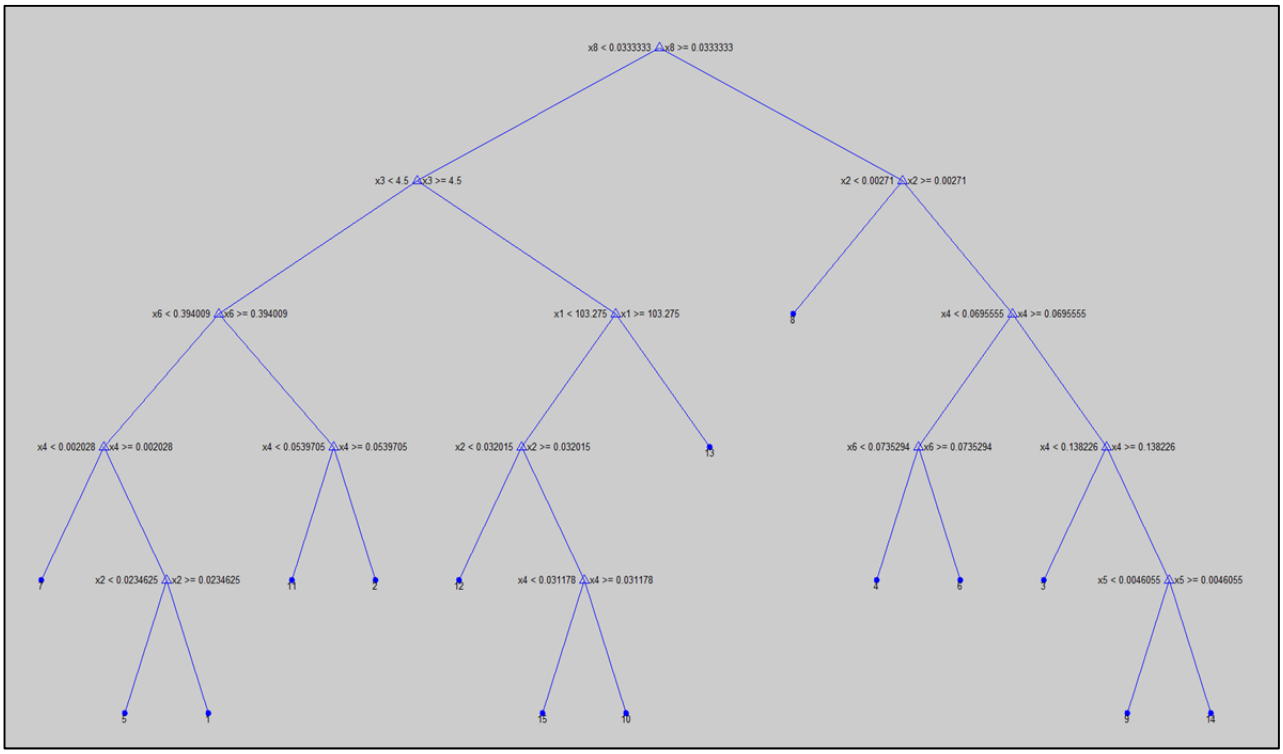

Figure 4.6: Example of a decision tree built for classification.

The Statistics toolbox [248] in MATLAB was used to fit the tree and predict the class of each of the test samples; the code used is provided in Appendix B.5. MATLAB's Statistics toolbox uses the CART algorithm to build classification trees [249]. Moreover, the tree structure, i.e. the order in which attributes were chosen to be tested at each node, varies each time when a different training set was selected. An example of a tree built by MATLAB for classifying the data in one of the cross-validation experiments is illustrated in Figure 4.6. The variables x_1 … x_9 denote to the features used to build the tree (refer to Section 4.2 for the full list of features).

Furthermore, two error rates were used to infer the performance, namely: FAR and FRR. They are both defined in Section 2.7 and they were chosen to be used here for reasons discussed in Section 3.2.2.4. Low error rates were produced by this study. The FAR and FRR derived by the decision tree classification process are listed in Table 4.2. Furthermore, the detailed results produced by each individual are provided in Appendix C.3.

Table 4.2: System performance of the non-conventional features.

	FAR	***FRR***
SVMs/ACO	0.0181	0.435
DTs	0.0104	0.25

The trade-off between FRR and FAR values [250] has also been noticed in this study. Nevertheless, the FAR being the lower of the two coincides with the aim of this research to protect the user against millions of potential impostors in a highly secured system.

Using these nine features simultaneously had a positive impact on the overall classification performance. This is due to the fact that DTs performs a form of feature selection in which only features that contribute to the overall-system decision are involved after building the tree and pruning it [251]. This is not the case when using only one or two features separately. This is due to the individual characteristics that each feature holds and that contribute collectively to the system performance [252].

For comparison purposes, Support Vector Machines (SVMs) were also used in this experiment as it is one of the most successful classification techniques [209]. SVMs were chosen as a rival classifier because it follows a completely different mechanism to that of decision trees [253]. In SVMs, the classification is performed by finding the optimum separating hyperplane [210], whereas classification in decision trees is purely based on decision rules [239]. Moreover, the one-vs-one method for multi-class classification was followed as it is faster than one-vs-rest in training. Although one-vs-one builds more classifiers than one-vs-rest, each classifier is built faster since one-vs-one includes only a part of the dataset [212]. Additionally, one-vs-one proved to outperform one-vs-rest in the key-pairing method experiment, carried-out in Section 3.3.

When using SVMs in classification, a feature subset selection is in place. This is because a number of the non-conventional features are correlated. Therefore, it is necessary to incorporate a feature subset selection mechanism when utilizing these features in order to reduce the dependencies between features [182]. Feature subset selection is also included in the building and pruning process of the decision tree where all redundant features are removed [251].

Feature subset selection is considered as an optimization problem, in which the space of all possible features is scrutinized to recognize the feature or set of features that produce optimal or near-optimal performance, i.e. those that minimize classification error [183]. Ant Colony Optimization (ACO) proved to be a good candidate for achieving that goal as proven in [182, 5].

The features selected by ACO were passed to the SVMs machine learning mechanism in order to be used as the basic data for differentiating between classes. Leave-one-out cross-validation was also used to treat seven of the samples as the training sample set and the remaining one as the testing sample, in each cross-validation experiment. The classification process was implemented on MATLAB with the aid of the LIBSVM library [227].

Feeding the classifier with features was done gradually by selecting one feature, using ACO, and then increasing the size of the features subset until all the features were included in the subset. Using only one or a small number of features produced substandard error rates. Similarly, using all or most of the nine features caused minor performance deterioration. The ideal size of the features subset was five features which produced a 0.0181 FAR and a 0.435 FRR. Table 4.3 illustrates the influence of increasing the feature set size on the overall system performance.

Table 4.3: Error rates using different feature subset sizes.

Number of Features	1	2	3	4	5	6	7	8	9
FAR	0.0373	0.0265	0.0229	0.0202	0.0181	0.0185	0.0188	0.0192	0.0221
FRR	0.895	0.635	0.55	0.485	0.435	0.445	0.45	0.46	0.53

Having the best features subset size to be only five features refers directly to the Curse of Dimensionality [222] (described in Section 3.3.5.3). Since there were limited number of samples per person in this experiment, there has to be a reduction in the number of features used for classification to the least amount possible. This guarantees that the large number of features is not affecting the classification process negatively.

Using ACO, the features that contribute the most to the system performance in the experimentation were: NegUD, Error Rate, RSB, LSA and LSB. Only using these features in the classification process eliminated the redundancy caused by using all nine features [254]. That clearly contributes to improving the overall system performance. Furthermore, as mentioned earlier, using only one or two of these features is not enough to find the fine differences between the typing behaviour of individuals in free-text keystroke dynamics [252].

As Figure 4.7 illustrates, FRR decreases dramatically until it reaches five features. After that, it increases but not as rapidly. Moreover, FAR decreases only slightly until it reaches five

features and it increases very slightly after that. Furthermore, FRR shows more variation of the two error rates when using different feature subset sizes.

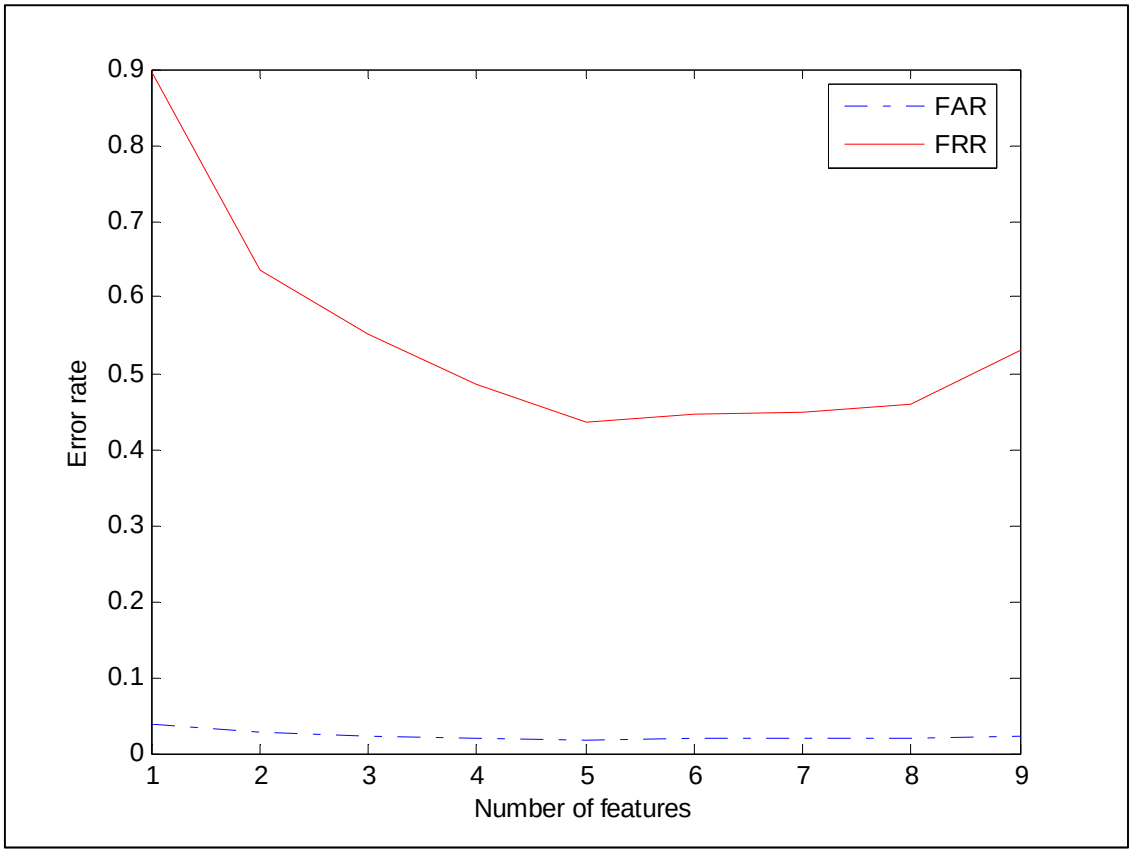

Figure 4.7: Error rates using different feature subset sizes.

Moreover, decision trees operate by automatically performing feature subset selection in which the non-important or redundant features are not involved after the tree building and pruning process [251]. Features: WPM, NegUD, Error rate, LSA and RSB contributed the most in building the decision tree as they formed the first levels of the tree structure. Thus, they have a high capability to split the targets [238], which allows for better differentiation between individuals. Therefore, these features correspond to the features with higher impact on the performance of the authentication system.

This partly matches the features extracted using ACO; as four of the five features extracted by ACO are common with the ones in the subset selected by the DTs (i.e. NegUD, Error rate, LSA and RSB). Thus, these features were found to have a considerable effect on system performance in both DTs and SVMs/ACO classification cases. Moreover, based on the features subsets selected by both ACO and DTs, the editing features appear to have a slightly higher influence on the system performance compared with semi-timing features. This is due

to the fact that the majority of the features extracted by ACO and DTs are from the editing features sub category.

Conclusively, DTs have a slight performance advantage over SVMs (as shown in Figure 4.8). DTs produced a higher accuracy system as the ROC is plotted closer to the upper left corner of the diagram in Figure 4.8.

Even though both DTs and SVMs are capable of handling non- linearly separable data [247], DTs performed slightly better in this experiment. This could be due to the nature of the data and the fact that it can be easily converted into rules that decision trees can apply to separate the data. As revealed in [247], the best classifier for a particular task is considered task-dependent.

Similar to the error rates produced by the key-pairing method, biased results are noticeable as there is a large difference between the FAR and the FRR values in the results produced by both SVMs/ACO and DT. This is because there were more negative samples used in training than positive samples [232].

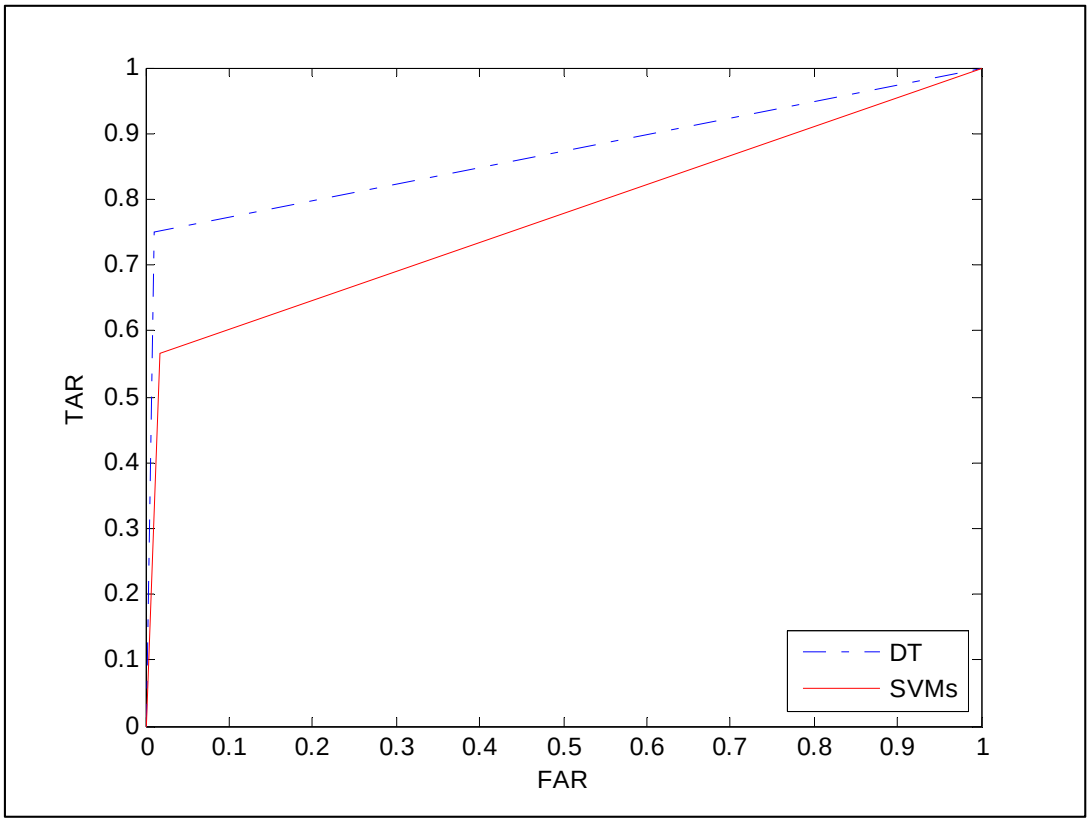

Figure 4.8: DTs and SVMs ROC curves.

4.4.4 Discussion

This study was performed using the data collected in the extended key-pairing approach experiment (described in Section 3.3) in which user classification based only on timing features was considered. These features included the hold time, up-up, down-down and up-down of specific key-pairs. Although the performance of that system was acceptable (FAR: 0.013, FRR: 0.384), there was a larger than desired FRR.

By using non-conventional features, the FRR has been dramatically improved with a value of 0.25 in this study. While this figure is still not ultimate, it is quite good when considering the small amount of text used to authenticate individuals. Moreover, a lower FAR was also produced by non-conventional features. The FAR, being as small as 0.0104, leads to high expectations for further research in this area.

Comparing the two error rates produced by the timing features and the non-conventional features demonstrates that the FRR is considerably lower in non-conventional features. However, the FAR produced by the non-conventional features is very slightly lower. It also has to be noted that the number of training samples used in the timing features experiments was significantly higher. There were twenty samples per person in the timing features experiment, fifteen of which were used for training. Meanwhile, the non-conventional features experiment made use of only eight samples per person, seven of which were used for training. The fact that the error rates produced by non-conventional features are lower than those produced by timing features, as shown in Figure 4.9, proves the superiority of non-conventional typing features over conventional timing ones, in user authentication.

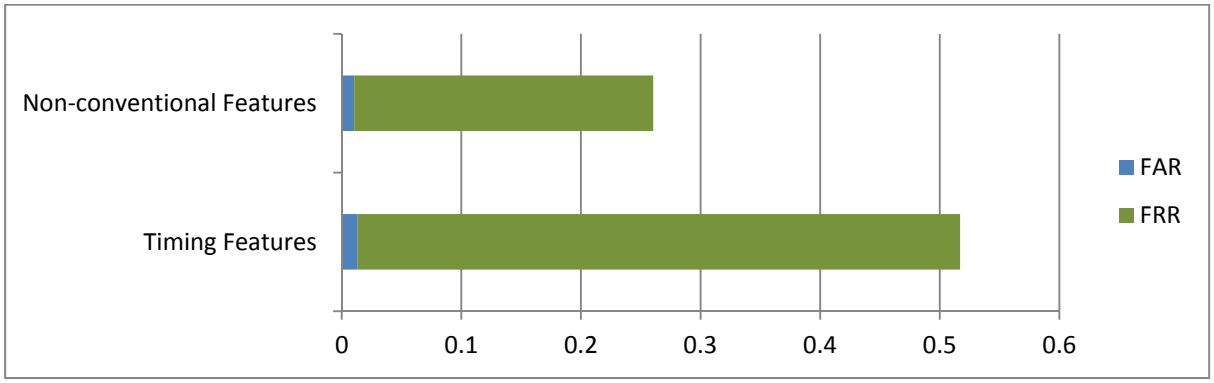

Figure 4.9: Comparison between the error rates produced by the timing features and the non-conventional features.

The use of non-conventional features proposed in this chapter has succeeded in providing a reliable medium for user authentication because employing these features broadens the amount of information that can be extracted from a user's input. This is because non-conventional typing features are extracted collectively during the whole time a text is being typed by the user. In this input more information is available, such as: words-per-minute, error rate, percentage of negative UDs … etc. Therefore, using this wide range of information that is available about the user's typing stream, the system is able to assemble better indication about the user's typing behaviour, thus effectively distinguish between individuals.

Similar research was conducted by Curtin et al. [79] in which 58 features were extracted. The features varied between conventional timing ones and non-conventional ones such as total time to enter the text, total number of key presses for Space, Backspace, Delete, Insert, Home, End, Enter, Ctrl, all four arrow keys, left and right shift keys and the number of left, right and double mouse clicks. Recognition accuracy of 98.5% resulted from data collected from eight subjects typing ten 600-characters long training samples and ten 300-characters long testing samples. This would have been a very encouraging result if the number of subjects was larger and/or the length of text was shorter.

Moreover, a dataset of 15 emails for each of 10 participants was created in the study conducted by Hempstalk et al. [80]. In their experiment, eight features were extracted, some of which were based around the typist's speed: (average words-per-minute (WPM) rate, peak WPM, trough WPM), error rate: (backspaces, paired backspaces, average backspace block length) or slurring of key press and release events: (press/release ordering, press/release rate). Using one-class SVMs an FAR of 0.331and an FRR of 0.113 were achieved. This shows a similar FRR to the one produced in this study yet this research proved to realize a better FAR.

Using a larger dataset and incorporating data from a greater number of participants will likely produce more reliable results. Therefore, to investigate the scalability of the non-conventional features method, an experiment was conducted to compare the error rates produced by different number of participants. To understand how increasing the sample size will affect the system performance, both DTs and SVMs/ACO experiments were performed on data from fifteen and thirty users in addition to the original twenty-five users experiment.

Using datasets of participant numbers varying between 15, 25 and 30 users delivered a noticeable reduction in the system performance when increasing the number of participants

from 15 to 25 (as shown in Table 4.4). Nonetheless, the increase from 25 users to 30 have produced very similar FAR and FRR. This demonstrates that the system can reach a stable level when enlarging the number of participants which proves the scalability of this method and its ability to cope with larger sample size [255].

Table 4.4: System performance using different number of participants.

	FAR			***FRR***		
Participants no.	**15**	**25**	**30**	**15**	**25**	**30**
DTs	0.007	0.0104	0.0109	0.1	0.25	0.28
SVMs	0.0125	0.0181	0.0183	0.175	0.435	0.444

4.5 Summary

In this part of the thesis, the usefulness of incorporating non-conventional keystroke features in the user authentication process was examined. Unlike conventional timing features, non-conventional features benefit from the extra information that can be extracted from the typing of a piece of free-text input. Features that have semi-timing properties such as words-per-minute, percentage of negative up-down time and percentage of negative up-up time were used. Moreover, features that explain the user's editing behaviour were also used. These included the error rate, percentage of CapsLock usage, and percentage of both right and left shift keys usage. DTs and SVMs were used to classify the typing samples collected from participants. In the case of SVMs, ACO was utilized to select features that contribute more to the system.

Non-conventional features such as those used in this study appear to be highly significant in keystroke dynamics applications such as user authentication. Moreover, decision tree classifiers demonstrated a high level of success in such cases. In addition, based on the features selected by ACO and DTs, the editing features seem to have a more positive effect on the system performance compared with semi-timing features.

The results obtained from this study are encouraging as low FAR and FRR were achieved in the experimentation phase; with the FAR being the slightly better of the two. This signifies that satisfactory overall system performance was achieved by using the non-conventional attributes proposed in this study. Thus, the use of non-conventional typing features improves the understanding of human typing behaviour and therefore, provides significant contribution to the authentication system when using little training data.

Non-conventional features succeeded in reducing the FRR produced by timing features. Yet the FAR produced by timing features was lower. This designates the opportunity of using both types of features in conjunction with each other (fused features) to produce even better system performance.

Lastly, due to their nature, non-conventional features have the quality of being language-independent, similar to the key-pairing method. This can be tested, as stated in the future work section, by applying these features to a language different than English.

Chapter 5

Performance Improvement by Fusion

5. 1 Introduction

This chapter is concerned with improving the methods introduced previously in order to produce better system performance. This is done by performing fusion of the key-pair timing features, referred to simply as "timing features", and the non-conventional features (introduced in Section 3.3 and Section 4 respectively) at different levels, i.e. feature-level and decision-level fusion. Moreover, the fusion is performed to overcome the limitations that each method suffers from when used separately. That will aid in reducing the error rates produced by the new and advanced system [256].

The feature-level fusion is done by joining the timing features and the non-conventional typing features and testing the system performance based on the concatenated data. Meanwhile, the decision-level fusion is performed by combining the outputs of two methods. They are: the method that involves classifying timing features using SVMs/ACO and the method utilizing non-conventional features in a DTs classification.

Moreover, a comparison is made between the two fusion schemes that were applied to free-text keystroke data, in this study. This is motivated by the demand of an improved authentication system which is able to provide users with the highest security using as little typing input as possible.

A number of research in the current literature has considered fusion in keystroke dynamics. The work in [72] proposed a decision-level fusion between two methods. The first being the Gaussian similarity score between a reference template and a test data template. The second being the Direction Similarity Measure (DSM) for comparing the typing patterns of a user. The two scores were fused by using a weighted sum rule. Fifty participants were requested to type-in their username, password and a special fixed phrase repeatedly ten times. The

performance achieved using only the Gaussian probability density function yielded an EER of 11.6897%, while the performance using only the DSM produced an EER of 19.74%. Combining the two methods delivered the best result of an EER of 6.36%.

Hocquet et al. [134] performed a study for authenticating users using a decision-level fusion of three methods. The first method used the mean and the variance of each latency time and compared it to a threshold. The second method used a measure of typing rhythm disorder where the time was classified into five different classes according to the speed. The difference between the numbers of the classes in the profile data and in the test data was calculated and then the sum of all these differences was compared to a threshold. The third and last method was based on the ranks of the latencies; this was performed by ordering the latencies based on their speed. The latency time of each observation was ordered from the slowest to the fastest for each user's profile. The Euclidean distance between the user's profile and the new data was then used to guess if the new observation belonged to that user. Even though these methods work well on their own, Hocquet et al. studied the possibility of combining all three with the help of a fusion rule after normalizing the scores from each method. For testing these methods, 15 users were asked to give a login password ten times, followed by 30-to-100 log-in attempts over a six months period. The results of using each of these techniques separately gave a best performance of 3.70% EER. The fusion of all three methods, on the other hand, significantly improved the performance to an EER of approximately 1.8%.

Work conducted in [257] employed feature-level fusion, in which the authors combined a pressure sequence and traditional keystroke dynamics in user authentication. In addition, three methods were combined for the authentication task using a decision-level fusion technique. They were: (1) global features of pressure sequences, (2) dynamic time warping, and (3) traditional keystroke dynamics. A total of 5000 password samples were obtained from 100 individuals for this study. All experiments utilized a dataset that combined both a pressure sequence and traditional keystroke dynamics. Utilizing all methods, an EER of 1.41% was achieved compared with a 2.32% EER for method (1), a 1.92% EER for method (2) and a 2.04% EER for method (3), on their own. Moreover, an EER of 2.04% was achieved using only traditional keystroke dynamics.

Based on the literature, not many studies have shown an interest in understanding the differences between using decision-level and feature-level fusion in the area of free-text

keystroke dynamics authentication. In this study, therefore, the two fusion techniques were explored with the aim of finding which of the two is more suitable for the task of uniquely recognizing the typist involved based on a reduced amount of free-text input.

5. 2 Fusion Overview

Decision support systems (DSS) are schemes to create a model that is able to produce correct decisions given a minimum amount of input data. DSS follows two different schemes [258]. The first of which suggests that the progress of DSS should be based on continuous improvement of existing methods and establishing new ones, and the second approach recommends combining existing well performing methods, anticipating that better results will be achieved as the limits of the existing individual method are reached and it is not possible to develop a better one. Such fusion of information seems to be worth applying in terms of reducing uncertainty [259]. As each of the individual methods produces some errors, different methods performed on different data may produce different errors. Assuming that all individual methods perform considerably well, the combination of such methods should reduce the overall classification error [258].

Fusion of data/information can be carried-out on three levels: data-level fusion, feature-level fusion, and decision-level fusion [260]. A brief description of each of these types is presented in this section.

5.2.1 Data-level Fusion

Data-level fusion combines multiple sensor data that measure correlated parameters [261]. Using multiple sensor systems has many advantages including: higher signal-to-noise ratio, increased robustness and reliability in the event of sensor failure, reduced uncertainty and increased confidence [261]. Thus, the integration of data from a multiple sensor system, i.e. data fusion, is important to improve decisions.

A number of data fusion frameworks have been developed, and these include, among others: Thomopoulos architecture [262], behavioural knowledge based data fusion [263] and the waterfall model [264].

One of the well-known frameworks for data-level fusion is the Thomopoulos [262], in which three-level architecture was proposed for data fusion. Each of which is integrating data at

different levels. These levels are: (1) signal level fusion: where data correlation takes place through learning due to the lack of a mathematical model describing the phenomenon being measured, (2) evidence level fusion: where data is combined at different levels of inference based on a statistical model and the assessment required by the user and (3) dynamics level fusion: where the fusion of data is done with the aid of an existing mathematical model. These levels of fusion are implemented sequentially or interchangeably based on the application in hand.

5.2.2 Feature-level Fusion

Feature sets obtained from several data sources can be fused to create a new feature set representing a single object. Compared with data-level fusion, feature-level fusion is more concerned with the meaningful features that were extracted from raw data. An example of feature-level fusion is illustrated in the following scenario (as described in [265]): the geometric features of the hand may be augmented with the eigen-coefficients of the face in order to build a new high-dimension feature vector. The concatenated feature set demonstrates better discrimination capability compared with the individual feature vectors obtained from hand geometry and face biometrics separately. Subsequently, a feature subset selection technique may be implemented to choose a minimal feature set from the high-dimensional feature vector [266].

The most common way to accomplish feature-level fusion is by a simple concatenation of the feature sets obtained from multiple information sources [265]. Let $X = \{x_1, x_2, ..., x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ denote feature vectors ($X \in R_m$ and $Y \in R_n$) representing the features extracted via two different sources. Combining these two feature sets in order to yield a new feature vector, Z, that would better represent the individual is the way followed in feature fusion. The vector Z is produced by first concatenating vectors X and Y, and then performing feature selection on the resulting features vector.

Another popular approach is image blending [267] which creates a single fused image stream by blending images from each of the used instruments. Object location and shape are then extracted by applying appropriate detection and segmentation algorithms to the fused image stream.

Union of features [268] is another popular method that postpones the fusion of information to a later stage in the pipeline. Such approaches normally involve complete image segmentation

routines in each image stream and then combining the segmentation results across the sensing domains.

5.2.3 Decision-level Fusion

It is also referred to as classifier-level fusion. A number of decision-level fusion methods have been introduced to find an alternative approach leading to potential improvement in the classification performance. There are two categories of decision-level fusion techniques. They are methods operating on classifiers and methods operating on outputs. Both categories are briefly described here.

5.2.3.1 Methods Operating on Classifiers

The methods in this category generally operate on classifiers and emphasise the development of the classifier structure. They do not consider classifier outputs until the combination process finds a single best classifier or a selected group of classifiers and only then are their outputs taken as a final decision or for further processing. There are three subcategories included here: Dynamic Classifier Selection (DCS), Classifier Structuring and Grouping and Hierarchical mixture of experts (HME).

Dynamic Classifier Selection (DCS) attempts to determine a single classifier, as opposed to a group of classifiers, which is the most likely to produce the correct classification label for an input sample [269]. As result, only the output of the selected classifier is taken as a final decision. Moreover, it includes partitioning of the input samples. All DCS methods heavily depend on training data and by only choosing the locally best classifier, they seem to lose some useful information available from other well performing local classifiers [269].

Classifier Structuring and Grouping, on the other hand, works by organising classifier combination. Classifiers and their combination functions may be organized in many different ways [269]. Usually it is done by organising them in parallel, and then use their outputs simultaneously and separately as an input for a combination function. Alternatively, apply several combination functions sequentially. A less frequently used strategy is to organise all classifiers into groups and to apply different fusion methods for each group. In general, classifiers may be arranged in a multistage structure. At each stage, different fusion methods are applied for different groups of classifiers.

A Hierarchical mixture of experts (HME) represents a supervised learning technique based on the divide-and-conquer principle [258]. The HME is organised in a tree-like structure of leaves. Each leaf represents an individual expert network, which given the input, vector x, tries to solve local supervised learning problem [258].

5.2.3.2 Methods Operating on Outputs

This category of methods operates mainly on the classifiers' outputs, and effectively the combination of classifiers' outputs is calculated [270]. This category is further divided according to the type of the output produced by individual classifiers. This category can be divided into three subcategories, namely: crisp labels, class rankings and soft/fuzzy outputs. A brief description of each type is provided here.

5.2.3.2.1 Crisp Labels

Classifiers producing crisp, single class labels (SCL) provide the least amount of useful information for the combination process [258]. However, they are still applied to a variety of real-life problems with well-produced performance. There are a number of methods for crisp labels. Two of the most representative methods for fusing SCL classifiers are generalised voting method and the Knowledge-Behaviour Space (BKS) method.

First, voting strategies can be applied to a multiple-classifier system assuming that each classifier gives a single class label as an output. Let the output of the classifiers form the decision vector d, defined as:

$\mathrm{d} = [\mathrm{d}_1, \mathrm{d}_2, \dots, \mathrm{d}_n]^{\mathrm{T}}$ where $\mathrm{d}_i \in [\mathrm{c}_1, \mathrm{c}_2, \dots, \mathrm{c}_m, \mathrm{r}]^{\mathrm{T}}$, c_i denotes the label of the i-th class and r is the rejection of assigning the input sample to any class. Let the binary characteristic function be defined as follows:

$$B_j(c_i) = \begin{cases} 1 \; if \; d_j = c_i \\ 0 \; if \; d_j \neq c_i \end{cases} \tag{5.1}$$

Then the general voting routine can be defined as:

$$E(d) = \begin{cases} c_i \;\; if \; \forall_{t \in \{1,\dots,m\}} \sum_{j=1}^{n} B_j(c_t) \leq \sum_{j=1}^{n} B_j(c_i) \geq \alpha.m + k(d) \\ r \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad otherwise \end{cases} \tag{5.2}$$

Where α is a parameter and k(d) is a function that provides additional voting constraints. The case where α= 0.5 is commonly known as the majority vote [258].

The Behaviour-Knowledge Space method (BKS) does not assume independence of the decisions made by individual classifiers, as opposite to other fusion methods [271]. It delivers a knowledge space by collecting the records of the decisions of all classifiers for each learned sample. If the decision fusion problem is defined as a mapping of K classifiers: $e_1, ..., e_K$ into M classes: $c_1, ..., c_M$, BKS operates on the K-dimensional space where each dimension corresponds to an individual classifier, which can produce M+1 crisp decisions, M class labels and one rejection decision. The final decision represents a balance between the current classifiers' decisions and the recorded behaviour information.

5.2.3.2.2 Class Rankings

Some additional useful information can be gained from classification methods generating outputs in a form of class rankings [258]. There are many fusion methods operating on class rankings, two of the most commonly used are listed here.

The first approach is the Class Set Reduction (CSR). Its objective is to reduce the set of considered classes to as small a number as possible while guaranteeing that the correct class is still represented in the reduced set. CSR is very appropriate for the early stage of combining multiple classifiers. The main goal of the class set reduction method is to find the trade-off between the minimising of the class set and the maximising of the probability of inclusion of the true class.

The second approach is the Class Set Reordering (CSRR). It strives to class set reordering in order to improve overall rank of the true class. The CSRR method is considered to be successful if it ranks the true class higher than any individual classifier.

5.2.3.2.3 Soft/Fuzzy Output

Fusion methods operating on classifiers with soft/fuzzy outputs are the largest group of decision-level fusion methods and can be expected to produce the greatest improvement in classification performance [272]. The outputs in this category refer to the real values in the range [0, 1]. These values are referred to as fuzzy measures, which involve all known measures of evidence: probability, possibility, necessity, belief and plausibility [272]. These

measures are all used to define information uncertainty in different dimensions. Therefore, the main aim of fusion methods in this category is to reduce the level of uncertainty by maximising suitable measures of evidence.

A large number of methods are applied on soft output classifiers. An example of such methods is the Bayesian methods which can be applied to the decision-level fusion subject to the condition that the outputs of the classifier are expressed in posterior probabilities. It works by assigning the decision to the class with the highest combined belief measure [258].

5. 3 Experimental Results and Discussion

This section presents the experiment results and discussion, in which the data collection, data space and the experimental results are indicated. A discussion about the experiment results and some comparisons with previous studies is performed in this section as well.

5.3.1 Data Collection

The features used in this part of the study are: the timing feature described in Section 3.3.2 and the non-conventional features discussed in Section 4.2. The data used in this experiment was collected from the same twenty-five participants involved in the extended key-pairing experiment, the details of which are discussed in Section 3.3.5.1. Moreover, the timing data and the non-conventional data were both extracted at the same time, i.e. whilst the participants were typing the same eight tasks, as described in Section 4.4.1.

5.3.2 Data Space

The data space for the timing features used in this section is similar to that discussed in the extended key-pairing experiment (Section 3.3.5.2). The only difference is that the data was not combined and partitioned into twenty chunks as done in the key-pairing experiment. Instead, each one of the eight typing tasks was considered as a sample. This was done because in non-conventional features, it is not the case to combine the samples and partition them as each of the features represent the whole piece of text included in a sample (as discussed in Section 4.2). For example, the word-per-minute feature was calculated for the whole sample, each sample separately. Thus, each user had eight timing features samples in their profile. Similarly, eight non-conventional features samples were stored for every user as described in Section 4.4.2.

5.3.3 Experiment and Results

As mentioned earlier, two types of fusion techniques are applied in this study. They are: feature-level fusion and decision-level fusion (as illustrated in Figure 5.1). A description of how each type of fusion was applied is presented in this section.

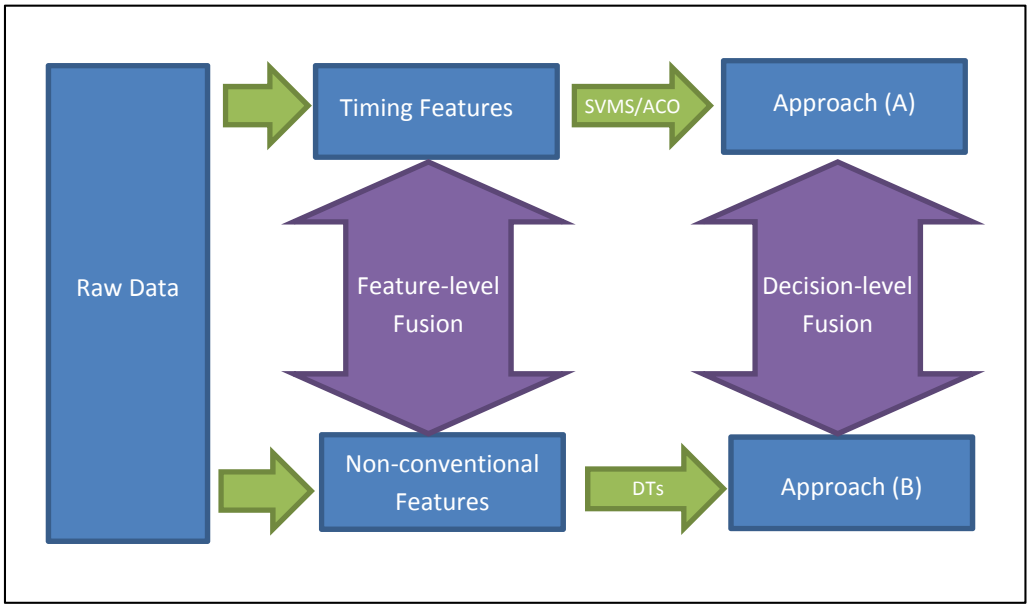

Figure 5.1: Feature-level fusion and decision-level fusion.

5.3.3.1 Feature-level Fusion

Timing features and non-conventional features were combined to produce a larger set of features. This allowed for the 55 timing features and the nine non-conventional typing features to be joined in order to create a 64 mixed features dataset. This set was exploited in two classification processes. One of which was done using the SVMs/ACO technique and the other was done using DTs.

The details of how SVMs/ACO and DTs were applied is similar to that explained in Section 4.4. Both classification methods were carried-out through cross-validation [243]. This involved eight samples being utilized in performing eight cross-validation experiments. Seven of the samples were treated as the training sample set and the remaining sample was regarded as the testing sample. In each experiment, a different sample was selected to act as the testing data.

Similarly, two error rates were used to infer the performance, FAR and FRR; both discussed in Section 2.7. These error rates were chosen for reasons discussed in Section 3.2.2.4. The final error rates resulting from the feature-level fusion are shown in Table 5.1. The details of each individual's results are shown in Appendix C.4.1.

Table 5.1: Feature-level fusion of timing features and non-conventional features

	FAR	***FRR***
SVMs/ACO	0.0156	0.375
DTs	0.00896	0.215

5.3.3.2 Decision-level Fusion

The decisions from two different approaches were fused to achieve decision level fusion in this section. Approach (A) utilized SVMs together with ACO to classify the timing features. The complete description of this method is found in Section 3.3. In Approach (B), the non-conventional features were analysed using a DTs classifier. Similarly, the complete explanation of this method is indicated in Section 4.

The decision to use the pair of the SVMs and DTs classifiers was based on the results produced in Chapter 3 and Chapter 4. It has been shown that good performance was produced when using SVMs with timing features and DTs with non-conventional features. Nonetheless, other combinations of classifiers exist, i.e. SVMs & SVMs, DTs & SVMs and DTs & DTs. However, these combinations were not considered in this study. The benefit of using these other combinations can be investigated in future work.

Both approaches were carried-out using cross-validation in the same manner described in the previous section. As both classifiers, from approaches (A) and (B), produced crisp, single class labels (SCL), the majority voting method was used to fuse the two classifiers. Besides, it was used because of its simplicity and effectiveness [273]. A brief description of the voting strategy is presented in Section 5.2.3.2.1.

Using cross validation, each of the eight typing tasks was considered as a single sample, all of which were used in the eight cross validation experiments. This was done for both approaches yielding in 16 different cross-validation experiments. Each experiment's result was considered as a vote, i.e. there were 16 results to be included in the overall voting for each class. Eight of these votes belong to approach (A) and the other eight belong to

approach (B). Using the majority voting scheme, the overall error rates resulting from this decision-level fusion yielded in a 0.00 FAR and a 0.00 FRR. The details of each individual's results are shown in Appendix C.4.2.

5.3.4 Discussion

This experiment is a continuation of the work previously conducted in Section 3.3 and Section 4. In Section 3.3, the timing features extracted from specific key-pairs were exploited and ACO was used to select a feature subset that was fed into a SVMs classifier. The resulting FAR was low while the resulting FRR was not satisfactory. Moreover, in the work carried-out in Section 4, the non-conventional typing features were utilized to distinguish between users using the DTs classifier. Slightly better FAR and FRR were produced. A fusion of these two methods was the next step in order to achieve better authentication performance [256]. Table 5.2 shows the error rates produced in the previous studies.

Table 5.2: Previous studies performance.

Study	*Approach*	*Features*	*Method*	*FAR*	*FRR*
Study1	Approach (A)	Timing features	SVMs/ACO	0.013	0.384
Study2	Approach (B)	Non-conv. features	DTs	0.0104	0.25

In feature-level fusion, SVMs/ACO produced average rates, yet it very slightly improved Approach (A)'s FRR. Meanwhile, DTs produced overall good error rates and it was able to slightly enhancing both error rates of Approach (B). Thus, the fusion of the two feature sets had a slight positive impact on the FRR value yet the FAR obtained from fused features was only improved in the DTs case. It has to be noted, however, that the number of samples in the study testing approach (A) was 20 samples per person, 15 of which were used for training, whilst only 8 samples per person were used in the study testing approach (B), 7 of which were used for training.

Moreover, in both cases, the trade-off between FAR and FRR [250] was in favour of the FAR as it produced lower error rates compared with FRR. This corresponds to the main goal of user authentication and the fact that it is more concerned with protecting users against the vast numbers of potential impostors in highly secured systems [274].

Furthermore, DTs proved to have better recognition outcome compared with SVMs/ACO as shown in Table 5.2 since it was able to produce lower error rates from the fused features.

This has a lot to do with the nature of the data in this case and its ability to construct rules that are used by the DTs to separate the data [247].

It is noteworthy that both methods include some form of feature subset selection. ACO is a feature subset selection technique applied to the fused features to select the most suitable features before being fed to SVMs [5]. DTs are also performing feature subset selection on the fused features when building and pruning the decision tree [275]. In addition, using all features with no feature selection did not produce good results as the level of noise was larger in such high features dimensionality [170]. The Curse of Dimensionality (described in Section 3.3.5.3) can be used to explain the performance deterioration when using a larger number of features [222]. As there are only eight samples per person, the number of features should be as small as possible to correctly represent the small number of samples.

When looking more closely at the features selected to be utilized in both methods, it is noted that non-conventional features were in the majority over timing ones. Four features out of five selected by ACO and five features out of the eight chosen by DTs were non-conventional features. This supports the belief that non-conventional features represent human typing patterns more precisely compared with timing features in free-text keystroke systems. The selected features for both methods are listed in Table 5.3.

Table 5.3: Selected features from the fused features set.

	Timing features	**Non-conventional features**
SVMs/ACO	SL-H2	Error rate | RSA | LSA | LSB
DTs	SL-H1 | SL-UD | FL-H1	WPM | NegUU | Error rate | RSA | LSA

Thus, non-conventional features appear to have a strong relationship between input values and target values, in this data set. A strong input-target relationship is formed when knowledge of the value of an input improves the ability to predict the value of the target which helps in understanding the characteristics of the target [276].

Moreover, in agreement with the outcomes of Section 4, the editing features appear to have more influence on the system performance in the feature-level fusion as four out of four and three out of five of the non-conventional features selected by ACO and DTs, respectively, were editing features.

In decision-level fusion, using majority voting has considerably improved the results of Approach (A) and Approach (B), individually. A perfect recognition rate was achieved using decision-level fusion. Even though not all votes were for the correct class, there was no common wrong class. The majority of votes were either for the correct class or for random classes which has no effect on the majority vote.

In addition, approach (B), i.e. DTs and non-conventional features, has contributed the most to this very good result. It produced more correct votes which lead to a correct overall decision. This supports the existing conclusion which states that non-conventional features represent the human typing behaviour better than timing ones.

A comparison between the error rates produced by feature-level fusion and decision-level fusion is illustrated in Figure 5.2. It demonstrates the low error rates produced by feature-level fusion in the case of the ACO/SVMs and DTs classifiers. It also shows a slight advantage of the DTs method over the ACO/SVMs. Moreover, the decision-level fusion produced zero FAR and FRR which proved that using majority voting decision-level fusion produces better system performance compared with the feature level-fusion and succeeds in producing a very good recognition system.

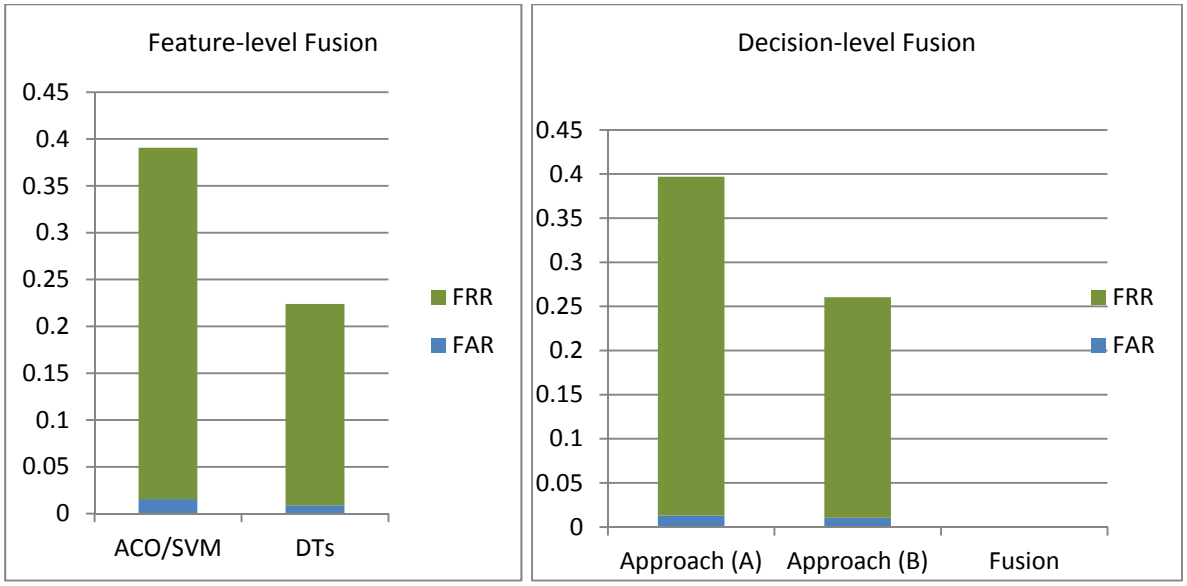

Figure 5.2: Comparison between the error rates produced by feature-level fusion and decision-level fusion.

Lastly, the bias between the FAR and the FRR values is still present in the feature-level fusion. It has occurred because of the uneven data set as the positive samples were much less than the negative ones [233]. Interestingly, however, the bias has disappeared in the decision-level fusion as the FAR and FRR were both found to be zeros.

5. 4 Summary

In this section of the thesis, an attempt to improve the methods introduced previously was carried-out using two types of fusion techniques. They are: decision-level fusion and feature-level fusion. Moreover, these fusion mechanisms have been applied to achieve performance improvement in free-text keystroke dynamics authentication with the requirement of reduced training data.

Feature-level fusion was performed to combine timing features and non-conventional features before providing the classifier with the fused set of features. Decision-level fusion, on the other hand, was carried-out to merge the outcomes of two classification approaches using the majority voting approach. In the first approach, SVMs were used to classify a subset of the timing features that was selected using ACO while the second approach used DTs to classify non-conventional features.

Feature-level fusion was able to reduce the error rates produced by the two feature sets separately, especially using the DTs classifier. Yet, decision-level fusion proved to be superior to feature-level fusion as it succeeded in producing zero FAR and FRR. This dramatic enhancement showcases the fact that using majority voting on the joint decision of both approaches provides a good indication of the users' typing behaviour. That will eventually provide a good assessment of the user's identity when applying a reduced training authentication scheme. It has also been noted that non-conventional features have the edge over the timing features as they provided a better recognition rate throughout the study.

Chapter 6

Exploration of Keystroke Dynamics in Arabic Language

6. 1 Introduction

This chapter is investigating the language-independency property of the key-pairing method, introduced in Section 3.3. The extended key-pairing method has been developed to accept input in any language. This is tested in this section by applying the key-pairing method to text in Arabic language.

The key-pairing approach involves the use of the keyboard's key-layout to group the text into key-pairs based on the two keys location on the keyboard. The method works after that by extracting a number of timing features from these specific key-pairs. Thus, the key-pairing method does not have to be used with a certain language as opposed to the standard keystroke dynamics authentication process, i.e. without the use of key-pairs. Standard keystroke methods such as the ones introduced in [8, 115] require the comparison between two samples to be done based on a certain combination of letters, which will be different for each language. However, using the same keyboard layout to collect the typing data will allow for the employment of the key-pairing method on any language.

Furthermore, most of the studies in keystroke dynamics involved only English input from the user. Other languages have not received the same attention that English has in the research literature to date. Whilst such experimentation is very important, there is clearly a lack of language variation used in such systems.

The work done by Gunetti et al. [88] is one of the few research studies applied to languages other than English in the area of keystroke dynamics authentication. In that study, a combination of the two measures developed by Gunetti and Picardi [8] are used to assess the

similarities between typing patterns. This was done using the duration time of di-graphs found in samples typed in both English and Italian. Italian was used since the two languages, i.e. English and Italian, have the same alphabet and thus share a considerable number of di-graphs (character-pairs). From experimenting with different combinations of training and testing samples, the best results were achieved when the user's profile contains samples in both languages yet the performance increases when the language of the testing samples is the dominant language in the samples forming the user profile.

Another relevant research was that by Samura and Nishimura [76], in which they conducted a study that examined keystroke dynamics for long free-texts in the Japanese language. In this study, hold time and flight time of Japanese language-specific keystrokes were used as timing features. In order to compare the test and training timing vectors, a weighted Euclidean distance was used. However, although this study was applied to the Japanese language, the keyboard used was an English standard keyboard. Subjects carried-out the typing process by entering the alphabet letters (in English) corresponding to the Japanese characters.

Moreover, until the time of writing this thesis, there has been no reported research that has utilized Arabic input in keystroke dynamics authentication. Therefore, an attempt to incorporate Arabic input in keystroke dynamics user authentication was investigated in this study. The Arabic language has completely different characteristics to those of English, thus using typing patterns for Arabic input to authenticate users is an important research area in the field of keystroke dynamics.

Arabic and English languages are very different to each other. Whereas English is a Germanic language from the Indo-European language family, Arabic is a Semitic language belonging to the Afro-Asiatic language family [277]. Arabic has 28 letters which are completely different from the English alphabet. Moreover, Arabic text is written from right to left which is a unique characteristic for merely Arabic and Hebrew scripts [278]. In addition, there is no distinction between upper and lower case in Arabic. Punctuation rules are much looser than those in English and less commonly used [279].

In the study conducted in this section, the extended key-pairing scheme (developed in this research as discussed in Section 3.3) is tested using text typed in Arabic. Moreover, SVMs were used for classification in association with the ACO feature selection technique. In parallel for comparison, DTs were also exploited to classify the typing data provided in the Arabic language.

6. 2 Feature Definition

The features used for applying the key-pairing scheme to Arabic input are described in this section. The way in which the key-pairs are formed and the timing features are extracted is similar to that followed for English input. Yet, an explanation is provided in this section in order to showcase the Arabic language characteristics and how the key-pairing method was applied to Arabic text.

6.2.1 Key-pair Formation

The text was classified into the same five categories of key-pair relationships, namely: adjacent, second adjacent, third adjacent, fourth adjacent and none adjacent. The complete description of this grouping was introduced in Section 3.3.2.1. The only difference is that the keyboard used in this study is the standard Arabic keyboard since it is the most commonly used Arabic keyboard [280]. The Arabic standard keyboard have the same layout as the QWERTY keyboard used in the English input experiment with the only difference being the characters produced by each key. Figure 6.1 illustrates the key relationship concept when considering the key 'ل' on the Arabic standard keyboard.

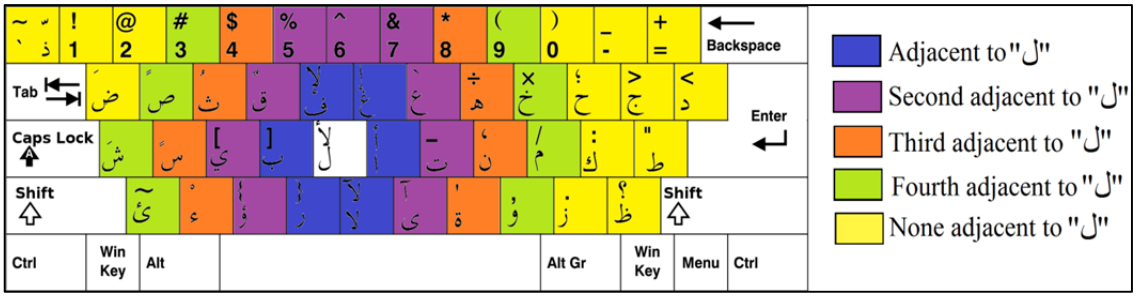

Figure 6.1: Key-pair relationship formation on Arabic keyboard.

These relationship categories can also fall into one of three overall locations on the keyboard (described in Section 3.3.2.1): keys on right side, keys on left side and keys on different sides. An example of the overall locations on the Arabic keyboard is provided in Figure 6.2.

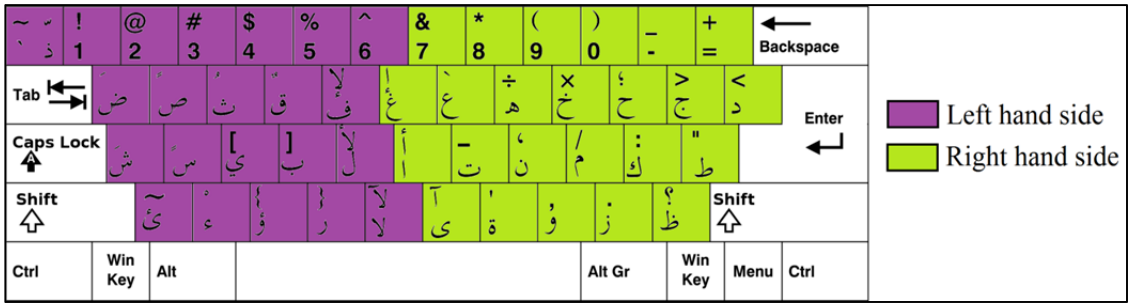

Figure 6.2: Overall key location on Arabic keyboard.

Similar to English input, there are fifteen different combinations of key-pairs in total that any two keys can be classified into. As an example, the key-pairs forming the text "نتلائم" is demonstrated. The text is entered from right to left as the following sequence: "ن ت لا ئ م". The key-pairs are:

- "ن ت": Adjacent/RightSide.
- "ت لا": SecondAdjacent/ DifferentSide.
- "لا ئ": FourthAdjacent/LeftSide.
- "ئ م": NonAdjacent/DifferentSide

As mentioned in Section 3.3.2.1, the key-pairing method significantly boosts the number of key-pairs that can be found and compared in the training and testing samples. Thus, it is used to increase the reliability of the mean of the timing features which will help in increasing the stability of the timing vectors. And so, it is able to utilize a small amount of typing data in the best possible way.

An example of that using Arabic text is shown in the following training and testing data:

- Training data: "قرد الغابة صغير"
- Testing data: "حوت البحر كبير"

Only two similar key-pairs ("ال" and "ير") can be used to compared the samples in the standard keystroke dynamics authentication process, i.e. without the use of key-pairs [8]. However, this is not the case when using the key-pair method as there are more instances of each key-pair extracted from both the training and testing data.

6.2.2 Feature Extraction

The five features used in the extended key-pairing scheme, described in Section 3.3.2.2, are also used when it is applied to Arabic input. They are H1, H2, UU, DD and UD. Thus, five timing features were defined for each key-pair appearance in the text. This was done for all fifteen types of key-pairs. Therefore, the overall number of timing features is 75 (5 timing features * 15 key-pairs). The means of these features are calculated and stored in the timing vector of each user, i.e. user's profile. Table 3.8 in Chapter 3 lists all the 75 features extracted from all key-pairs.

6. 3 Experimental Results and Discussion

This section describes the experimentation process, in which the data collection, data space and the experimental results are specified. A discussion about the experiment results and a comparison with the results produced by English input is provided as well.

6.3.1 Data Collection

A total of twenty-one users participated in this study for data collection. All participants were native Arabic language speakers. They had different levels of typing skills that varied between moderate and very good. They all were accustomed to typing in Arabic but not all of them often type in English. Seven of the participants were involved in the English input study conducted in Section 3.3. Moreover, there were participants from both genders and all the participants were in the age group between 18 and 45 years old. Table 6.1 describes some of the demographic characteristics of the participants.

Table 6.1: Characteristics of the participants in the Arabic language experiment.

Gender		**Age**			**Native language**		**Typing skills**		
Male	**Female**	**18-27**	**28-37**	**38+**	**English**	**Non-English**	**Good**	**Moderate**	**Poor**
4	17	8	11	2	0	21	14	7	0

During data collection, the participants were asked to perform two typing tasks. The tasks involved copying Arabic text that consisted of around 900 characters. The text employed was an excerpt from an Arabic online newspaper. The text included, in addition to letters, numbers and punctuation marks.

Data collection was performed in a similar manner to that described in Section 3.3.5.1. Figure 6.3 illustrates a screenshot of the data collection program demonstrating the Arabic text appearing when the first typing task is clicked. Moreover, attributes captured for every key action performed on the keyboard is exactly the same as that captured for every key action in English input.

As the native language in the country this research is performed at is not Arabic, twelve out of the twenty-one participants were recruited from outside of the country. The data collection program was e-mailed to the non-local participants and the data was also e-mailed back to the researcher when the data collection process was completed by the participants.

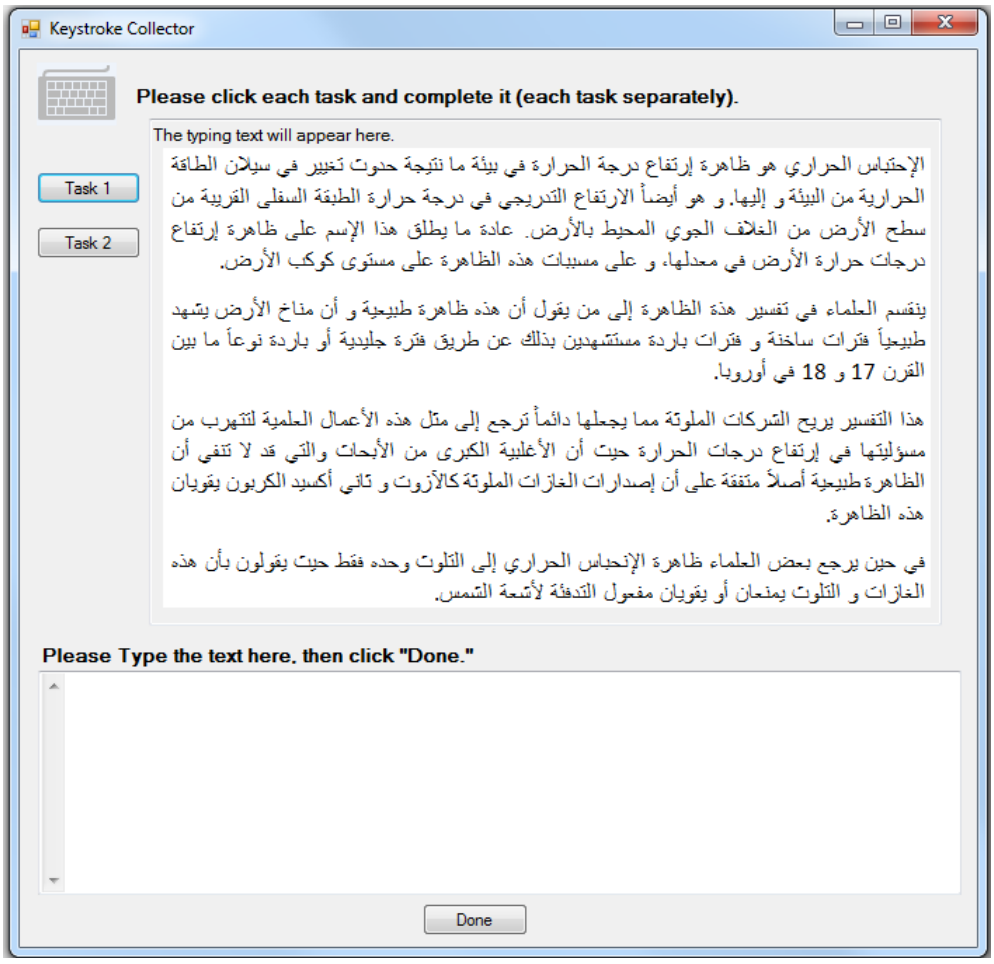

Figure 6.3: A screen-shot of the data collection program for the Arabic experiment.

6.3.2 Data Space

The data space in this experiment is similar to that discussed in Section 3.3.5.2. Although 15 key-pairs from which 75 timing features were captured, there were not enough instances that

appeared in the used text for some of the key-pairs. This made it unfeasible to include them in the final features set. The omitted key-pairs (with less than 10 instances) were: ThirdAdjacent/LeftSide, FourthAdjacent/LeftSide, NonAdjacent/RightSide and NonAdjacent/LeftSide. In total, 20 timing features were excluded from the final feature set; resulting in the inclusion of only 55 features in the final feature vector. It has to be noted that two of the key-pairs removed from the Arabic experiment's features set are different to those removed from the English experiment's features set (found in Section 3.3.5.2). Table 6.2 lists the final 55 features set extracted from all key-pairs in the Arabic experiment.

Table 6.2: Final features set for the Arabic experiment.

Key-pair Category	**Feature Set**				
Adjacent/RightSide	AR-H1	AR-H2	AR-DD	AR-UU	AR-UD
Adjacent/LeftSide	AL-H1	AL-H2	AL-DD	AL-UU	AL-UD
Adjacent/DifferentSide	AD-H1	AD-H2	AD-DD	AD-UU	AD-UD
SecondAdjacent/RightSide	SR-H1	SR-H2	SR-DD	SR-UU	SR-UD
SecondAdjacent/LeftSide	SL-H1	SL-H2	SL-DD	SL-UU	SL-UD
SecondAdjacent/DifferentSide	SD-H1	SD-H2	SD-DD	SD-UU	SD-UD
ThirdAdjacent/RightSide	TR-H1	TR-H2	TR-DD	TR-UU	TR-UD
ThirdAdjacent/DifferentSide	TD-H1	TD-H2	TD-DD	TD-UU	TD-UD
FourthAdjacent/RightSide	FR-H1	FR-H2	FR-DD	FR-UU	FR-UD
FourthAdjacent/DifferentSide	FD-H1	FD-H2	FD-DD	FD-UU	FD-UD
NonAdjacent/DifferentSide	ND-H1	ND-H2	ND-DD	ND-UU	ND-UD

The final step of data pre-processing involves creating the timing vector and storing it in the database as the user's profile. This process was carried-out by combining the data from the two tasks and then dividing it into eight equal sections. Each one of the eight sections is considered as a single typing sample.

This was done (similar to Section 3.3.5.2) by extracting the feature vector (which included all the instances of that feature) for each of the 55 features from each of the two typing tasks separately. And then the two feature vectors from the two tasks are concatenated to produce 55 large vectors, one for each feature.

Similarly, outlier elimination and data normalization was performed exactly the same as described in Section 3.3.5.2. Around 0.01% of the overall data was considered outliers and therefore discarded form the dataset used for the experiment. Moreover, this small amount of discarded data does not massively affect the final dataset.

After that, each of the large vectors is divided equally into 8 parts. The mean of each feature among these eight divisions is computed and then stored in the corresponding user's timing vector (V). There are eight timing vectors (Vs) for each user which were employed as the user's typing samples. Each single timing vector (V) is represented similar to the manner followed in Equation 3.39 in Chapter 3.

6.3.3 Experiment and Results

After creating users' profiles, feature subset selection is performed using ACO [281] for each user's data. The selected features were then passed to the SVMs machine learning mechanism in order to be used as the basic data for differentiating between classes. Similar to the extended key-pairing method (discussed in Section 3.3), the Radial Basis Kernel multiclass classification SVMs [282] was implemented on MATLAB with the aid of the LIBSVM library [227]. Moreover, the one-vs-one method for Multi-class classification was followed here as it is faster in training compared with one-vs-rest [212]. In addition one-vs-one produced lower error rates compared with one-vs-rest in the English experiment performed in Section 3.3.

DTs, on the other hand, is capable of performing feature selection in the tree building and pruning phase [251]. Therefore, it was fed with the complete set of features as discussed in Section 4.4.3. The Statistics toolbox in MATLAB [248] was used to fit the tree and predict the class of each of the test samples.

Similar to the approach followed in Section 4.4.3, classification for both classifiers was carried-out through the leave-one-out cross-validation. Eight samples were used to perform eight cross-validation experiments. Furthermore, FAR and FRR (described in Section 2.7) were used to infer the system performance, as shown in Table 6.3. FAR and FRR were chosen for reasons discussed in Section 3.2.2.4. The details of each individual's results are shown in Appendix C.5.1.

Table 6.3: System performance using Arabic input.

	FAR	*FRR*
ACO/SVMs	0.0256	0.512
DTs	0.0211	0.423

DTs showed a slight performance advantage over SVMs, as shown in Table 6.3. This can be due to the nature of the data and its ability to transformed into rules applicable for separating the data using decision trees [247].

Moreover, the trade-off between FAR and FRR [250] was noticeable in both the SVMs and DTs cases. The FAR produced lower error rates compared with FRR which corresponds to the importance of protecting users against any imposters' access in an authentication system with high security.

The features selected by the ACO were AR-H2, TR-UD, TD-H1, TD-H2, and FD-UU. Meanwhile, the features selected by DTs were: AL-H1, TR-H2, TR-UU, TR-UD, and ND-H1. The two subsets are different except for one feature: TR-UD. Moreover, three out of the five features selected by both ACO and DTs were duration features. This agrees with the conclusions previously presented in Section 3.3.5.3 in which hold features appear more significant compared with latency features in the key-pairing method applied to English text.

6.3.4 Discussion

The main goal of conducting the key-pairing method's experiment on Arabic text is to investigate the applicability of this method on languages other than English. By applying it on a language that has a different alphabet such as Arabic, the key-pairing method is proved to be language-independent.

In addition, in this section, the same 21 subjects provided English input to be tested using the key-pairing method. This is done to compare the system performance when using text in two different languages produced by the same group of people.

The results of the English input experiment are shown in Table 6.4 (The details of each individual's results are shown in Appendix C.5.2). Similar to Arabic input, DTs produced results that outperformed those produced by SVMs as they produced slightly lower error rates. SVMs used features selected by ACO in the classification stage. These features were:

AL-H1, AL-H2, AD-UU, AD-UD and SR-H2. Meanwhile, with decision trees, feature subset selection was performed in the tree building and pruning stage. The features chosen by DTs were: FR-H1, FR-H2, ND-H1, ND-H2 and ND-DD. Similarly, the majority of features selected by both ACO and DTs are duration features.

Table 6.4: System performance using English input.

	FAR	***FRR***
ACO/SVMs	0.0351	0.702
DTs	0.0307	0.613

It is worth mentioning that there was a difference between the key-pairs in English and Arabic. In Arabic, the number of key-pairs on the right hand side of the keyboard is more than those on the left, as most of the commonly used letters are on the right hand side of the keyboard. English, on the other hand, has a greater number of the most used letters (letters like: e, t, a, s and r) [283] on the left hand side of the keyboard, thus key-pairs from that side are larger in number than those on the right hand side. Due to that, the key-pairs that were used to create the users' Arabic profile have some differences to those used for creating the user's English profile, as mentioned in Section 6.3.2. This is because most of the key-pairs on the right side were included in the Arabic study while most of those on the left side were included in the English study. The key-pairs included in the English experiment can be found in Table 3.10 (in Chapter 3) and the key-pairs included in the Arabic experiment can be found in Table 6.2.

Moreover, Arabic input results were generally better than those based on English input due to the fact that all of the subjects chosen to be part of the experiments of this study were native Arabic speakers and fourteen of the twenty-one subjects do not type in English regularly. As Arabic speakers are used to typing in Arabic, their typing skills have developed in Arabic and they are more familiar with an Arabic keyboard and how to operate it. This provides their typing with enough uniformity to be used to correctly identify the users based on their typing patterns. In addition, subjects who are not familiar with English typing have less familiarity with an English keyboard and typing in that language. Therefore, the absence of English experience has caused non-English natives to lack the typing consistency needed to correctly identify users based on their typing patterns [284].

The same key-pairing approach has been used in the study performed in Section 3.3 on English text, in which the SVMs/ACO method resulted in an FAR of 0.013 and an FRR of 0.384. The performance of that experiment outperformed the English experiment performed in this section. This was because eight of the twenty-five participants in the English experiment performed in Section 3.3 were native English speakers and the rest were very familiar with English and were used to typing in English on a daily basis. This clearly improves the consistency of the typing patterns of such users compared to the subjects in the study performed in this section in which the majority were not familiar with English typing.

It should be remarked that there were twenty samples per person in the experiment performed in Section 3.3 fifteen of which were used for training the 25 users whilst eight samples were used for each of the 21 individuals in the experiment performed in this section, seven of the samples were used for training.

Both the experiments in this section and in Section 3.3 provided results demonstrating the effect that the most commonly used language has on the system performance. Lower error rates are achieved when users are using their native language or a language that they are familiar with. In the experiment performed in this section, the use of Arabic language was shown to achieve higher performance compared with the experiment using English text as all of the subjects involved in the experiment were native to Arabic and are more familiar with typing in Arabic. In contrast, in the study conducted in Section 3.3, the English experiment yielded good results due to the participants being either native to English or more familiar with English typing. Figure 6.4 illustrates the effect that the input language and the participants' familiarity with the language have on the overall system performance.

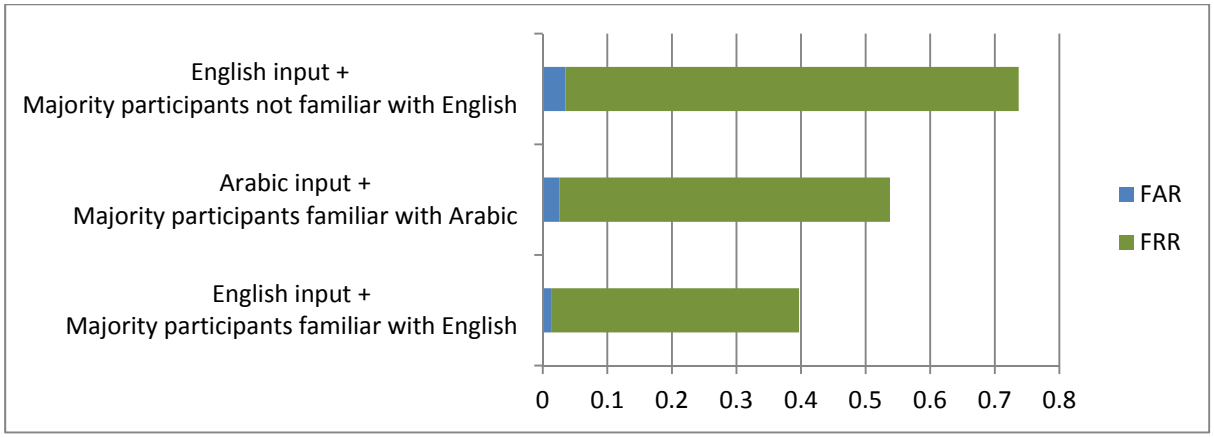

Figure 6.4: Effect of the input language and the participants' familiarity with the language on the error rates.

Similar to the previous studies conducted in Sections 3, 4 and 5, the bias between the FAR and the FRR values is also present in the results for the Arabic input experiment. The large difference between the two error rates has occurred because of the uneven data set used for training as the positive samples were much less than the negative ones [233].

Finally, although the non-conventional features are language-independent, they were hard to extract in the Arabic language. This is due to Arabic not having any distinction between uppercase and lowercase letters. Therefore, it is very rare that a user uses the shift key or the CapsLock key. In addition, the symbols that the shift key produces are very infrequent and they are hardly ever used in Arabic text. Therefore, five of the non-conventional features were not applicable to Arabic. These are CapsLock usage, RSA, RSB, LSA and LSB.

6. 4 Summary

In this part of the thesis, the usefulness of applying free-text keystroke dynamics user authentication on Arabic text was investigated. This was carried-out by using the keyboard's key-layout based method. This key-pairing approach works by classifying every two characters typed consecutively based on their location in relation to each other and their overall location on the keyboard. For each key-pair, five timing features were extracted to be used in the user's feature vector.

SVMs and DTs were employed to classify individuals based on the proposed timing features extracted from Arabic input. The experiment accomplished user authentication based on the smallest amount of training possible. The FAR and FRR rates were both satisfactory with the FAR being the slightly better of the two.

This study proved that the proposed method has been able to authenticate users based on their Arabic typing using reduced training data. The method was originally created to be used with English typing, yet it has successfully crossed over to Arabic input. This proves the language-independency property of the key-pairing method. Moreover, in a comparative study, DTs produced lower error rates compared with SVMs. Duration times also proved to contribute more in increasing the system performance when compared with the latency times.

In addition, experimenting with two languages showed that the user's familiarity with a certain language has a high impact on the user's typing patterns in that language. This considerably affects the system performance as lower error rates are produced from systems incorporating a language native or familiar to the users.

Chapter 7

Conclusions

In this chapter, the overall conclusions with regards to the research performed are summarized. The main contributions of the thesis are illustrated. The limitations faced when conducting this research are also discussed. Future work extending from this research is also presented in this section.

7.1 Summary of Research Findings

Keystroke dynamics is a non-intrusive method for user authentication because it only uses the behavioural data that users convey during regular typing tasks. In addition, it is inexpensive; the only hardware that is required is the standard keyboard. However, a very important benefit of free-text keystroke systems, specifically, is that the typing patterns can still be used for authenticating users even after the authentication phase is over. This is done by extracting the keystrokes for authentication during the course of using the system without intruding into the user's experience. Furthermore, free-text authentication provides a valuable balance between security and usability, as it does not involve any memorization of pre-defined text, which is very desirable by the end user.

This research examined the effectiveness of using an original method, based on the keyboard key-layout, for free-text keystroke dynamics authentication to achieve a relaxed training requirement. This method involved the employment of specific key-pairs from which five timing features are extracted. The main reason for using key-pairs was to reduce the amount of training data the user is required to provide because the need for huge training data is a critical drawback in keystroke systems.

The initial experimentation using the original key-pairing method has produced reasonable results considering the fact that it used merely one short sample of free-text for

authentication, which provided a good balance between the system's security and the user's comfort. Using individual features produced undesired high FAR. However, better FAR was obtained using a combination of more than one feature, yet the FRR was higher using the combined features.

The extended key-pairing method expanded the original key-pairing algorithm used in the original method in order to include a sense of hand positioning. A larger number of features were deployed to find the features that best represented the human typing patterns without losing the advantage of the reduced training requirement of the key-pairing method. ACO and MANOVA were used for the features subset selection. SVMs (both one-vs-one and one-vs-rest multiclass classification) was chosen to be the classifier in this study. The features selected by ACO and MANOVA were dominated by duration times. This corresponds to the duration time having a higher impact on the system performance compared with latency times. The extended method produced lower error rates, more specifically using the features subset extracted using ACO. Nonetheless, the FRR was still higher than desired.

It is hardly surprising that the authentication performance is not perfect, given the nature of this experiment and the fact that its priority is the user's comfort at the enrolment phase. However, this study, which used only a few short training samples, resulted in an FAR that was quite close to most studies found in literature which required much more training. Nevertheless, the FRR results were not as promising, and more work is needed to improve it.

Moreover, non-conventional features were explored. These features include semi-timing features along with editing features. They were extracted from the users' typing stream as an attempt to understand the patterns a user follows when typing a whole piece of text. Both DTs and SVMs were used for classification, with the DTs producing less error rates. The non-conventional features succeeded in reducing the FAR and FRR of the timing features which leads to believing that non-conventional features are slightly more superior compared with the conventional timing ones.

Whilst the timing features and non-conventional features produced encouraging yet not perfect results, the next step in the research was to fuse these two features. This was done in order to combine the two sets of features to achieve a better understanding of the user's typing behaviour. Both feature-level and decision level fusions were implemented. Feature-level fusion saw reduced error rates, yet decision-level fusion succeeded in achieving zero

error rates. Figure 7.1 illustrates the results produced by all experiments performed in this research on English text.

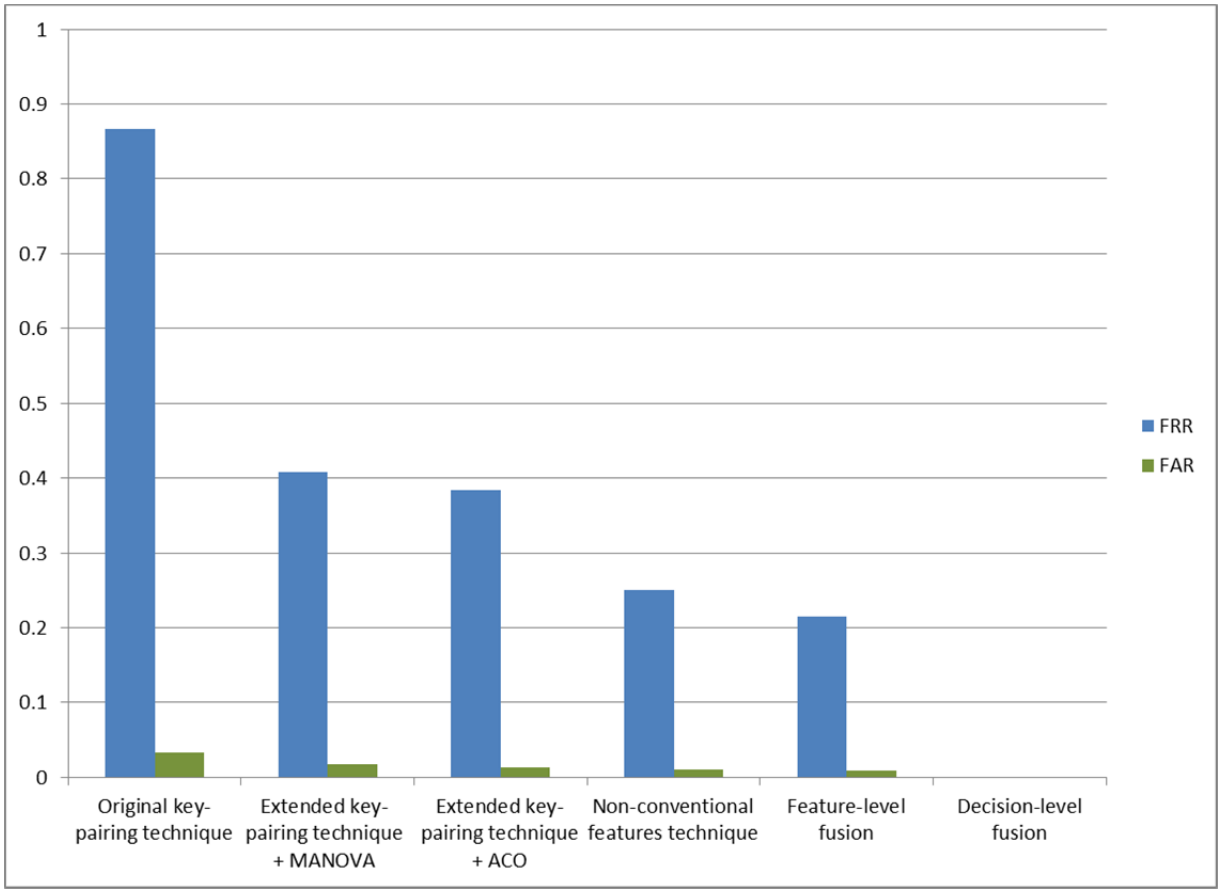

Figure 7.1: Results produced by all experiments performed on English text.

As the key-pairing method was language-independent, this had inspired the investigation of how it can be applied to text in languages other than English. Arabic input was chosen for this test due to the vast differences between the Arabic and English languages. SVMs and DTs were also used in the Arabic input experiment. Similar to English, FAR produced better rates compared with FRR. Arabic and English inputs produced similar results despite the massive differences between the two languages which reveals the ability of the key-pairing method to employ text from different languages.

Moreover, experimenting with these two languages proved that the user's familiarity with a certain language has a noticeable influence on the user's typing patterns in that language, which significantly affects the performance of the authentication system.

In this research, the compromise between the user comfort and the system performance was considerably reduced by using key-pairs that worked to increase the number of di-graphs used for comparing samples in the course of authenticating users. In addition, non-conventional features were also gathered to boost the amount of information the system retains about the users typing behaviour. Therefore, this research is on the right track for creating a simple yet practical system for authenticating users while producing the lowest possible amount of irritation in the training phase.

Furthermore, a trade-off between the FAR and FRR [18] was experienced in all of the studies included in this research. Based on the application of user authentication in real life, there is a huge number of potential impostors that act as a threat to the security of any authentication system [224]. Protecting the system from these risks is very important for applications of a high security nature. This study was successful in achieving that as it protects the system against risk from most imposter's (low FAR). The high security requirement of the authentication system, unfortunately, has caused more legitimate users to be denied access (higher FRR).

Moreover, the biased results illustrated in the noticeable difference between the FAR and FRR values (produced by all studies) is caused by the imbalance of the dataset [232]. The dataset used is considered imbalanced because the positive samples are much fewer than negative samples.

It was stated earlier that comparing the performance of keystroke dynamics systems and therefore determining the method to follow for achieving the best authentication accuracy is not a straightforward task [4]. The reason goes back to the variation of conditions that might be affecting the study; specifically: the participants, environment and procedures included in each study.

An attempt to compare the results produced in this research with results found in similar studies in the field is carried-out (shown in Table 7.1). Nonetheless, there were a great deal of differences between the manner these methods were executed. First, the FAR and FRR are higher in the study performed using the key-pairing method introduced in this research compared by that that produced by Davoudi and Kabir [117], which does not involve any key-pairs. Additionally, the FAR produced by the key-pairing method achieved in this research is lower than that generated by the key-pairing technique introduced by Zahid et al.

[46] to use keystroke dynamics on smart phones. Nonetheless, the FRR was slightly lower in the Zahid et al. technique [46].

Table 7.1: Comparison with state of the art studies.

Study	Participant no.	Characters no.	Features		System performance		
			Convent.	*Non-convent.*	*Accuracy*	*FAR*	*FRR*
Davoudi & Kabir [117] (doesn't involve key-pairs)	21	11700- 13500	√			0.0008	0.188
Zahid et al. [46] (involve key-pairs)	25	12500	√			0.292	0.308
Key-pairing method (developed in this study)	25	7200	√			0.013	0.384
Hempstalk et al. [80]	10	10800- 30000		√		0.113	0.331
Non-conventional features method (developed in this study)	25	7200		√		0.0104	0.25
Curtin et al. [79]	8	15000	√	√	0.985		
Feature-level fusion method (developed in this study)	25	7200	√	√	0.80	0.00896	0.215
Ahmed & Traore [285]	53	11000	√			0.00052	0.0482
Decision-level fusion method (developed in this study)	25	7200	√	√		0.00	0.00

Moreover, the non-conventional features in this research were able to produce lower error rates compared with those produced by the non-conventional features exploited by Hempstalk et al. [80]. In addition, the accuracy produced from fusing the features of the key-pair timing method and the non-conventional features is fairly similar to that delivered by fusing the timing and non-conventional features, in the work conducted by Curtin et al. [79], despite the very low number of participants included in the latter study.

Furthermore, decision-level fusing of the two methods suggested in the study conducted by Ahmed & Traore [258] have produced low error rates, yet the decision-level fusion proposed here succeeded to achieve zero error rates. The number of participants, however, is larger in the Ahmed & Traore study which might contribute to the higher error rate.

It has to be noted, however, that the error rates produced in this research are considered reasonable, despite the requirement for far fewer training data and the much more practical nature of the study. As illustrated in table 7.1, in the studies mentioned for comparison, long

input was collected from users as the system required substantial amounts of data for training. This contradicts with the main goal of this research which is relieving the users from the tedious training input to achieve a user-friendly authentication system. This was achieved as both the key-pairing method and non-conventional features method were developed to perform user authentication using reduced amount of training. Nonetheless, more research is needed to improve the system performance of the key-pairing method and non-conventional methods in order to produce error rates close to that produced by methods that does not involve key-pairs such as the one conducted in [117] whilst satisfying the reduced training requirement.

Finally, keystroke dynamics is faced with a large amount of challenges that have to be overcome in order for it to be an operational biometric for distinguishing between users. Nevertheless, because of its semi-autonomous and cost-effective nature, keystroke dynamics has the potential to develop in the field of information security. Moreover, the idea of using keystroke dynamics is certainly not only restricted to the traditional keyboard; it can be adapted to many other mechanisms, such as ATM machines and cell phones, which provides better everyday protection for the standard user [286].

7.2 Concluding Remarks

The work produced in this research has provided evidence that the key-pairing method was able to reduce the training used for free-text keystroke dynamics authentication. In addition, the use of non-conventional features has produced satisfactory results using the least amount of training possible. Additionally, the fusion of these two types of features has further improved the system performance as the decision-level fusion produced perfect recognition with zero error rates.

Therefore, it is possible to verify the identity of a user by using a method depending on the timing of particular key-pairs on the keyboard as well as other general inputting and editing features. Such authentication will not add-on the burden of long training on the user as it is able to stretch a small amount of training data to its limits and retrieve as much information as possible about the user's typing pattern.

The key-pairing method was capable of adapting input from a language other than English. This was tested by applying the key-pairing method to text in Arabic language. Even though Arabic is a language completely different to English in its characteristics and alphabet,

Arabic text was utilized in a similar way to achieve user authentication. This proves that the key-pairing method is not language dependent. Moreover, experimenting with Arabic and English proved that the user's familiarity with the language has a high impact on his or her typing behaviour which consequently affects the performance of the authentication system.

Using free-text keystroke dynamics in user authentication is considered semi-autonomous and cost-effective. This is due to its nature and the freedom it provides users in choosing any text of any length for the authentication process. Yet more research is particularly needed to improve the performance of the free-text keystroke dynamics system in order to satisfy the need for a reduced training requirement.

Detailed conclusions derived from Chapters 2 to 6 are listed in Table 7.2.

Table 7.2: Conclusions list.

Chapter	Conclusions
Chapter 2 *Literature Review*	- Free-text keystroke dynamics is more user-friendly and more applicable in real life than fixed-text. - Free-text keystroke dynamics is impaired by the huge amount of training it needs. - Keystroke dynamics are influenced by the user's state and by experimental conditions. - Very little research was performed on text in languages other than English.
Chapter 3 *Key-pairing Method*	- The use of combination of features produce better error rates compared with individual timing features. - ACO thrived to select a feature subset that produced lower error rates compared with MANOVA. - Duration time has more impact on the system performance compared to latency times. - One-vs-one SVMs has an advantage over one-vs-rest in the case of this study. - Biased error rates are produced due to the difference between the positive and negative sample sizes.
Chapter 4 *Non-conventional Features*	- Non-conventional features produce error rates slightly superior to those produced by timing features. - DTs classified non-conventional features more effectively than SVMs. - Editing features perform better than semi-timing features in user authentication.
Chapter 5 *Performance Improvement by Fusion*	- Fusion of the key-pairing method and the non-conventional method improved the error rates. - Decision-level fusion produced better results compared with feature-level fusion. - Majority voting produced zero error rates.
Chapter 6 *Exploration of Arabic Language*	- The key-pairing method was successful in authenticating users based on their Arabic typing. - The key-pairing method is language-independent. - The user's familiarity with a language has a high impact on the performance of the authentication system.

7.3 Contributions to Knowledge

This research has produced original contributions in the field of free-text keystroke dynamics authentication. These contributions are concerned with two main issues that require more investigation in this area. Below is a discussion of these two issues and how they were addressed in this research to provide original contributions to the knowledge:

- The need for a large amount of typing data to train the authentication system in the free-text keystroke dynamics area is considered a huge hurdle as it forms a large burden for the users of these systems. Typing large amounts of text in the enrolment phase is time consuming and not user-friendly. This issue is dealt with in this thesis to produce the following contributions:
  - Development of a framework for a keyboard-layout-based key-pairing method.
  - Definition and facilitation of non-conventional features.
  - Implementation of fusion between the key-pair timing features and the non-conventional features.

- There is a lack of language variety in the systems developed previously for free-text keystroke dynamics authentication. Most of the research done in keystroke dynamics had included text only in English. This is addressed by this thesis to achieve the following contribution:
  - Design the key-pairing framework to accept text in any language. This allows for user-authentication to take place even when the text is typed in a language other than English which should appeal to more users.
  - Application of the keyboard-layout-based key-pairing method to Arabic text.

7.4 Limitations and Future Work

It is inevitable to come across some obstacles and hindrances over the duration of any research. This section discusses the challenges that faced this research and how it affected the course of the study. In addition, this section discusses some improvement techniques that have emerged from the work done in this research. These suggested ideas could be implemented in future studies.

Recruiting subjects to participate in the experiments conducted in this research was a hard undertaking. This was especially due to either the subjects' busy schedules or lack of interest. Moreover, the fact that no reward was offered to participants could be a reason for the limited number of participants that completed all the required typing. It is understandable that using a larger database and incorporating data from a greater number of participants will more than likely produce more reliable results. However, the datasets used in this study seem to fairly illustrate the point raised in this research which is concerned with evaluating the efficiency of the newly developed algorithms introduced in this research.

Furthermore, the number of samples collected per subject was limited. Only eight typing tasks were produced by each individual, from which twenty samples and eight samples were extracted for the key-pairing method experiment and the non-conventional features experiment, respectively. The main reason for not including more samples is similar to the previous point; it is hard to interest volunteers to provide a large number of samples. As the accuracy of the system generally escalated when the number of samples in the user's profile increased as proven by [88], this might be a reason for some of the underwhelming results produced by the studies conducted in this research.

Moreover, the biased results produced by most of the studies conducted in this research are due to the imbalanced data set. This noticeable difference between FAR and FRR is mainly caused by having more negative samples than positive samples [232]. This causes the SVMs predicted hyperplane to lie closer to the group with smaller samples, i.e. positive samples. This is the main reason for classifying a new data point as the class with the larger number of samples, i.e. negative class. There are many suggested ways to cope with this problem. One of which is attempting to balance the training set by either duplicating the positive samples or discarding some of the negative samples, especially those close to the positive ones [287]. Another solution includes using different balanced data subsets to train several classifiers [288]. Each of these subsets is created by incorporating all positive samples and only a part (with comparable size) of the negative samples. The final prediction is computed by combining each of the classifiers' predictions.

Determining the performance of a keystroke dynamics system and therefore the best method to follow for achieving the best authentication accuracy is not a straightforward task. Due to the variation of conditions that might be affecting the study participants, environment or procedure, the comparison between two or more methods is not always accurate [4]. Hence,

comparing results from this research with other studies in the current literature is not a direct task.

One concern about keystroke systems is that it tends to be instable, in the sense that it might be influenced by the user's state or by experimental conditions [5]. Indeed, some level of instability might occur even without any obvious reasons. This causes marginally higher error rates in keystroke dynamics systems, which stops it from acting as a user identifier in highly secure systems. However, this can be controlled using adaptation methods [6] that acts as a post-processing step. Adaptation methods allow the initial training data to be extended and updated during the log-in session. That is done by allowing the user to start with a small training sample and gradually build it up, using text typed during the log-in phase. An adaptation method similar to that used in the study conducted in [6] seems to have the desired effect because it adds new timing vectors to the user's profile during log-in time until a predefined number of samples is reached. It then adapts to the user's changing behaviour by swapping the older samples in the user's profile with newer ones.

Moreover, due to its instability, in the way that it is affected by many conditions relating to the user or the experiment environment, it is recommended to use free-text keystroke dynamics in combination with other authentication methods in a multi-factor authentication system [1]. This is done by using free-text keystroke dynamics together with another authentication method such as those discussed in Section 1.2. Properly combining more than one authentication scheme increases the overall performance of the system and compensates for the weakness that one of the methods suffers from [1]. This will allow for a highly secure system whilst fulfilling the low training requirement desired by the users of the keystroke dynamics authentication system.

One of the restrictions of the key-pairing method is its complete reliance on the keyboard's layout. The only type of keyboard layout considered in the data collection process was the QWERTY keyboard as it is the most commonly used keyboard layout [129]. However, other layouts, such as AZERTY and Dvorak [289], are still in use. Therefore, identifying the keyboard layout and adjusting the algorithm used in key-pairing accordingly may help this approach to appeal to a larger audience.

There are two entry-modes followed in this research, i.e. copy-mode and freetyping-mode. A debate about whether the use of these two modes have different effects on the authentication rate is found in literature [49, 130]. Therefore, implementing a comparison between these two

entry modes is important in order to find answers about which entry mode is more adequate to use in research similar to the one conducted here.

Non-conventional features have the property of being language-independent. This could be investigated by extracting these features from text in a language other than English. The tested language, however, have to satisfy a certain condition. It has to include a significant amount of shift key usage. Languages such as Arabic do not fulfil such condition. Therefore, choosing an appropriate language and applying the non-conventional method to it will provide more knowledge about this method's functionality.

Moreover, the fact that the key-pairing scheme is language-independent introduces the incentive for investigating its applicability on text typed in more languages. This can be done by applying it to languages other than English and Arabic. Comparisons can then be performed to understand the similarities and differences between languages in regards to this method's application.

The next step, after applying the key-pairing method to other languages is to investigate its use with samples typed in two different languages. To the date of writing this thesis, only one research study has evaluated users' typing behaviours in a particular language, based on keystroke data collected from another language [88]. However, only languages that have identical alphabets were discussed. Since the key-pairing algorithm depends only on the location of the key on the keyboard, rather than on the actual characters, it is possible to use data typed in English as profile data and, then ask the participants to provide testing samples in another language, provided that both languages are collected using the same keyboard layout. Arabic is a good choice for the second language since it has a completely different set of letters compared with English. An investigation that evaluates the capability of the key-pairing method to authenticate users based on text typed in a language other than that of the training samples will open new horizons for the application of such a method in many real-life conditions.

Applying the key-pairing and non-conventional methods on multiple languages can be facilitated by the common features found in the typing of different languages, such as the use of chords, i.e. pressing more than one key simultaneously to generate additional characters, the opportunity to use the two hands, the possibility to measure key adjacency, the correction of typing mistakes via backspace, the use of non-language keys (such as: arrow keys, escape key, function keys … etc.). Nonetheless, the differences that exist between languages might

introduce a challenge when designing a multiple language system for keystroke dynamics. Differences between languages include, for example, the lack of using punctuation in Arabic and the larger character sets in Chinese. However, uniqueness of these characteristics to a certain language can be considered in single-language systems to extract more distinctive features from users input.

The relationship between the system performance and the user's personal traits could be investigated as well. Personal traits, such as age, gender, native language and computer skills, can be analysed in each user's case to investigate their effect on the success of the method. The initial analysis showed a higher level of consistency in the input produced by the individual with high level of familiarity with the typed language. A more in-depth analysis might be beneficial to provide additional information about this association and others that may exist as well.

References

[1] M. Ciampa, *Security + guide to network security fundamentals*, 4th ed. Thomson Course Technology, 2005.

[2] D. Florencio and C. Herley, "A large-scale study of web password habits," in *the 16th International Conference on World Wide Web (WWW2007)*, 2007, pp. 657–666.

[3] H. K. Sarohi and F. U. Khan, "Graphical password authentication schemes : current status and key issues," *International Journal of Computer Science Issues (IJCSI)*, vol. 10, no. 2, pp. 437–443, 2013.

[4] S. P. Banerjee and D. L. Woodard, "Biometric authentication and identification using keystroke dynamics : a survey," *Journal of Pattern Recognition Research*, vol. 7, no. 1, pp. 116–139, 2012.

[5] M. Karnan and M. Akila, "Personal authentication based on keystroke dynamics using soft computing techniques," in *Second International Conference on Communication Software and Networks*, 2010, pp. 334–338.

[6] R. Giot, M. El-Abed, and C. Rosenberger, "Keystroke dynamics with low constraints SVM based passphrase enrollment," in *IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems*, 2009, pp. 1–6.

[7] S. Haider, A. Abbas, and A. K. Zaidi, "A multi-technique approach for user identification through keystroke dynamics," in *IEEE International Conference on Systems, Man and Cybernetics.*, 2000, vol. 2, pp. 1336–1341.

[8] D. Gunetti and C. Picardi, "Keystroke analysis of free text," *ACM Transactions on Information and System Security*, vol. 8, no. 3, pp. 312–347, Aug. 2005.

[9] C. Vielhuauer, *Biometric user authentication for IT security*. Springer, 2006.

[10] T. Olzak, "Keystroke dynamics: low impact biometric verification biometrics," *Adventures in Security*, pp. 1–10, 2006.

[11] L. Coventry, A. De Angeli, and G. Johnson, "Honest it's me! Self service verification.," in *CHI Workshop on Human-Computer Interaction, Adoption Study and Security Systems, Fort Lauderdale, FL*, 2003, pp. 1–6.

[12] D. Raghu, C. R. Jacob, and Y. V. K. D. Bhavani, "Neural network based authentication and verification for web based key stroke dynamics," *International Journal of Computer Science and Information Technologies*, vol. 2, no. 6, pp. 2765–2772, 2011.

[13] F. Monrose, M. K. Reiter, and S. Wetzel, "Password hardening based on keystroke dynamics," in *the 6th ACM Conference on Computer and Communications Security*, 1999, pp. 73 – 82.

[14] M. Rhodes-Ousley, R. Bragg, and K. Strassberg, *Network security: the complete reference*, 1st ed. McGraw-Hill Osborne Media, 2004.

[15] P. Narainsamy and K. M. S. Soyjaudah, "Minimizing the number of retry attempts in keystroke dynamics through inclusion of error correcting schemes," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 8, no. 7, pp. 19–25, 2010.

[16] K. R. Allendoerfer and S. Pai, "Human factors considerations for passwords and other user identification techniques part 2 : field study , results and analysis," *William J. Hughes Technical Center, Tech. Rep. DOT/FAA/TC- 06/09*, 2006.

[17] H. Rossnagel, J. W. G. Frankfurt, and D. Royer, "Investing in security solutions - can qualified electronic signatures be profitable for mobile operators ?," in *the Eleventh Americas Conference on Information Systems*, 2005, pp. 3248–3257.

[18] D. Shanmugapriya and G. Padmavathi, "A survey of biometric keystroke dynamics : approaches , security and challenges," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 5, no. 1, pp. 115–119, 2009.

[19] M. Karnan, M. Akila, and N. Krishnaraj, "Biometric personal authentication using keystroke dynamics: a review," *Applied Soft Computing*, vol. 11, no. 2, pp. 1565–1573, 2011.

[20] M. Zviran and W. J. Haga, "Cognitive passwords: the key to easy access control," *Computers and Security*, vol. 9, no. 8, pp. 723–736, 1990.

[21] D. Kirovski, N. Jojic, and P. Roberts, "Click passwords," in *21st International Information Security Conference (SEC)*, 2006, pp. 351–363.

[22] D. Davis, F. Monrose, and M. K. Reiter, "On user choice in graphical password schemes," in *the USENIX Security Symposium*, 2004.

[23] J. Brainard, A. Juels, R. L. Rivest, M. Szydlo, and M. Yung, "Fourth-factor authentication : somebody you know," in *the 13th ACM Conference on Computer and Communications Security*, 2006, pp. 168–178.

[24] D. E. Denning and P. F. MacDoran, "Location-based authentication : grounding cyberspace for better security," *Computer Fraud & Security*, pp. 12–16, 1996.

[25] H. A. Crawford, "A framework for continuous, transparent authentication on mobile devices," *PhD thesis, School of Computing Science, College of Science and Engineering, University of Glasgow*, 2012.

[26] O. S. Adeoye, "Evaluating the performance of two-factor authentication solution in the banking sector," *International Journal of Computer Science Issues*, vol. 9, no. 4, pp. 457–462, 2012.

[27] C. Chantan, S. Sinthupinyo, and T. Rungkasiri, "Improving accuracy of authentication process via text using Bayesian network," *International Journal of Computer Science Issues*, vol. 9, no. 2, pp. 10–16, 2012.

[28] S. Wiedenbeck, J. Waters, J. C. Birget, A. Brodskiy, and N. Memon, "PassPoints: design and longitudinal evaluation of a graphical password system," *International Journal of Human Computer Studies*, vol. 63, no. 1–2, pp. 102–127, 2005.

[29] B. Vanathi, K. Shanmugam, and V. R. Uthairaj, "A secure m-commerce architecture for service provider to improvize quantity and quality of the products using fingerprint authentication and gender classification," *Asian Journal of Information Technology*, vol. 15, no. 2, pp. 232–242, 2016.

[30] E. Vazquez-Fernandez and D. Gonzalez-Jimenez, "Face recognition for authentication on mobile devices," *Image and Vision Computing*, pp. 10–12, 2016.

[31] V. Tulceanu, "Comprehensive brainwave authentication using emotional stimuli," in *20th European Signal Processing Conference (EUSIPCO12)*, 2012, pp. 1772–1776.

[32] C. Hegde, H. R. Prabhu, D. S. Sagar, P. D. Shenoy, K. R. Venugopal, and L. M. Patnaik, "Heartbeat biometrics for human authentication," *Signal, Image and Video Processing*, vol. 5, no. 4, pp. 485–493, 2011.

[33] F. Monrose and A. Rubin, "Authentication via keystroke dynamics," in *4th ACM Conference on Computer and Communications security*, 1997, pp. 48–56.

[34] D. Umphress and G. Williams, "Identity verification through keyboard characteristics," *Man–Machine Studies*, vol. 23, no. 3, pp. 263–273, 1985.

[35] H. Barghouthi, "Keystroke dynamics – how typing characteristics differ from one application to another," *Master's Thesis, Department of Computer Science and Media Technology, Gjovik University College*, 2009.

[36] R. Joyce and G. Gupta, "Identity authentication based on keystroke latencies," *Communications of the ACM*, vol. 33, no. 2, pp. 168–176, Feb. 1990.

[37] R. J. Spillane, "Keyboard apparatus for personal identification.," *IBM Technical Disclosure Bulletin*, vol. 17, pp. 33–46, 1975.

[38] G. Forsen, M. Nelson, and R. Staron, "Personal attributes authentication techniques," *Rome Air Development Centre, Tech. Rep. RADC-TR-77-333*, 1977.

[39] R. Gaines, W. Lisowski, S. Press, and N. Shpiro, "Authentication by keystroke timing : some preliminary results," *Rand Corporation, Santa Monica, CA, USA, Tech. Rep. R-256-NSF*, 1980.

[40] J. Modi, H. G. Upadhyay, and M. Thakor, "Password less authentication using keystroke dynamics : a survey," *International Journal of Innovative Research in Computer and Communication Engineering*, pp. 7060–7064, 2014.

[41] K. S. Killourhy, "A scientific understanding of keystroke dynamics," *PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, USA*, 2012.

[42] A. Schlenker and M. Sarek, "Behavioural biometrics for multi-factor authentication in biomedicine," *European Journal for Biomedical Informatics*, vol. 8, no. 5, pp. 19–24, 2012.

[43] 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," in *the International Joint Conference on Biometrics*, 2011, pp. 1–8.

[44] P. Bours, “Continuous keystroke dynamics: a different perspective towards biometric evaluation,” *Information Security Technical Report*, vol. 17, no. 1–2, pp. 36–43, Feb. 2012.

[45] E. Lau, X. Liu, C. Xiao, and X. Yu, “Enhanced user authentication through keystroke biometrics,” *Computer and Network Security*, vol. 6, 2004.

[46] S. Zahid, M. Shahzad, S. A. Khayam, and M. Farooq, “Keystroke-based user identification on smart phones,” in *the 12th International Symposium on Recent Advances in Intrusion Detection (RAID ’09)*, 2009, pp. 224–243.

[47] P. M. Lee, W. H. Tsui, and T. C. Hsiao, “The influence of emotion on keyboard typing: an experimental study using auditory stimuli,” *Plos One*, vol. 10, no. 6, 2015.

[48] P. S. Teh, A. J. Teoh, and S. Yue, “A survey of keystroke dynamics biometrics,” *The ScientificWorld Journal*, 2013.

[49] M. Villani, C. Tappert, G. Ngo, J. Simone, H. St. Fort, and S. Cha, “Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions,” in *the IEEE Computer Society Workshop on Biometrics*, 2006.

[50] S. Sing and K. V. Arya, “Key classification : a new approach in free text keystroke authentication system,” in *Third Pacific-Asia Conference on Circuits, Communications and System*, 2011, pp. 1–5.

[51] C. M. Tey, P. Gupta, and D. Gao, “Keystroke biometrics : the user perspective,” in *the 4th ACM conference on Data and application security and privacy*, 2013, pp. 289–296.

[52] R. Giot, M. El-Abed, B. Hemery, and C. Rosenberger, “Unconstrained keystroke dynamics authentication with shared secret,” *Computers & Security*, vol. 30, no. 6, pp. 427–445, Sep. 2011.

[53] J. Garcia, “Personal identification apparatus,” *U.S. Patent 4621334*, 1986.

[54] M. S. Obaidat and B. Sadoun, “Verification of computer users using keystroke dynamics,” *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics*, vol. 27, no. 2, pp. 261–9, Jan. 1997.

[55] C. Jiang, S. Shieh, and J. Liu, “Keystroke statistical learning model for web authentication,” in *the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS ’07)*, 2007, pp. 359–361.

[56] M. Rybnik, P. Panasiuk, and K. Saeed, “User authentication with keystroke dynamics using fixed text,” in *International Conference on Biometrics and Kansei Engineering*, 2009, pp. 70–75.

[57] N. Bartlow and B. Cukic, “Evaluating the reliability of credential hardening through keystroke dynamics,” in *the 17th International Symposium on Software Reliability Engineering*, 2006, pp. 117–126.

[58] D. X. Song, D. Wagner, and X. Tian, “Timing analysis of keystrokes and timing attacks on SSH,” in *the 10th USENIX Security Symposium*, 2001, pp. 337–352.

[59] S. Mandujano and R. Soto, “Deterring password sharing: user authentication via fuzzy c-means clustering applied to keystroke biometric data,” in *the Fifth Mexican International Conference in Computer Science*, 2004, pp. 181–187.

[60] S. F. P. Downland, “A long-term trail of keystroke profiling using digraph, trigraph and keyword latencies,” in *IFIP/SEC 19th International Conference on Information Security*, 2004, pp. 275–289.

[61] S. J. Shepherd, “Continuous authentication by analysis of keyboard typing characteristics,” in *European Convention on Security and Detection*, 1995, pp. 111–114.

[62] J. D. Marsters, “Keystroke dynamics as a biometric,” *PhD thesis, Faculty of Engineering, science and mathematics, School of electronics and computer science, University of Southampton*, 2009.

[63] S. Furnell, J. Morrissey, P. Sanders, and C. Stockel, “Applications of keystroke analysis for improved login security and continuous user authentication,” in *the Information and System Security Conference*, 1996.

[64] H. Saevanee and P. Bhattarakosol, “Authenticating user using keystroke dynamics and finger pressure,” in *the 6th IEEE Consumer Communications and Networking Conference*, 2009, pp. 1–2.

[65] N.M.Gunathilake, A.P.B.Padikaraarachchi, S.P.Koralagoda, M.G.Jayasundara, P.A.I.M.Paliyawadana, C. D. Manawadu, and U.U.S. Rajapaksha, “Enhancing the security of online banking systems via keystroke dynamics,” in *the 8th International Conference on Computer Science & Education*, 2013, pp. 561–566.

[66] L. C. F. Araujo, L. H. R. Sucupira, and M. G. Lizarraga, “User authentication through typing biometrics features,” *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 851–855, 2005.

[67] K. S. Killourhy and R. A. Maxion, “Comparing anomaly-detection algorithms for keystroke dynamics,” in *IEEE/IFIP International Conference on Dependable Systems & Networks*, 2009, pp. 125–134.

[68] N. Bakelman, J. V. Monaco, S. H. Cha, and C. C. Tappert, “Continual keystroke biometric authentication on short bursts of keyboard input,” in *Pace University CSIS Research Day*, 2012, vol. 1, pp. 1–7.

[69] S. Park, J. Park, and S. Cho, “User authentication based on keystroke analysis of long free texts with a reduced number of features,” in *Second International Conference on Communication Systems, Networks and Applications*, 2010, pp. 433–435.

[70] D. Stefan and D. Yao, “Keystroke-dynamics authentication against synthetic forgeries,” in *the 6th International ICST Conference on Collaborative Computing: networking, Applications, Worksharing*, 2010, pp. 1–8.

[71] D. Gunetti and G. Ruffo, “Intrusion detection through behavioural data,” in *the Third International Symposium on Advances*, 1999, pp. 383–394.

[72] P. S. Teh, A. B. J. Teoh, T. S. Ong, and H. F. Neo, "Statistical fusion approach on keystroke dynamics," in *Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, 2007, pp. 918 – 923.

[73] F. Bergadano, D. Gunetti, and C. Picardi, "User authentication through keystroke dynamics," *ACM Transactions on Information and System Security*, vol. 5, no. 4, pp. 367–397, 2002.

[74] W. Bond and A. Awad E.A., "Touch-based static authentication using a virtual grid," in *the 3rd ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '15)*, 2015, pp. 129–134.

[75] T. Sim and R. Janakiraman, "Are digraphs good for free-text keystroke dynamics?," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–6.

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

[77] P. Bours and H. Barghouthi, "Continuous authentication using biometric keystroke dynamics," in *the Norwegian Information Security Conference*, 2009, pp. 1–12.

[78] P. S. Dowland, H. Singh, and S. M. Furnell, "A preliminary investigation of user authentication using continuous keystroke analysis," in *the IFIP 8th Annual Working Conference on Information Security Management and Small Systems Security*, 2001.

[79] M. Curtin, C. Tappert, M. Villani, G. Ngo, J. Simone, H. S. Fort, and S. Cha, "Keystroke biometric recognition on long-text input : a feasibility study," in *IMECS*, 2006, pp. 1–5.

[80] K. Hempstalk, E. Frank, and I. H. Witten, "One-class classification by combining density and class probability estimation," in *the European Conference on Machine and Learning and Principles and Practice of Knowledge Discovery in Database*, 2005, pp. 505–519.

[81] H. R. Lv, Z. L. Lin, W. J. Yin, and J. Dong, "Emotion recognition based on pressure sensor keyboards," in *IEEE International Conference on Multimedia and Expo*, 2008, pp. 1089–1092.

[82] S. S. Shen, S. H. Lin, T. H. Kang, and W. Chien, "Enhanced keystroke dynamics authentication utilizing pressure detection," in *International Conference on Applied System Innovation (ICASI)*, 2016, pp. 1–4.

[83] G. C. Boechat, J. C. Ferreira, and E. C. B. C. Filho, "Investigation about authentication biometrics using keystroke dynamics," in *Biometrics Symposium*, 2007, pp. 11–15.

[84] S. Bajaj and S. Kaur, "Typing speed analysis of human for password protection (based on keystrokes dynamics)," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 3, no. 2, pp. 88–91, 2013.

[85] J. Roth, X. Liu, and D. Metaxas, "On continuous user authentication via typing behavior," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4611–4624, 2014.

[86] L. Hansen and L. Willmore, "Keystroke rhythm and intensity as biometrics for user ID," *Project Checkpoint – CS229, Stanford University*, pp. 1–5, 2013.

[87] G. Pahuja and T. N. Nagabhushan, “Biometric authentication and identification through behavioral biometrics: a survey,” in *International Conference on Cognitive Computing and Information Processing(CCIP)*, 2015, pp. 1–7.

[88] D. Gunetti, C. Picardi, and G. Ruffo, “Keystroke analysis of different languages : a case study,” in *Advances in Intelligent Data Analysis VI*, no. 2, 2005, pp. 133–144.

[89] J. R. M. Filho and E. O. Freire, “On the equalization of keystroke timing histograms,” *Pattern Recognition Letters*, vol. 27, no. 13, pp. 1440–1446, Oct. 2006.

[90] R. Giot, A. Ninassi, M. El-Abed, and C. Rosenberger, “Analysis of the acquisition process for keystroke dynamics,” in *International Conference of the Biometrics Special Interest Group, Darmstadt*, 2012.

[91] A. A. E. Ahmed, I. Traore, and A. Almulhem, “Digital fingerprinting based on keystroke dynamics,” in *the Second International Symposium on Human Aspects of Information Security and Assurance*, 2008.

[92] R. Saifan, A. Salem, D. Zaidan, and A. Swidan, “A Survey of behavioral authentication using keystroke dynamics: touch screens and mobile devices,” *Journal of Social Sciences*, vol. 55, no. 11, pp. 29–41, 2016.

[93] S. K. Card, T. P. Moran, and A. Newel, “The keystroke-level model for user performance time with interactive systems,” *Communications of the ACM*, vol. 23, pp. 396–410, 1980.

[94] W. E. Cooper, *Cognitive aspects of skilled typewriting*. Springer-Verlag, 1983.

[95] P. M. Fitts, “The information capacity of the human motor system in controlling the amplitude of movement.,” *Journal of Experimental Psychology*, vol. 47, no. 6, pp. 381–391, 1954.

[96] R. W. Soukoreff and I. S. MacKenzie, “Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts’ law research in HCI,” *International Journal of Human Computer Studies*, vol. 61, no. 6, pp. 751–789, 2004.

[97] H. Zhao, “Fitts’ law: modeling movement time in HCI,” *Theories in Computer human interaction - University of Maryland*, 2002. [Online]. Available: https://www.cs.umd.edu/class/fall2002/cmsc838s/tichi/fitts.html. [Accessed: 02-Dec-2016].

[98] S. Zhai, M. Hunter, and B. a. Smith, “The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design,” in *the 13th Annual ACM Symposium on User Interface Software and Technology*, 2000, vol. 2, pp. 119–128.

[99] S. Bleha, C. Slivinsky, and B. Hussien, “Computer-access security systems using keystroke dynamics,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 12, pp. 1217–1222, 1990.

[100] M. S. Obaidat and D. T. Macchairolo, “A multilayer neural network system for computer access security,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 5, pp. 806–813, 1994.

[101] W. G. d. Ru and J. H. P. Eloff, “Enhanced password authentication through fuzzy logic,” *IEEE Expert*, vol. 12, no. 6, pp. 38–45, 1997.

[102] S. Cho and S. Hwang, “Artificial rhythms and cues for keystroke dynamics based authentication,” in *IAPR International Conference on Biometrics*, 2006.

[103] S. Hwang, H. Lee, and S. Cho, “Improving authentication accuracy using artificial rhythms and cues for keystroke dynamics-based authentication,” *Expert Systems with Applications*, vol. 36, no. 7, pp. 10649–10656, 2009.

[104] R. A. Maxion and K. S. Killourhy, “Keystroke biometrics with number-pad input,” in *EEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, 2010, pp. 201–210.

[105] J. Liu, B. Zhang, H. Zeng, L. Shen, J. Liu, and J. Zhao, “The BeiHang keystroke dynamics systems, databases and baselines,” *Neurocomputing*, vol. 144, pp. 271–281, 2014.

[106] D. T. Lin, “Computer-access authentication with neural network based keystroke\nidentity verification,” in *International Conference on Neural Networks (ICNN’97)*, 1997, vol. 1, pp. 174–178.

[107] J. A. Robinson, V. M. Liang, J. A. M. Chambers, and C. L. MacKenzie, “Computer user verification using login string keystroke dynamics,” *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 28, no. 2, pp. 236–241, 1998.

[108] S. Modi and S. J. Elliott, “Keystroke dynamics verification using a spontaneously generated password,” in *International Carnahan Conference on Security Technology*, 2006, pp. 116–121.

[109] Z. Jin, A. B. J. Teoh, T. S. Ong, and C. Tee, “Typing dynamics biometric authentication through fuzzy logic,” in *International Symposium on Information Technology*, 2008, pp. 1–6.

[110] S. D. Abualgasim and I. Osman, “An application of the keystroke dynamics biometric for securing PINs and passwords,” *World of Computer Science and Information Technology Journal (WCSIT)*, vol. 1, no. 9, pp. 398–404, 2011.

[111] H. B. Kekre, V. A. Bharadi, P. Shaktia, V. Shah, and A. A. Ambardekar, “Keystroke dynamic analysis using relative entropy & timing sequence Euclidian distance,” in *the International Conference & Workshop on Emerging Trends in Technology (CWET ’11)*, 2011, pp. 220–223.

[112] Y. Zhong, Y. Deng, and A. K. Jain, “Keystroke dynamics for user authentication,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2012, pp. 117–123.

[113] A. Rezaei and S. Mirzakochaki, “A novel approach for keyboard dynamics authentication based on fusion of stochastic classifiers,” *International Journal of Computer Science and Network Security*, vol. 12, no. 8, pp. 60–68, 2012.

[114] P. Bours and V. Komanpally, “Performance of keystroke dynamics when allowing typing corrections,” in *International Workshop on Biometrics and Forensics (IWBF)*, 2014, no. 2, pp. 1–6.

[115] J. Hu, D. Gingrich, and A. Sentosa, “A k-nearest neighbor approach for user authentication through biometric keystroke dynamics,” in *IEEE International Conference on Communications*, 2008, pp. 1556–1560.

[116] H. Davoudi and E. Kabir, “A new distance measure for free text keystroke authentication,” in *14th International CSI Computer Conference*, 2009, pp. 570–575.

[117] H. Davoudi and E. Kabir, “User authentication based on free text keystroke patterns,” in *the 3rd Joint Congress on Fuzzy and Intelligent Systems*, 2010.

[118] H. Davoudi and E. Kabir, “Modification of the relative distance for free text keystroke authentication,” in *5th International Symposium on Telecommunications*, 2010, pp. 547–551.

[119] A. Darabseh and A. Siami Namin, “On accuracy of keystroke authentications based on commonly used English words,” in *International Conference of the Biometrics Special Interest Group (BIOSIG)*, 2015, pp. 1–8.

[120] R. Janakiraman and T. Sim, “Keystroke dynamics in a general setting,” in *Advances in Biometrics*, 2007, pp. 584–593.

[121] T. Buch, A. Cotoranu, E. Jeskey, F. Tihon, M. Villani, and N. York, “An enhanced keystroke biometric system and associated studies,” in *Pace University CSIS Research Day*, 2008, pp. 1–7.

[122] K. Pilsung, P. Jooseong, P. Sunghoon, Y. Joonha, and C. Sungzoon, “Keystroke dynamics analysis based on long and free text,” in *Fall Korean Industrial Engineering Conference*, 2009.

[123] J. V. Monaco, J. C. Stewart, S. Cha, and C. C. Tappert, “Behavioral biometric verification of student identity in online course assessment and authentication of authors in literary works,” in *IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, 2013, pp. 1–8.

[124] P. Kang and S. Cho, “Keystroke dynamics-based user authentication using long and free text strings from various input devices,” *Information Sciences*, vol. 308, pp. 72–93, 2014.

[125] Y. Matsubara, T. Samura, and H. Nishimura, “Profile generation methods for reinforcing robustness of keystroke authentication in free text typing,” *Journal of Information Security*, vol. 6, no. 2, pp. 131–141, 2015.

[126] P. Panasiuk, M. Rybnik, K. Saeed, and M. Rogowsk, “The impact of database quality on keystroke dynamics authentication,” in *International Conference Of Numerical Analysis And Applied Mathematics*, 2016.

[127] I. Buciu and A. Gacsadi, “Biometrics Systems and Technologies : a survey,” *International Journal Of Computers Communications & Control*, vol. 11, no. 3, pp. 315–330, 2016.

[128] R. Giot, B. Dorizzi, and C. Rosenberger, “A review on the public benchmark databases for static keystroke dynamics,” *Computers & Security*, vol. 55, pp. 46–61, 2015.

[129] J. Noyes, “The QWERTY keyboard : a review,” *International Journal of Man-Machine Studies*, vol. 18, pp. 265–281, 1983.

[130] K. S. Killourhy and R. A. Maxion, “Free vs. transcribed text for keystroke-dynamics evaluations,” in *ACM International Conference Proceeding Series*, 2012, pp. 1–8.

[131] J. Montalvão, E. O. Freire, M. a. Bezerra Jr., and R. Garcia, "Contributions to empirical analysis of keystroke dynamics in passwords," *Pattern Recognition Letters*, vol. 52, pp. 80–86, 2015.

[132] S. Mondal, P. Bours, and S. Idrus, "Complexity measurement of a password for keystroke dynamics: preliminary study," in *the 6th International Conference on Security of Information and Networks*, 2013, pp. 3–7.

[133] X. Ke, R. Manuel, M. Wilkerson, and L. Jin, "Keystroke dyamics: a web-based biometric solution," in *the 13th USENIX Security Symposium*, 2004.

[134] S. Hocquet, J. Ramel, and H. Cardot, "Fusion of methods for keystroke dynamic authentication," in *Fourth IEEE Workshop on Automatic Identification Advanced Technologies*, 2005, pp. 224–229.

[135] R. Giot, M. El-Abed, and C. Rosenberger, "Web-based benchmark for keystroke dynamics biometric systems: a statistical analysis," *Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 11–15, Jul. 2012.

[136] D. Y. Liliana and D. Satrinia, "Adaptive behaviometrics using dynamic keystroke for authentication system," in *International Conference on Future Information Technology (IPCSIT)*, 2011, vol. 13, pp. 191–195.

[137] S. Chauhan and K. V. Prema, "Effect of dimensionality reduction on performance in artificial neural network for user authentication," in *the 3rd IEEE International Advance Computing Conference (IACC13)*, 2013, pp. 788–793.

[138] I. De Mendizabal-Vazquez, D. De Santos-sierra, J. Guerra-casanova, and S. Carmen, "Supervised classification methods applied to keystroke dynamics through mobile devices," in *International Carnahan Conference on Security Technology (ICCST)*, 2014.

[139] P. H. Pisani and A. C. Lorena, "Adaptive approaches for keystroke dynamics," *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2015.

[140] S. Douhou and J. R. Magnus, "The reliability of user authentication through keystroke dynamics," *Statistica Neerlandica*, vol. 63, no. 4, pp. 432–449, 2009.

[141] D. Gunetti, C. Picardi, and G. Ruffo, "Dealing with different languages and old profiles in keystroke analysis of free text," in *the Advances in Artificial Intelligence*, 2005, pp. 347–358.

[142] D. Jamil and M. A. Khan, "Keystroke pattern recognition preventing online fraud," *International Journal of Engineering Science and Technology (IJEST)*, vol. 3, no. 3, pp. 1953–1958, 2011.

[143] C. Epp, M. Lippold, and R. L. Mandryk, "Identifying emotional states using keystroke dynamics," in *the Annual Conference on Human Factors in Computing Systems (CHI '11)*, 2011, pp. 715–724.

[144] R. Bixler and S. D'Mello, "Detecting boredom and engagement during writing with keystroke analysis, task appraisals, and stable traits," in *International Conference on Intelligent User Interfaces (IUI '13)*, 2013, pp. 225–233.

[145] L. Vizer, “Different strokes for different folks: individual stress response as manifested in typed text,” in *Extended Abstracts on Human Factors in Computing Systems*, 2013, pp. 2773–2778.

[146] A. K. Jain, S. C. Dass, and K. Nandakumar, “Can soft biometric traits assist user recognition?,” in *Defense and Security*, 2004, pp. 561–572.

[147] M. Fairhurst and M. Da Costa-Abreu, “Using keystroke dynamics for gender identification in social network environment,” in *4th International Conference on Imaging for Crime Detection and Prevention (ICDP 2011)*, 2011, pp. 1–6.

[148] R. Giot and C. Rosenberger, “A new soft biometric approach for keystroke dynamics based on gender recognition,” *International Journal of Information Technology and Management*, vol. 11, no. 1–2, pp. 35–49, 2012.

[149] S. Z. Syed Idrus, E. Cherrier, C. Rosenberger, and P. Bours, “Soft biometrics for keystroke dynamics: profiling individuals while typing passwords,” *Computers & Security*, vol. 45, pp. 147–155, 2014.

[150] D. G. Brizan, A. Goodkind, P. Koch, K. Balagani, V. V Phoha, and A. Rosenberg, “Utilizing linguistically enhanced keystroke dynamics to predict typist cognition and demographics,” *Journal of Human Computer Studies*, vol. 82, pp. 57–68, 2015.

[151] N. Ahmad, A. Szymkowiak, and P. a Campbell, “Keystroke dynamics in the pre-touchscreen era.,” *Frontiers in Human Neuroscience*, vol. 7, no. 12, p. 835, 2013.

[152] S. Hwang, S. Cho, and S. Park, “Keystroke dynamics-based authentication for mobile devices,” *Computers & Security*, vol. 28, no. 1–2, pp. 85–93, 2009.

[153] N. J. Grabham and N. M. White, “Use of a novel keypad biometric for enhanced user identity verification,” in *IEEE Instrumentation and Measurement Technology Conference*, 2008, pp. 12–16.

[154] M. Trojahn, F. Arndt, and F. Ortmeier, “Authentication with keystroke dynamics on touchscreen keypads-effect of different n-graph combinations,” *The Third International Conference on Mobile Services, Resources, and Users Authentication (MOBILITY13)*, pp. 114–119, 2013.

[155] M. Antal, L. Szabó, and I. László, “Keystroke dynamics on Android platform,” in *the 8th International Conference Interdisciplinarity in Engineering (INTER-ENG)*, 2014, pp. 9–15.

[156] C. L. Liu, C. J. Tsai, T. Y. Chang, W. J. Tsai, and P. K. Zhong, “Implementing multiple biometric features for a recall-based graphical keystroke dynamics authentication system on a smart phone,” *Journal of Network and Computer Applications*, vol. 53, pp. 128–139, 2015.

[157] V. D. Stanciu, R. Spolaor, and C. Giuffrida, “On the effectiveness of sensor-enhanced keystroke dynamics against statistical attacks,” in *the Sixth ACM Conference on Data and Application Security and Privacy*, 2016, pp. 105–112.

[158] K. A. Rahman and K. S. B. V. Phoha, “Snoop-forge-replay attacks on continuous verification With keystrokes,” *IEEE Transactions on Information Forensics And Security*, vol. 8, no. 3, pp. 528–541, 2013.

[159] S. Sagiroglu and G. Canbek, "Keyloggers," *IEEE Technology and Society Magazine*, vol. 28, no. 3, pp. 10 – 17, 2009.

[160] "Unix time," *Wikipedia*, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Unix_time. [Accessed: 12-Apr-2016].

[161] C. Hsu, C. Chang, and C. Lin, "A practical guide to support vector classification," *Department of Computer Science, National Taiwan University, Tech. Rep.*, 2010.

[162] M. Kantardzic, *Data mining: concepts, models, methods, and algorithms*, 2nd ed. John Wiley & Sons, 2011.

[163] M. Greenacre and R. Primicerio, "Measures of distance between samples: Euclidean," in *Multivariate Analysis of Ecological Data*, Fundacion BBVA, 2013, pp. 47–59.

[164] "The issues with biometric systems," *Biometric newsporter*, 2008. [Online]. Available: http://www.biometricnewsportal.com/biometrics_issues.asp. [Accessed: 25-Apr-2016].

[165] K. A. Provins and D. J. Glencross, "Handwriting, typewriting and handedness.," *The Quarterly Journal of Experimental Psychology*, vol. 20, no. 3, pp. 282–289, 1968.

[166] M. W. L. Jin, X. Ke, R. Manuel, "Keystroke dynamics: a software-based biometric solution," in *13th USENIX Security Symposium*, 2004.

[167] R. M. K. Killourhy, "The effect of clock resolution on keystroke dynamics," in *International Workshop on Recent Advances in Intrusion Detection*, 2008, pp. 331–350.

[168] "How do I time things in Java?," *Javamex*, 2014. [Online]. Available: http://www.javamex.com/faq/timing.shtml. [Accessed: 25-Nov-2015].

[169] K. Sung and S. Cho, "GA SVM wrapper ensemble for keystroke dynamics authentication," in *the International Conference on Biometrics*, 2006, pp. 654–660.

[170] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, Oct. 2007.

[171] R. Kohavi and H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, pp. 273–324, 1997.

[172] J. Holland, *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, USA, 1975.

[173] J. Kittler, "Feature set search algorithms," in *Pattern Recognition and Signal Processing*, 1978, pp. 41–60.

[174] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[175] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm feature subset selection," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 2, pp. 44–49, 1998.

[176] H. J. Seltman, *Experimental design and analysis*. Carnegie Mellon University, 2012.

[177] P. Jafari and F. Azuaje, “An assessment of recently published gene expression data analyses: reporting experimental design and statistical factors,” *BMC Medical Informatics and Decision Making*, vol. 6, no. 2, 2006.

[178] M. Hall, “Correlation-based feature selection for machine learning,” *PhD Thesis, Department of Computer Science, Waikato University, New Zealand*, 1999.

[179] D. Koller and M. Sahami, “Toward optimal feature selection,” in *the Thirteenth International Conference on Machine Learning*, 1996, pp. 284–292.

[180] E. Yu and S. Cho, “GA-SVM wrapper approach for feature subset selection in keystroke dynamics identity verification,” in *International Joint Conference on Neural Networks (INNS-IEEE)*, 2003, vol. 3, pp. 2253–2257.

[181] E. Yu and S. Cho, “Keystroke dynamics identity verification—Its problems and practical solutions,” *Computers & Security*, vol. 23, no. 5, pp. 428–440, 2004.

[182] D. Shanmugapriya and G. Padmavathi, “An efficient feature selection technique for user authentication using keystroke dynamics,” *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 11, no. 10, pp. 191–195, 2011.

[183] M. Dorigo, A. Colorni, and V. Maniezzo, “The ant system: an autocatalytic optimizing process,” *Université Libre de Bruxelles, Milano, Italy, Tech. Rep. 91-016*, 1991.

[184] M. Dorigo, “Optimization, learning and natural algorithms,” *PhD thesis, Dipartimento di Elettronica, Politecnico di Milano*, 1992.

[185] P. P. Grassé, “The theory of stigmergy : a possible interpretation of the behaviour of termites manufacturers,” *Insectes Sociaux*, vol. 6, pp. 41–80, 1959.

[186] C. Blum, “Ant colony optimization: Introduction and recent trends,” *Physics of Life Reviews*, vol. 2, no. 4, pp. 353–373, 2005.

[187] E. Bonabeau, M. Dorigo, and G. Theraulaz, “Swarm intelligence: from natural to artificial systems,” *Oxford University Press*, 1999.

[188] M. Dorigo and T. Stutzle, *Ant colony optimization*. The MIT Press, 2004.

[189] E. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, *The travelling salesman problem*. John Wiley & Sons, 1985.

[190] M. Dorigo and C. Blum, “Ant colony optimization theory: a survey,” *Theoretical Computer Science*, pp. 243 –278, 2005.

[191] M. Dorigo, V. Maniezzo, and A. Colorni, “The ant system: optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, vol. 26, pp. 1–13, 1996.

[192] L. M. Gambardella and M. Dorigo, “Ant-Q: a reinforcement learning approach to the traveling salesman problem,” in *the 12th International Conference on Machine Learning (ML-95)*, 1995, pp. 252–260.

[193] M. Dorigo and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 53–66, 1997.

[194] T. Stützle and H. Hoos, “Improving the ant system: a detailed report on the MAXMIN ant system,” *FG Intellektik, FB Informatik, TU Darmstadt, Germany, Tech. Rep. AIDA-96-12*, 1996.

[195] C. Blum, A. Roli, and M. Dorigo, “HC-ACO: the hyper-cube framework for ant colony optimization,” in *4th Metaheuristics International Conference (MIC’2001)*, 2001, pp. 399–403.

[196] S. Nemati, M. E. Basiri, N. Ghasem-Aghaee, and M. H. Aghdam, “A novel ACO–GA hybrid algorithm for feature selection in protein function prediction,” *Expert Systems with Applications*, vol. 36, pp. 12086–1209, 2009.

[197] M. H. Aghdam, N. Ghasem-Aghaee, and M. E. Basiri, “Text feature selection using ant colony optimization,” *Expert Systems with Applications*, vol. 36, no. 3, pp. 6843–6853, Apr. 2009.

[198] E. R. Girden, *ANOVA repeated measures*. Sage, 1992.

[199] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.

[200] P. R. Hinton, *Statistics explained*, 2nd ed. Routledge Ltd, 2004.

[201] M. Roberts and R. Russo, *A student’s guide to analysis of variance*. Routledge Ltd, 1999.

[202] H. R. Barker and B. M. Barker, *Multivariate analysis of variance (MANOVA) – A practical guide to its use in scientific decision making*. University of Alabama press, 1984.

[203] A. French, M. Macedo, J. Poulsen, T. Waterson, and A. Yu, “Multivariate analysis of variance (MANOVA),” *San Francisco State University*, pp. 1–8, 2002.

[204] A. C. Rencher and W. F. Christensen, *Methods of multivariate analysis*, 3rd ed. Wiley, 2012.

[205] D. Mahar, R. Napier, M. Wagner, R. D. Henderson, and M. Hiron, “Optimizing digraph-latency based biometric typist verification systems: inter and intra typist differences in digraph latency distributions,” *International Journal of Human-Computer*, vol. 43, no. 4, pp. 579–592, 1995.

[206] R. Napier, W. Laverty, D. Mahar, R. Henderson, M. Hiron, and M. Wagner, “Keyboard user verification: toward an accurate, efficient and ecological valid algorithm,” *International Journal of Human-Computer Studies*, vol. 43, pp. 213–222, 1995.

[207] M. Analoui, A. Mirzaei, and H. Davarpanah, “Using multivariate analysis of variance algorithm in keystroke identification,” in *International Symposium on Telecommunications (IST2003),* 2003, pp. 391–395.

[208] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273 – 297, 1995.

[209] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[210] T. Fletcher, “Support vector machines explained,” *UCL*, 2009. [Online]. Available: http://sutikno.blog.undip.ac.id/files/2011/11/SVM-Explained.pdf. [Accessed: 05-Dec-2016].

[211] V. Roth and V. Steinhage, “Nonlinear discriminant analysis using kernel functions,” in *Advances in Neural Information Processing Systems*, 1999, vol. 12, pp. 568–574.

[212] A. Vlachos, “Active learning with support vector machines,” *Masters Dissertation, University of Edinburgh*, 2004.

[213] C. Hsu and C. Lin, “A comparison on methods for multi-class support vector machines”,” *Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan,Tech. Rep.*, 2001.

[214] V. N. Vapnik, *Statistical learning theory*. Wiley, 1998.

[215] U. Krebel, “Pairwise classification and support vector machines,” in *B. Scholkopf, C. J. C. Burges and A. J. Smola, “Advances in kernel methods - support vector learning,”* MIT Press, 1999, pp. 255–268.

[216] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, “Large margin DAGs for multiclass classification,” *Advances in Neural Information Processing Systems*, vol. 12, pp. 547–552.

[217] K. Heller, K. Svore, A. D. Keromytis, and S. Stolfo, “One class support vector machines for detecting anomalous windows registry accesses,” in *Workshop on Data Mining for Computer Security (DMSEC)*, 2003.

[218] G. Saggio, G. Costantini, and M. Todisco, “Cumulative and ratio time evaluations in keystroke dynamics to improve the password security mechanism,” *Journal of Computer and Information Technology*, vol. 1, pp. 2–11, 2011.

[219] K. M. Salama and A. A. Freitas, “Learning bayesian network classifiers using ant colony optimization,” *Swarm Intelligence*, vol. 7, no. 2–3, pp. 229–254, 2013.

[220] G. Castellanos, E. Delgado, G. Daza, L. G. Sanchez, and J. F. Suarez, “Feature selection in pathology detection using hybrid multidimensional analysis,” in *the 28th IEEE EMBS Annual International Conference*, 2006, pp. 5503–5506.

[221] B. Surendiran and A. Vadivel, “Feature selection using stepwise anova discriminant analysis for mammogram mass classification,” *ACEEE International Journal on Signal & Image Processing*, vol. 02, no. 01, pp. 1–3, 2011.

[222] A. K. Jain, R. P. W. Duin, and J. Mao, “Statistical pattern recognition: a review,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4 – 37, 2000.

[223] P. Bryan, “Is your brain on autopilot when you type?,” *RedOrbit*, 2013. [Online]. Available: http://www.redorbit.com/news/science/1113021896/your-brain-is-on-autopilot-when-you-type-120613/. [Accessed: 12-Mar-2016].

[224] I. Kirshenbaum and M. Sela, “A computer user virtual mark - keystroke dynamics using binary and multi-class SVM,” 2011. [Online]. Available: https://code.google.com/archive/p/keystroke-dynamics-svm/. [Accessed: 06-May-2016].

[225] G. Lamp, “Why use SVM?,” 2012. [Online]. Available: http://www.yaksis.com/posts/why-use-svm.html. [Accessed: 30-Nov-2015].

[226] P. Lameski, E. Zdravevski, R. Mingov, and A. Kulakov, “SVM parameter tuning with grid search and its impact on reduction of model over-fitting,” in *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, 2015, pp. 464–474.

[227] C. C. Chang and C. J. Lin, “LIBSVM: a library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.

[228] “RBF SVM parameters,” 2014. [Online]. Available: http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html. [Accessed: 14-Mar-2016].

[229] J. Milgram, M. Cheriet, and R. Sabourin, “‘One against one’ or ‘one against all’: which one is better for handwriting recognition with SVMs?,” in *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006, pp. 1–6.

[230] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, “An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes,” *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, 2011.

[231] K. B. Duan and S. S. Keerthi, “Which is the best multiclass SVM method? An empirical study,” in *International Workshop on Multiple Classifier Systems*, 2005, pp. 278–285.

[232] J. Brank and M. Grobelnik, “Training text classifiers with SVM on very few positive examples,” *Microsoft Research*, 2003.

[233] H. He and A. Ghodsi, “Rare class classification by support vector machine,” in *Proceedings - International Conference on Pattern Recognition*, 2010, pp. 548–551.

[234] J. Ilonen, “Keystroke dynamics,” *Lappeenranta University of Technology*, 2006.

[235] B. Deshpande, *Decision tree digest - Understand, build and use decision trees for common business problems with RapidMiner*. SimaFore, 2014.

[236] W. Y. Loh, “Classification and regression trees,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[237] D. Hand, H. Mannila, and P. Smyth, *Principles of data mining*. The MIT Press, 2001.

[238] L. Rokach and O. Maimon, “Decision tree,” in *Data Mining and Knowledge Discovery Handbook*, 2005, p. pp 165–192.

[239] S. R. Safavian and D. Landgrebe, “A survey of decsion tree clasifier methodology,” *IEEE Transactions on Systems, Man, And Cybernetics*, vol. 21, no. 3, pp. 660–674, 1990.

[240] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and regression trees*. Wadsworth Int. Group, 1984.

[241] I. Bratko and M. Bohanec, "Trading accuracy for simplicity in decision trees," *Machine Learning*, vol. 15, no. 3, pp. 223–250, 1994.

[242] H. Almuallim, "An efficient algorithm for optimal pruning of decision trees," *Artificial Intelligence*, vol. 83, no. 2, pp. 347–362, 1996.

[243] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation," *Encyclopedia of Database Systems*, pp. 532–538, 2009.

[244] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning data mining, inference, and prediction*, 2nd ed. Springer, 2008.

[245] M. Kearns and D. Ron, "Algorithmic stability and sanity-check bounds for leave-one-out cross-validation," *Neural Computing*, vol. 11, no. 6, pp. 1427–1453, 1999.

[246] M. Friedl and C. Brodley, "Decision tree classification of land cover from remotely sensed data," *Remote Sensing of Environment*, vol. 61, no. 3, pp. 399–409, 1997.

[247] E. Chen, "Choosing a machine learning classifier," 2011. [Online]. Available: http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/. [Accessed: 14-Jan-2016].

[248] "Decision trees." [Online]. Available: http://uk.mathworks.com/help/stats/classification-trees-and-regression-trees.html. [Accessed: 03-Feb-2016].

[249] W. W. Hsieh, *Machine learning methods in the environmental sciences*. Cambridge University Press, 2009.

[250] M. Golfarelli, D. Maio, and D. Maltoni, "On the error-reject trade-off in biometrie verification systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 786–796, 1997.

[251] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*, 3rd ed. Prentice Hall, Englewood Cliffs, 2010.

[252] F. Löw, U. Michel, S. Dech, and C. Conrad, "Impact of feature selection on the accuracy and spatial uncertainty of per-field crop classification using Support Vector Machines," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 85, pp. 102–119, 2013.

[253] K. Warwick and R. Craddock, "An introduction to radial basis functions for system identification a comparison with other neural network methods .," in *35th IEEE International Conference on Decision and Control*, 1996, pp. 464–469.

[254] S. G. Nancy and S. A. alias Balamurugan, "A comparative study of feature selection methods for cancer classification using gene expression dataset," *Journal of Computer Applications (JCA)*, vol. 6, no. 3, pp. 79–84, 2013.

[255] A. K. Jain, R. Bolle, and S. Pankanti, *Biometrics: personal identification in networked society*. Springer, 2006.

[256] N. Poh and S. Bengio, "Evidences of equal error rate reduction in biometric authentication fusion," *IDIAP*, 2004.

[257] H. Lv and W. Wang, “Biologic verification based on pressure sensor keyboards and classifier fusion techniques,” *IEEE Transactions on Consumer Electronics*, vol. 52, no. 3, pp. 1057–1063, 2006.

[258] D. Ruta and B. Gabrys, “An overview of classifier fusion methods,” *Computing and Information Systems*, vol. 7, pp. 1–10, 2000.

[259] Y. Zhang and Q. Ji, “Efficient sensor selection for active information fusion,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 40, no. 3, pp. 719–728, 2010.

[260] J. Bezdek, J. Keller, R. Krisnapuram, and N. Pal, *Fuzzy models and algorithms for pattern recognition and image processing*. Springer, 1999.

[261] J. Esteban, A. Starr, R. Willetts, P. Hannah, and P. Bryanston-cross, “A review of data fusion models and architectures: towards engineering guidelines,” *Neural Computing & Applications*, vol. 14, no. 4, pp. 273–281, 2005.

[262] S. C. Thomopoulos, “Sensor integration and data fusion,” in *Sensor Fusion II: Human and Machine Strategies*, 1989, pp. 178–191.

[263] L. F. Pau, “Sensor data fusion,” *Journal of Intelligent and Robotic Systems*, vol. 1, pp. 103–116, 1988.

[264] C. J. Harris, A. Bailey, and T. J. Dodd, “Multi-sensor data fusion in defence and aerospace,” *Aeronautical Journal*, vol. 102, no. 1015, pp. 229–244, 1998.

[265] A. Ross and R. Govindarajan, “Feature level fusion using hand and face biometrics,” in *SPIE Conference on Biometric Technology for Human Identification*, 2005, pp. 196–204.

[266] V. Sharma and J. W. Davis, “Feature-level fusion for object segmentation using mutual information,” *Augmented Vision Perception in Infrared*, pp. 295–319, 2008.

[267] S. Das and W. Krebs, “Sensor fusion of multi-spectral imagery,” *Electronics Letters*, vol. 36, pp. 1115–1116, 2000.

[268] J. Davis and V. Sharma, “Background-subtraction using contour-based fusion of thermal and visible imagery,” *Computer Vision and Image Understanding*, vol. 106, pp. 162–182, 2007.

[269] T. H. Ho, J. J. Hull, and S. N. Srihari, “Decision combination in multiple classifier system,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 66–75, 1994.

[270] P. D. Gader, M. A. Mohamed, and J. M. Keller, “Fusion of handwritten word classifiers,” *Pattern Recognition Letters*, vol. 17, pp. 577–584, 1996.

[271] Y. S. Huang and C. Y. Suen, “A method of combining multiple experts for the recognition of unconstrained handwritten numerals,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 90–94, 1995.

[272] G.J.Klir and T. A. Folger, *Fuzzy sets, uncertainty, and information*, Internatio. Prentice Hall, 1988.

[273] U. Bhattacharya and B. B. Chaudhuri, “A majority voting scheme for multiresolution recognition of handprinted numerals,” in *the Seventh International Conference on Document Analysis and Recognition (ICDAR03)*, 2003, pp. 16 – 20.

[274] M. Bachmann and B. Smith, “Internet fraud,” in *Encyclopedia of Cyber Behavior*, Z. Yan, Ed. 2012, pp. 931–943.

[275] C. Ratanamahatana and D. Gunopulos, “Scaling up the naive bayesian classifier : using decision trees for feature selection,” in *Workshop Data Cleaning and Preprocessing (DCAP ’02), at IEEE International Conference on Data Mining (ICDM '02)*, 2002.

[276] B. D. Ville and P. Neville, *Decision trees for analytics using SAS enterprise miner*. SAS Institute, 2013.

[277] K. Katzner, *The languages of the world*. Routledge Ltd, 1975.

[278] R. L. Cooper, E. Olshtain, G. R. Tucker, and M. Waterbury, “The acquisition of complex English structures by adult native speakers of Arabic and Hebrew,” *Language Learning*, vol. 29, no. 2, pp. 255–275, 1979.

[279] K. Versteegh, *The Arabic language*, 2nd ed. Edinburgh University Press, 2014.

[280] T. M. Malas, S. S. Taifour, and G. a. Abandah, “Toward optimal Arabic keyboard layout using genetic algorithm,” in *9th Intnationl Middle Eastern Multiconference on Simulation and Modeling (MESM08)*, 2008, pp. 50–54.

[281] R. Jensen, “Performing feature selection with ACO,” *Swarm Intelligence in Data Mining*, vol. 34, pp. 45–73, 2006.

[282] T. Ambwani, “Multi class support vector machine implementation to intrusion detection,” in *the International Joint Conference on Neural Networks*, 2003, vol. 3, pp. 2300 – 2305.

[283] I. E. Fang, “It isn’t ETAOIN SHRDLU; it's ETAONI RSHDLC,” *Journalism Quarterly*, vol. 43, pp. 761–762, 1966.

[284] H. Thorson, “Using the computer to compare foreign and native language writing processes: a statistical and case study approach,” *The Modern Language Journal*, vol. 84, no. 2, pp. 155–169, 2000.

[285] A. A. Ahmed and I. Traore, “Biometric recognition based on free-text keystroke dynamics,” *IEEE Transactions on Cybernetics*, vol. 44, no. 4, pp. 458 – 472, 2014.

[286] P. S. Teh, N. Zhang, A. B. J. Teoh, and K. Chen, “A survey on touch dynamics authentication in mobile devices,” *Computers and Security*, vol. 59, pp. 210–235, 2016.

[287] N. Japkowicz, “Learning from imbalanced data sets: a comparison of various strategies,” in *Learning from Imbalanced Data Sets: Papers from the AAAI Workshop*, N. Japkowicz, Ed. AAAI Press, 2000, pp. 10–15.

[288] P. K. Chan and S. J. Stolfo, “Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection,” in *4th International Conference on Knowledge Discovery and Data Mining (KDD-98)*, 1998, pp. 164–168.

[289] F. P. Miller, A. F. Vandome, and J. McBrewster, *Keyboard layout: keyboard (computing), typewriter, alphanumeric keyboard, QWERTY, Portuguese alphabet, QWERTZ, AZERTY, Dvorak simplified keyboard, chorded keyboard, Arabic keyboard, Hebrew keyboard*. Alpha Press, 2009.

Appendix A

Ethical Approval

A. 1 Consent Form

School of Systems Engineering

Consent Form

Keystroke Dynamics Authentication

Investigator: Prof. Kevin Warwick and Dr. Hong Wei

Student Investigator: Arwa Alsultan

Participant No.: **Please Initial Box**

1. I confirm that I have read and understood the information contained in the "Information Sheet" needed to participate in this research exercise. ☐
2. I agree that the data collection will involve recording typing behaviour that includes typing time and editing activities. ☐
3. I consent to the use of the recorded data for research purposes; the data will be stored securely in line with the university data protection policies; portions of the recorded data may be made available to members of the research community outside the University in order to undertake research into keystroke dynamics recognition; data collected is anonymous – my identity is not associated with the data. ☐
4. I agree to take part in the above research and data collection exercise and I consent that I can withdraw from the study at any time if I chose to do so. ☐

_____ _____

Name of Participant Signature/Date

_____ _____

Name of Researcher Signature/Date

A. 2 Information Sheet

School of Systems Engineering

University of Reading

Information Sheet

Research Title: Keystroke Dynamics Authentication

Investigator: Prof. Kevin Warwick and Dr. Hong Wei

Student Investigator: Arwa Alsultan

Project Purposes

This experiment is part of a research conducted on free-text keystroke dynamics to study the human typing patterns. Particularly, the application of keystroke dynamics in user authentication is the main focus of this research.

Keystroke dynamics uses the manner and rhythm in which an individual types characters on a keyboard to develop a unique biometric template of the user's typing pattern to be used for future authentication. Raw data collected from the keyboard include the time each key is pressed and released. This raw data can be, then, used to determine many features that can be used in keystroke dynamics to distinguish between individuals using their typing patterns.

Participating

The participants are randomly selected 18-60 years old males and females. All participants are treated anonymously in this research.

It is up to the participant to decide whether or not to take part of this experiment. If the participant decides to take part he/she will be given this information sheet to keep and be asked to sign a consent form. If the participant decides to take part he/she is still free to withdraw at any time and without giving a reason.

Participants are, also, entitled to request the research results if they so wish.

Research Summary

In this experiment participants are kindly requested to install an application on their PC and use it to collect experimental typing data; installation details are provided in the "Participants Instructions" document. If the participants feel uncomfortable installing this software they are permitted not to continue. This application includes various typing tasks that the participant is asked to perform in the convenience of their own machine. These tasks are either copy-typing tasks in which participants will copy a number of paragraphs from an article or free-typing tasks in which the participants are free to type whatever they want.

The typing information will be collected and stored in the participant's system. This data can be, then, sent to the researcher in order to be used in further research. This data has to be sent back to the researcher by the

participant; no information is broadcasted to the internet as the application doesn't have any access to the internet.

The data that will be collected from the participants' machines include:

- Hold time: the time each key is pressed until it is released.
- Latency Time: There are three types of latencies:
  - Down-Down (DD) time: the interval time between two successive key presses.
  - Up-UP (UU) time: the interval time between two successive key releases.
  - Up-Down (UD) time: the interval time between a key release and the next key press.
- Other typing features: this includes typing speed, frequency of error, shift key usage and editing patterns.

Although the data collection process will be conducted on the participants own machines, it only captures keystrokes typed on the actual application. Any other keystrokes typed on any other application on the participants PC will not be captured. Therefore, all sensitive and private information is completely safe as this application does not work in the backside of the machine but have to be executed by the participants to capture keystrokes typed only on this application. Moreover, the collected data does not contain any information related to the participant or his/her machine.

After the experiment is complete the software can be uninstalled; details of the uninstallation process are provided in the "Participants Instructions" document.

Research Ethics

This project has been subject to ethical review, according to the procedures specified by the University Research Ethics Committee, and has been given a favourable ethical opinion for conduct.

All the data recorded in this experiment is for research purposes only. The data will be stored securely in line with the university data protection policies. Portions of the recorded data may be published to members of the research community for future research.

All data collected is anonymous and any information about the participants collected during the project is confidential.

A. 3 Participating Instructions

School of Systems Engineering

University of Reading

Participating Instructions - Keystroke Data Collection

Dear Participant

The following three parts are guides for you to install the application, use it to collect data and then uninstall it:

To install visual Studio 2013 & the data collection application, please follow the instructions below:

1. Go to https://login.live.com/ webpage.
2. Enter your Microsoft account username/password to log-in. If you don't have a Microsoft account, use this: username: user1_2014@outlook.com, password: U.ser2014
3. Complete your information, and then click "Express 2013 for Windows Desktop."
4. Follow the installing wizard: opt-in the "agree on terms" option, then click "Install", then click "Yes".
5. Wait for few minutes until the installation is complete.
6. Depending on your machine configuration, click "Restart" and wait for the PC to reboot or click "Launch" and close the visual studio interface.
7. After that, unzip the folder "project1_folder", attached to this e-mail, to be used later.

To complete the data collection process, please follow the instructions below:

1. Run "project1.exe" by clicking on the "project1" file located inside the unzipped "project1_folder".
2. Click on the first task. Then text will appear in the empty space in the upper box of the screen.
3. Perform the task by typing the text in the lower empty space of the screen (the white space).
4. Click Done. After the first task, you will find that a text file called "data.txt" has been created and saved in the "project1_folder", it will contain all the collected data from all typing tasks.
5. Perform steps 1, 2, 3 and 4 again for the remaining tasks, each task is performed separately.
6. After completing all 8 tasks, please e-mail me back the text file "data.txt" which is located in the "project1_folder".

To uninstall visual Studio 2013 & the data collection application, please follow the instructions below:

1. Delete the folder "project1_folder" from your machine. This will delete the "project1" application and the data collected in the file "data.txt".
2. Uninstall visual Studio 2013 express from control panel on you PC.

Note:

- When installing Visual Studio or running the project1 application, you might encounter warnings that the file's content may be harmful. This is because they are executable files, they are completely safe. Please ignore these warnings and carry on the insulation or the execution.
- Please type in the usual way that you normally do. You can use any key on the keyboard including the Backspace, Enter, Shift, Numpad keys … etc.

- Each typing task must be performed separately; you have the freedom to choose the time between tasks as long as it is more than a day.
- If you forgot what task have you completed last, you can open the "data.txt" file and check the last line in the file, it will be something like this: "The last task you have completed was Task #:1".

Please don’t hesitate to e-mail the researcher if you have any questions at: a.f.a.alsultan@pgr.reading.ac.uk

Thank you for you for your assistance with the data collection task.

Appendix B

Selected Source Code

B. 1 Keystroke Collector

B.1.1 Header File

```
#include <Windows.h>
#include <iostream>
#include <sstream>
#include <fstream>
#include <list>
#include <stdlib.h>
#include <vector>


using namespace std;

#pragma comment(lib, "user32.lib")

#define DBOUT( s )            \
{                             \
    std::wostringstream os_;    \
    os_ << s;                   \
    OutputDebugStringW(os_.str().c_str());  \
}

 % node class
public class Mynode {

public:
    Mynode(char _x, LARGE_INTEGER _y, std::string _z, std::string _q)
    {
            charec = _x;
            time = _y;
            type = _z;
            status = _q;
    }
    Mynode()
    {}

    char get_charc()
    {
            return charec;
    }
    LARGE_INTEGER get_time()
    {
            return time;
    }
    std::string get_type()
    {
            return type;
    }
    std::string get_status()
    {
```

```
49.             return status;
50.     }
51.     void change_charc(char c)
52.     {
53.             charec = c;
54.     }
55.
56. private:
57.     char charec;
58.     LARGE_INTEGER time;
59.     std::string type;
60.     std::string status;
61.
62. };
63. Mynode mylist[3000];
64. Mynode mynewlist[3000];
65.
66. % pair class
67. public class Mypair {
68.
69. public:
70.     Mypair(char _x, char _y, double _z, double _q, double _w, double _e, double _r, std::string
    _s)
71.     {
72.             char1=_x;
73.             char2=_y;
74.             hold1=_z;
75.             hold2=_q;
76.             dd=_w;
77.             uu=_e;
78.             ud=_r;
79.             type = _s;
80.     }
81.
82.     Mypair()
83.     {}
84.
85.     char get_char1()
86.     {
87.             return char1;
88.     }
89.     char get_char2()
90.     {
91.             return char2;
92.     }
93.     double get_h1()
94.     {
95.             return hold1;
96.     }
97.     double get_h2()
98.     {
99.             return hold2;
100.    }
101.    double get_dd()
102.    {
103.            return dd;
104.    }
105.    double get_uu()
106.    {
107.            return uu;
108.    }
109.    double get_ud()
110.    {
111.            return ud;
112.    }
113.    std::string get_type()
114.    {
115.            return type;
116.    }
117.    void add_hold2(double h)
118.    {
119.            hold2 = h;
120.    }
121.    void add_hold1(double h)
```

```
122.    {
123.            hold1 = h;
124.    }
125.    void add_type(std::string s)
126.    {
127.            type = s;
128.    }
129.
130.private:
131.    char char1;
132.    char char2;
133.    double hold1;
134.    double hold2;
135.    double dd;
136.    double uu;
137.    double ud;
138.    std::string type;
139.};
140.Mypair mypairlist[3000];
141.
142.% vars
143.int caplock = 0, numlock = 0, letter_caplock = 0;
144.LARGE_INTEGER mystart = { 0 }, myend = { 0 };
145.std::string gl_st="";
146.int i = 0,j=0,k=0,q=0, task=0;
147.
148. % print list
149.void print_keyslist(Mynode list[], int type)
150.{
151.    LARGE_INTEGER frequency;
152.    int m;
153.    QueryPerformanceFrequency(&frequency);
154.            switch (type)
155.            {
156.            case 0: gl_st=gl_st+"Raw list :\n";
157.                     m = i;
158.                    break;
159.            case 1: gl_st=gl_st+ "Ordered list :\n";
160.                     m = j;
161.                    break;
162.            }
163.
164.            for (int it = 0; it <m; it++)
165.            {
166.                    std::string s1 = std::to_string(list[it].get_time().QuadPart);
167.                    gl_st=gl_st+ list[it].get_charc() + " | " + s1 + " | " + list[it].get_type() +
    " | " + list[it].get_status() + "\n";
168.            }
169.            gl_st=gl_st+ ("_____________________________________\n");
170.
171.}
172.
173.% print list
174.void print_pair()
175.{
176.            gl_st=gl_st+ ("char1 | char2 |   hold1 | hold2 |   uu |    dd  |   du  |   status\n");
177.            for (int it = 0; it != k; it++)
178.            {
179.                    std::string s1 = std::to_string(mypairlist[it].get_h1());
180.                    std::string s2 = std::to_string(mypairlist[it].get_h2());
181.                    std::string s3 = std::to_string(mypairlist[it].get_uu());
182.                    std::string s4 = std::to_string(mypairlist[it].get_dd());
183.                    std::string s5 = std::to_string(mypairlist[it].get_ud());
184.                    gl_st=gl_st+mypairlist[it].get_char1() + " | " + mypairlist[it].get_char2() +
    " | " + s1 + " | " + s2 + " | " + s3 +  " | " + s4 + " | " + s5 + " | " +
    mypairlist[it].get_type() + "\n";
185.            }
186.}
187.
188.void add_h2( double h)
189.{
190.    if (k != 0)
191.    {
192.            mypairlist[k - 1].add_hold2(h);
```

```
    }
}

void add_h2_final( double h)
{
    if (k != 0)
    {
            mypairlist[k - 1].add_hold2(h);
    }
}

double d[88];
% find value of keys
 void initingarr()
{
    d['`' - 35] = 100.0;
    d['1' - 35] = 101.0;
    d['2' - 35] = 102.0;
    d['3' - 35] = 103.0;
    d['4' - 35] = 104.0;
    d['5' - 35] = 105.0;
    d['6' - 35] = 106.0;
    d['7' - 35] = 107.0;
    d['8' - 35] = 108.0;
    d['9' - 35] = 109.0;
    d['0' - 35] = 110.0;
    d['-' - 35] = 111.0;
    d['=' - 35] = 112.0;

    //my keyboard second row is about 20% indented from first row
    d['Q' - 35] = 200.5;
    d['W' - 35] = 201.5;
    d['E' - 35] = 202.5;
    d['R' - 35] = 203.5;
    d['T' - 35] = 204.5;
    d['Y' - 35] = 205.5;
    d['U' - 35] = 206.5;
    d['I' - 35] = 207.5;
    d['O' - 35] = 208.5;
    d['P' - 35] = 209.5;
    d['[' - 35] = 210.5;
    d[']' - 35] = 211.5;

    //my keyboard third row is about 20% indented from second row
    d['A' - 35] = 300.2;
    d['S' - 35] = 301.2;
    d['D' - 35] = 302.2;
    d['F' - 35] = 303.2;
    d['G' - 35] = 304.2;
    d['H' - 35] = 305.2;
    d['J' - 35] = 306.2;
    d['K' - 35] = 307.2;
    d['L' - 35] = 308.2;
    d[';' - 35] = 309.2;
    d['\'' - 35] = 310.2;
    d['#' - 35] = 311.2;

    //fourth row is about 50% indented from third row
    d['\\' - 35] = 399.5;
    d['Z' - 35] = 400.5;
    d['X' - 35] = 401.5;
    d['C' - 35] = 402.5;
    d['V' - 35] = 403.5;
    d['B' - 35] = 404.5;
    d['N' - 35] = 405.5;
    d['M' - 35] = 406.5;
    d[',' - 35] = 407.5;
    d['.' - 35] = 408.5;
    d['/' - 35] = 409.5;
}

% order list if -ud
void order_list(Mynode list[], Mynode newlist[])
 {
```

```
267.     for (int it = 0 ; it != i; it++)
268.     {
269.             if (list[it].get_type() == "DOWN")
270.             {
271.                     if (!(it != 0 && (list[it - 1].get_type() == "DOWN") && (list[it].get_charc()
    == list[it - 1].get_charc())))
272.                     {
273.                             for (int jt = it; jt != i; jt++)
274.                             {
275.                                     if (list[it].get_charc() == list[jt].get_charc() &&
    list[jt].get_type() == "UP")
276.                                     {
277.                                             Mynode anode(list[it].get_charc(),
    list[it].get_time(), list[it].get_type(), list[it].get_status());
278.                                             newlist[j] = anode;
279.                                             j++;
280.                                             Mynode anode2(list[jt].get_charc(),
    list[jt].get_time(), list[jt].get_type(), list[jt].get_status());
281.                                             newlist[j] = anode2;
282.                                             j++;
283.                                             list[jt].change_charc('*');
284.                                             break;
285.                                     }
286.                             }
287.                     }
288.             }
289.     }
290. }
291.% test key-pair relation
292.bool prox(int jt)
293. {
294.
295.     try
296.     {
297.             double a1 = d[mypairlist[jt].get_char2() - 35];
298.             double a2 = d[mypairlist[jt].get_char1() - 35];
299.
300.             double dummy = abs(a1 - a2);
301.
302.             return dummy == 0 || dummy == 1 || (dummy >= 98.5 && dummy <= 100.4);
303.     }
304.     catch (int e)
305.     {
306.             DBOUT("error1"<<e);
307.             return false;
308.     }
309. }
310.
311. % test key-pair relation
312.bool prox2( int jt)
313. {
314.     try
315.     {
316.             double a1 = d[mypairlist[jt].get_char2() - 35];
317.             double a2 = d[mypairlist[jt].get_char1() - 35];
318.
319.             double dummy = abs(a1 - a2);
320.
321.             return dummy == 2 || (dummy >= 97.5 && dummy <= 101.4) || (dummy >= 197.2 && dummy <=
    200);
322.     }
323.     catch (int e)
324.     {
325.             DBOUT("error2" << e);
326.             return false;
327.     }
328. }
329.
330. % test key-pair relation
331.bool prox3(int jt)
332. {
333.     try
334.     {
335.             double a1 = d[mypairlist[jt].get_char2() - 35];
```

```
336.             double a2 = d[mypairlist[jt].get_char1() - 35];
337.
338.             double dummy = abs(a1 - a2);
339.
340.             return dummy == 3 || (dummy >= 96.5 && dummy <= 102.4) || (dummy > 196 && dummy <=
    201) || (dummy >= 296.5 && dummy <= 300);
341.     }
342.     catch (int e)
343.     {
344.             DBOUT("error3" << e);
345.             return false;
346.     }
347. }
348.
349. bool prox4(int jt)
350. {
351.     try
352.     {
353.             double a1 = d[mypairlist[jt].get_char2() - 35];
354.             double a2 = d[mypairlist[jt].get_char1() - 35];
355.
356.             double dummy = abs(a1 - a2);
357.
358.             return dummy == 4 || (dummy >= 95.5 && dummy <= 103.4) || (dummy > 195 && dummy <=
    202) || (dummy >= 295 && dummy <= 300.5);
359.     }
360.     catch (int e)
361.     {
362.             DBOUT("error4" << e);
363.             return false;
364.     }
365. }
366.
367.% return symbol
368. char getchar(int x)
369. {
370.     switch (x)
371.     {
372.     case 221:return']';
373.     case 219:return'[';
374.     case 222: return '#';
375.     case 192: return '\'';
376.     case 186: return ';';
377.     case 191: return '/';
378.     case 190: return '.';
379.     case 188: return ',';
380.     case 187: return '=';
381.     case 189: return '-';
382.     case 220: return '\\';
383.     case 223: return '`';
384.     case 111: return '/';
385.     case 106: return '*';
386.     case 109: return '-';
387.     case 103: return '7';
388.     case 104: return '8';
389.     case 105: return '9';
390.     case 107: return '+';
391.     case 100: return '4';
392.     case 101: return '5';
393.     case 102: return '6';
394.     case 97: return '1';
395.     case 98: return '2';
396.     case 99: return '3';
397.     case 96: return '0';
398.     case 110: return '.';
399.     case 16: return '$';
400.     default:return(char)x;
401.     }
402. }
403.
404.% right side
405. bool isright(char x)
406. {
407.     switch (x)
```

```
408.     {
409.     case '=': return true;
410.     case '-': return true;
411.     case '0': return true;
412.     case '9': return true;
413.     case '8': return true;
414.     case '7': return true;
415.     case ']': return true;
416.     case '[': return true;
417.     case 'P': return true;
418.     case 'O': return true;
419.     case 'I': return true;
420.     case 'U': return true;
421.     case 'Y': return true;
422.     case '#': return true;
423.     case '\'': return true;
424.     case ';': return true;
425.     case 'L': return true;
426.     case 'K': return true;
427.     case 'J': return true;
428.     case 'H': return true;
429.     case '/': return true;
430.     case '.': return true;
431.     case ',': return true;
432.     case 'M': return true;
433.     case 'N': return true;
434.     }
435.     return false;
436. }
437.
438. % numlock
439.bool isnumlock(int s)
440. {
441.     switch (s)
442.     {
443.     case VK_NUMPAD0: return true;
444.     case VK_NUMPAD1: return true;
445.     case VK_NUMPAD2: return true;
446.     case VK_NUMPAD3: return true;
447.     case VK_NUMPAD4: return true;
448.     case VK_NUMPAD5: return true;
449.     case VK_NUMPAD6: return true;
450.     case VK_NUMPAD7: return true;
451.     case VK_NUMPAD8: return true;
452.     case VK_NUMPAD9: return true;
453.     case 110: return true;
454.     case 107: return true;
455.     case 109: return true;
456.     case 106: return true;
457.     case 111: return true;
458.     }
459.     return false;
460. }
461.
462.% special charachter
463. char returnsymbol(int s)
464. {
465.
466.     switch (s)
467.     {
468.     case 187: return '=';
469.     case 189: return '-';
470.     case 48: return '0';
471.     case 57: return '9';
472.     case 56: return '8';
473.     case 55: return '7';
474.     case 54: return '6';
475.     case 53: return '5';
476.     case 52: return '4';
477.     case 51: return '3';
478.     case 50: return '2';
479.     case 49: return '1';
480.     case 223: return '`';
481.     case 221: return ']';
```

```
482.     case 219: return '[';
483.     case 80: return 'p';
484.     case 79: return 'o';
485.     case 73: return 'i';
486.     case 85: return 'u';
487.     case 89: return 'y';
488.     case 84: return 't';
489.     case 82: return 'r';
490.     case 69: return 'e';
491.     case 81: return 'w';
492.     case 87: return 'q';
493.     case 222: return '#';
494.     case 192: return '\'';
495.     case 186: return ';';
496.     case 76: return 'l';
497.     case 75: return 'k';
498.     case 74: return 'j';
499.     case 72: return 'h';
500.     case 71: return 'g';
501.     case 70: return 'f';
502.     case 68: return 'd';
503.     case 83: return 's';
504.     case 65: return 'a';
505.     case 191: return '/';
506.     case 190: return '.';
507.     case 188: return ',';
508.     case 77: return 'm';
509.     case 78: return 'n';
510.     case 66: return 'b';
511.     case 86: return 'v';
512.     case 67: return 'c';
513.     case 88: return 'x';
514.     case 90: return 'z';
515.     case 220: return '\\';
516.     case 32: return ' ';
517.     case 8: return (char)8;
518.     case 13: return (char)13;
519.     case 20: return (char)VK_CAPITAL;
520.     case 16: return (char)VK_RSHIFT;
521.     }
522.    return '@';
523. }
524.
525.% prent key-pair relation
526. void print_keypair()
527. {
528.     std::string mytype = "";
529.
530.             gl_st=gl_st+ "_____________________________________\n";
531.             gl_st = gl_st + "hold1 | hold2 |   uu |    dd  |   du\n";
532.
533.             for (int j = 0; j < 17; j++)
534.             {
535.                     switch (j)
536.                     {
537.                     case 0: mytype = "adj_right";
538.                             break;
539.                     case 1: mytype = "second_right";
540.                             break;
541.                     case 2: mytype = "third_right";
542.                             break;
543.                     case 3: mytype = "fourth_right";
544.                             break;
545.                     case 4: mytype = "non_right";
546.                             break;
547.                     case 5: mytype = "same_right";
548.                             break;
549.                     case 6: mytype = "adj_left";
550.                             break;
551.                     case 7: mytype = "second_left";
552.                             break;
553.                     case 8: mytype = "third_left";
554.                             break;
555.                     case 9: mytype = "fourth_left";
```

```
556.                         break;
557.                     case 10: mytype = "non_left";
558.                         break;
559.                     case 11: mytype = "same_left";
560.                         break;
561.                     case 12: mytype = "adj_rl";
562.                         break;
563.                     case 13: mytype = "second_rl";
564.                         break;
565.                     case 14: mytype = "third_rl";
566.                         break;
567.                     case 15: mytype = "fourth_rl";
568.                         break;
569.                     case 16: mytype = "non_rl";
570.                         break;
571.                     }
572.                     gl_st = gl_st + mytype + "\n";
573.                     for (int it = 0; it != k; it++)
574.                     {
575.                                                             if (mypairlist[it].get_type() ==
    mytype)
576.                             {
577.                                     std::string s1 = std::to_string(mypairlist[it].get_h1());
578.                                     std::string s2 = std::to_string(mypairlist[it].get_h2());
579.                                     std::string s3 = std::to_string(mypairlist[it].get_uu());
580.                                     std::string s4 = std::to_string(mypairlist[it].get_dd());
581.                                     std::string s5 = std::to_string(mypairlist[it].get_ud());
582.                                     gl_st = gl_st + s1 + "|" + s2 + "|" + s3 + "|" + s4 + "|" +
    s5 + "\n";
583.                             }
584.                     }
585.             }
586.             gl_st=gl_st+ ("_____________________________________\n");
587.
588.     }
589.
590. % find non-conventional features
591.  void print_stat(int space, int backspace, int m_du, int m_uu, int ksorderl, int ksorderr, int
    skorderl, int skorderr, int tas)
592. {
593.     ofstream myfile("data.txt", ios::app);
594.     if (myfile.is_open())
595.     {
596.             LARGE_INTEGER frequency;
597.             QueryPerformanceFrequency(&frequency);
598.
599.             double total_time = ((double)myend.QuadPart - (double)mystart.QuadPart) /
    (double)frequency.QuadPart;
600.             double mins =  total_time / 60;
601.
602.            double cpm =space/ mins;
603.
604.             myfile << "Total typing time in seconds: " << total_time<<"\n";
605.             myfile << "Total typing time in minutes: " << mins << "\n";
606.             myfile << "Total number of words: " << space << "\n";
607.             myfile << "Charchters Per Munit: " << cpm << "\n";
608.             myfile << "Total number of errors: " << backspace << ", Percentage: " <<
    ((double)backspace / (k+1)) * 100 << "%\n";
609.             myfile << "Number of Caps Lock used: " << caplock / 2 << "\n";
610.             myfile << "Number of litters typed using CapLock: " << letter_caplock << "\n";
611.             myfile << "Number of litters typed on Numpad: " << numlock << "\n";
612.             myfile << "Total number of key pairs: " << k<< "\n";
613.             myfile << "Total number of minus DU: " << m_du << ", Percentage: " << ((double)m_du /
    k) * 100 << "%\n";
614.             myfile << "Total number of minus UU: " << m_uu << ", Percentage: " << ((double)m_uu /
    k) * 100 << "%\n";
615.             myfile << "Total number of times shift was released BEFORE litter (right shift): " <<
    skorderr << "\n";
616.             myfile << "Total number of times shift was released AFTER litter (right shift): " <<
    ksorderr << "\n";
617.             myfile << "Total number of times shift was released BEFORE litter (left shift): " <<
    skorderl<< "\n";
618.             myfile << "Total number of times shift was released AFTER litter (left shift): " <<
    ksorderl<< "\n";
```

```
619.             myfile <<
    "*************************************************************************************************
    ****************************\n";
620.             myfile << "The last task you have Completed was Task #:" << tas << "\n";
621.
622.             myfile.close();
623.     }
624.     else cout << "Unable to open file";
625. }
626.
627. % printing
628. void print_st(std::string st)
629. {
630.     ofstream myfile("data.txt", ios::app);
631.     if (myfile.is_open())
632.     {
633.             myfile << st;
634.             myfile.close();
635.     }
636.     else cout << "Unable to open file";
637. }
638.
639. void print_freq(LARGE_INTEGER x)
640. {
641.             std::string s1 = std::to_string(x.QuadPart);
642.             gl_st = gl_st + "Frequency: " + s1 + "\n";
643.
644. }
```

B.1.2 Main File

```
1.  #include "Header.h"
2.
3.  #pragma once
4.
5.  namespace Project1 {
6.
7.      using namespace std;
8.      using namespace System;
9.      using namespace System::ComponentModel;
10.     using namespace System::Collections;
11.     using namespace System::Windows::Forms;
12.     using namespace System::Data;
13.     using namespace System::Drawing;
14.
15.     /// Summary for MyForm
16.
17.     public ref class MyForm : public System::Windows::Forms::Form
18.     {
19.     public:
20.             MyForm(void)
21.             {
22.                     InitializeComponent();
23.
24.             }
25.
26.     protected:
27.
28.             ~MyForm()
29.             {
30.                     if (components)
31.                     {
32.                             delete components;
33.                     }
34.             }
35.     private: System::Windows::Forms::TextBox^  textBox1;
36.     private: System::Windows::Forms::Label^  label1;
37.     private: System::Windows::Forms::Button^  button1;
38.     private: System::Windows::Forms::Label^  label2;
39.     private: System::Windows::Forms::PictureBox^  pictureBox1;
40.
41.     private: System::Windows::Forms::Button^  button2;
```

```
42.     private: System::Windows::Forms::Button^  button3;
43.
44.     private: System::Windows::Forms::TextBox^  textBox2;
45.     private: System::Windows::Forms::PictureBox^  pictureBox2;
46.     private: System::Windows::Forms::PictureBox^  pictureBox3;
47.     protected:
48.
49.     private:
50.
51.             System::ComponentModel::Container ^components;
52.
53. #pragma region Windows Form Designer generated code
54.
55.             void InitializeComponent(void)
56.             {
57.                     System::ComponentModel::ComponentResourceManager^  resources = (gcnew
    System::ComponentModel::ComponentResourceManager(MyForm::typeid));
58.                     this->textBox1 = (gcnew System::Windows::Forms::TextBox());
59.                     this->label1 = (gcnew System::Windows::Forms::Label());
60.                     this->button1 = (gcnew System::Windows::Forms::Button());
61.                     this->label2 = (gcnew System::Windows::Forms::Label());
62.                     this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
63.                     this->button2 = (gcnew System::Windows::Forms::Button());
64.                     this->button3 = (gcnew System::Windows::Forms::Button());
65.                     this->textBox2 = (gcnew System::Windows::Forms::TextBox());
66.                     this->pictureBox2 = (gcnew System::Windows::Forms::PictureBox());
67.                     this->pictureBox3 = (gcnew System::Windows::Forms::PictureBox());
68.             (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))-
    >BeginInit();
69.             (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox2))-
    >BeginInit();
70.             (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox3))-
    >BeginInit();
71.                     this->SuspendLayout();
72.                     // textBox1
73.                     resources->ApplyResources(this->textBox1, L"textBox1");
74.                     this->textBox1->Name = L"textBox1";
75.                     this->textBox1->KeyDown += gcnew System::Windows::Forms::KeyEventHandler(this,
    &MyForm::textBox1_KeyDown);
76.                     this->textBox1->KeyUp += gcnew System::Windows::Forms::KeyEventHandler(this,
    &MyForm::textBox1_KeyUp);
77.                     // label1
78.                     resources->ApplyResources(this->label1, L"label1");
79.                     this->label1->Name = L"label1";
80.                     // button1
81.                     resources->ApplyResources(this->button1, L"button1");
82.                     this->button1->Name = L"button1";
83.                     this->button1->UseVisualStyleBackColor = true;
84.                     this->button1->Click += gcnew System::EventHandler(this,
    &MyForm::button1_Click);
85.                     // label2
86.                     resources->ApplyResources(this->label2, L"label2");
87.                     this->label2->Name = L"label2";
88.                     // pictureBox1
89.                     resources->ApplyResources(this->pictureBox1, L"pictureBox1");
90.                     this->pictureBox1->Name = L"pictureBox1";
91.                     this->pictureBox1->TabStop = false;
92.                     //
93.                     // button2
94.                     //
95.                     resources->ApplyResources(this->button2, L"button2");
96.                     this->button2->Name = L"button2";
97.                     this->button2->UseVisualStyleBackColor = true;
98.                     this->button2->Click += gcnew System::EventHandler(this,
    &MyForm::button2_Click);
99.                     // button3
100.                    resources->ApplyResources(this->button3, L"button3");
101.                    this->button3->Name = L"button3";
102.                    this->button3->UseVisualStyleBackColor = true;
103.                    this->button3->Click += gcnew System::EventHandler(this,
    &MyForm::button3_Click);
104.                    // textBox2
105.                    resources->ApplyResources(this->textBox2, L"textBox2");
106.                    this->textBox2->Name = L"textBox2";
```

```
107.                    this->textBox2->ReadOnly = true;
108.                    // pictureBox2
109.                    resources->ApplyResources(this->pictureBox2, L"pictureBox2");
110.                    this->pictureBox2->Name = L"pictureBox2";
111.                    this->pictureBox2->TabStop = false;
112.                    // pictureBox3
113.                    resources->ApplyResources(this->pictureBox3, L"pictureBox3");
114.                    this->pictureBox3->Name = L"pictureBox3";
115.                    this->pictureBox3->TabStop = false;
116.                    // MyForm
117.                    resources->ApplyResources(this, L"$this");
118.                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
119.                    this->Controls->Add(this->pictureBox3);
120.                    this->Controls->Add(this->pictureBox2);
121.                    this->Controls->Add(this->button3);
122.                    this->Controls->Add(this->button2);
123.                    this->Controls->Add(this->textBox2);
124.                    this->Controls->Add(this->pictureBox1);
125.                    this->Controls->Add(this->label2);
126.                    this->Controls->Add(this->button1);
127.                    this->Controls->Add(this->label1);
128.                    this->Controls->Add(this->textBox1);
129.                    this->Name = L"MyForm";
130.                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
    >pictureBox1))->EndInit();
131.                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
    >pictureBox2))->EndInit();
132.                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
    >pictureBox3))->EndInit();
133.                    this->ResumeLayout(false);
134.                    this->PerformLayout();
135.
136.            }
137.
138.#pragma endregion
139.
140.% key down action
141.    private: System::Void textBox1_KeyDown(System::Object^  sender,
    System::Windows::Forms::KeyEventArgs^  e) {
142.
143.                            LARGE_INTEGER t1, frequency,m={0};
144.                            std::string status = "";
145.
146.                            QueryPerformanceCounter(&t1);
147.                            QueryPerformanceFrequency(&frequency);
148.
149.                            if (mystart.QuadPart == m.QuadPart)
150.                            {
151.                                    mystart = t1;
152.                            }
153.
154.                            char mycodechar = getchar((int)e->KeyCode);
155.
156.                            if (returnsymbol((int)e->KeyCode) == '@' && !isnumlock((int)e-
    >KeyCode) && (int)e->KeyCode != VK_LSHIFT)
157.                            {
158.                                    //System.Diagnostics.Debug.WriteLine("Not a litter");
159.                                    status = "nonletter";
160.                            }
161.                            else if ((int)e->KeyCode == VK_CAPITAL)
162.                            {
163.                                    //System.Diagnostics.Debug.WriteLine("CapLock down");
164.                                    status = "capital";
165.                            }
166.                            else if (GetKeyState(VK_CAPITAL))
167.                            {
168.                                    //System.Diagnostics.Debug.WriteLine("capslock list d");
169.                                    status = "caplockitem";
170.                            }
171.                             else if (isnumlock((int)e->KeyCode))
172.                             {
173.                                    //System.Diagnostics.Debug.WriteLine("NumLock D");
174.                                    status = "numlock";
175.                             }
```

```
176.                                else if ((int)e->KeyCode == VK_SHIFT)
177.                                {
178.                                        int ret = GetKeyState(VK_LSHIFT);
179.                                        if (ret & 0x80000000)
180.                                        {
181.                                                status = "lshift";
182.                                        }
183.                                        else
184.                                        {
185.                                                status = "rshift";
186.                                        }
187.                                }
188.                                else
189.                                {
190.                                        int ret = GetKeyState(VK_LSHIFT);
191.                                        int ret2 = GetKeyState(VK_RSHIFT);
192.                                        if (ret & 0x80000000)
193.                                        {
194.                                                status = "lshiftitem";
195.                                        }
196.                                        else if (ret2 & 0x80000000)
197.                                        {
198.                                                status = "rshiftitem";
199.                                        }
200.                                }
201.                                Mynode anode (mycodechar, t1, "DOWN", status);
202.                                mylist[i]=anode;
203.                                i++;
204.    }
205.
645.    % key up action
206.private: System::Void textBox1_KeyUp(System::Object^  sender,
    System::Windows::Forms::KeyEventArgs^  e) {
207.
208.                                LARGE_INTEGER t2, frequency;
209.                                std::string status = "";
210.
211.                                QueryPerformanceCounter(&t2);
212.                                QueryPerformanceFrequency(&frequency);
213.
214.                                myend = t2;
215.                                char mycodechar = getchar((int)e->KeyCode);
216.                                                                        if (returnsymbol((int)e->KeyCode) ==
    '@' && !isnumlock((int)e->KeyCode) && (int)e->KeyCode != VK_SHIFT)
217.                                {
218.                                        status = "nonletter";
219.                                }
220.                                else if ((int)e->KeyCode == VK_CAPITAL)
221.                                {
222.                                        caplock++;
223.                                        status = "capital";
224.                                }
225.                                else if (GetKeyState(VK_CAPITAL))
226.                                {
227.                                        status = "caplockitem";
228.                                        letter_caplock++;
229.                                }
230.                                 else if (isnumlock((int)e->KeyCode))
231.                                 {
232.                                        numlock++;
233.                                        status = "numlock";
234.                                 }
235.                                else if ((int)e->KeyCode == VK_SHIFT)
236.                                {
237.                                        int ret = GetKeyState(VK_LSHIFT);
238.                                        if (ret & 0x80000000)
239.                                        {
240.                                                status = "lshift";
241.                                        }
242.                                        else
243.                                        {
244.                                                status = "rshift";
245.                                        }
246.                                }
```

```
247.                                else
248.                                {
249.                                        int ret = GetKeyState(VK_LSHIFT);
250.                                        int ret2 = GetKeyState(VK_RSHIFT);
251.                                        if (ret & 0x80000000)
252.                                        {
253.                                                status = "lshiftitem";
254.                                        }
255.                                        else if (ret2 & 0x80000000)
256.                                        {
257.                                                status = "rshiftitem";
258.                                        }
259.                                }
260.
261.                                Mynode anode(mycodechar, t2, "UP", status);
262.                                mylist[i]=anode;
263.                                i++;
264.
265.    }
266.
267.% press finish button
268.
269.    private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
270.
271.                                LARGE_INTEGER frequency;
272.                                double hold,dd,uu,ud;
273.                                int space_num = 1, backspace_num = 0, tri_num = 0, ksorderr = 0,
    skorderr = 0, r_shift = 0, ksorderl = 0, skorderl = 0, l_shift = 0, mi_du = 0, mi_uu=0;
274.                                std::string my_s = "";
275.                                QueryPerformanceFrequency(&frequency);
276.                                initingarr();   //keyboardlayout
277.                                print_freq(frequency);
278.
279.                                //print rawl list, order it and print ordered list
280.                                print_keyslist(mylist,0);
281.                                order_list(mylist,mynewlist);
282.                                print_keyslist(mynewlist,1);
283.
284.                                //find hold,dd,uu,ud
285.                                int my_iter = 0, my_iter2 = 0, my_iter3 = 0, my_iter4 = 0;
286.
287.                                if (my_iter2 != j) my_iter2++;
288.                                if (my_iter3 !=j) my_iter3++;
289.                                if (my_iter3 != j)my_iter3++;
290.                                if (my_iter4 != j)my_iter4++;
291.                                if (my_iter4 != j)my_iter4++;
292.                                if (my_iter4 != j)my_iter4++;
293.
294.
295.
296.                                while (my_iter != j)
297.                                {
298.                                        if (mynewlist[my_iter].get_type() == "DOWN")
299.                                        {
300.                                               if (my_iter2 != j)
301.                                               {
302.                                                       hold =
    (mynewlist[my_iter2].get_time().QuadPart - mynewlist[my_iter].get_time().QuadPart) * 1000.0 /
    frequency.QuadPart;
303.                                                       if (my_iter != 0)
304.                                                       {
305.                                                               add_h2( hold);
306.                                                       }
307.                                                       if (my_iter4 != j)
308.                                                       {
309.                                                               uu =
    (mynewlist[my_iter4].get_time().QuadPart - mynewlist[my_iter2].get_time().QuadPart) * 1000.0 /
    frequency.QuadPart;
310.                                                       }
311.                                                       if (my_iter3 != j)
312.                                                       {
313.                                                               ud =
    (mynewlist[my_iter3].get_time().QuadPart - mynewlist[my_iter2].get_time().QuadPart) * 1000.0 /
    frequency.QuadPart;
```

```
314.                                                    }
315.                                            }
316.                                            if (my_iter3 != j)
317.                                            {
318.                                                    dd = (mynewlist[my_iter3].get_time().QuadPart
    - mynewlist[my_iter].get_time().QuadPart) * 1000.0 / frequency.QuadPart;
319.                                                    my_s = mynewlist[my_iter].get_status() +
    mynewlist[my_iter2].get_status() + mynewlist[my_iter3].get_status() +
    mynewlist[my_iter4].get_status();
320.                                                    Mypair apair(mynewlist[my_iter].get_charc(),
    mynewlist[my_iter3].get_charc(), hold, 0, dd, uu, ud, my_s);
321.                                                    mypairlist[k]=apair;
322.                                                    k++;
323.                                                    if (ud < 0 && my_s.find("shift")==-1)
324.                                                    {
325.                                                            mi_du++;
326.                                                    }
327.                                                    if (uu < 0 && my_s.find("shift") == -1)
328.                                                    {
329.                                                            mi_uu++;
330.                                                    }
331.                                            }
332.                                    }
333.
334.                                     if (my_iter2 != j)
335.                                     {
336.                                             my_iter2++;
337.                                     }
338.                                     if (my_iter3 != j)
339.                                     {
340.                                             my_iter3++;
341.                                     }
342.                                     if (my_iter4 != j)
343.                                     {
344.                                             my_iter4++;
345.                                     }
346.                                     my_iter++;
347.                             }
348.                             add_h2_final(hold);
349.
350.                             //classify pairs
351.                             for (int it = 0; it != k; it++)
352.                             {
353.                                     if (mypairlist[it].get_char1() == (char)13 ||
    mypairlist[it].get_char2() == (char)13)
354.                                     {
355.                                             if (mypairlist[it].get_char1() == (char)13)
356.                                             {
357.                                                     space_num++;
358.                                                     tri_num = 0;
359.                                             }
360.                                     }
361.                                     else if ((mypairlist[it].get_char1() == ' ' &&
    mypairlist[it].get_char2() != '$') || mypairlist[it].get_char2() == ' ')
362.                                     {
363.                                             if (mypairlist[it].get_char1() == ' ')
364.                                             {
365.                                                     space_num++;
366.                                                     tri_num = 0;
367.                                             }
368.                                     }
369.                                     else if (mypairlist[it].get_char1() == (char)8 ||
    mypairlist[it].get_char2() == (char)8)
370.                                     {
371.                                             if (mypairlist[it].get_char1() == (char)8)
372.                                             {
373.                                                     backspace_num++;
374.                                                     tri_num = 0;
375.                                             }
376.                                     }
377.                                     else if (mypairlist[it].get_char1() ==
    mypairlist[it].get_char2())
378.                                     {
379.                                             if (isright(mypairlist[it].get_char1()))
```

```
                                                {
                                                        mypairlist[it].add_type("same_right");
                                                        add_tri_item( "Same | rightside",
    mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
                                                }
                                                else
                                                {
                                                        mypairlist[it].add_type("same_left");
                                                        add_tri_item( "Same | leftside",
    mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
                                                }
                                        }
                                        else if (mypairlist[it].get_char1().Equals('$') ||
    mypairlist[it].get_char2().Equals('$') || mypairlist[it].get_type().find("rshift")!=-1 ||
    mypairlist[it].get_type().find("lshift")!=-1)
                                        {
                                                if (mypairlist[it].get_char1() == '$' &&
    mypairlist[it].get_type()==("rshiftrshiftrshiftitem"))
                                                {
                                                        skorderr++;
                                                        r_shift++;
                                                        tri_num = 0;
                                                }
                                                else if (mypairlist[it].get_char1() == '$' &&
    mypairlist[it].get_type()==("lshiftrshiftlshiftitem"))
                                                {
                                                        skorderl++;
                                                        l_shift++;
                                                        tri_num = 0;
                                                }
                                                else if (mypairlist[it].get_char1() == '$' &&
    mypairlist[it].get_type()==("rshiftrshiftrshiftitemrshiftitem"))
                                                {
                                                        ksorderr++;
                                                        r_shift++;
                                                        tri_num = 0;
                                                }
                                                else if (mypairlist[it].get_char1() == '$' &&
    mypairlist[it].get_type()==("lshiftrshiftlshiftitemlshiftitem"))
                                                {
                                                        ksorderl++;
                                                        l_shift++;
                                                        tri_num = 0;
                                                }
                                        }
                                        else if (mypairlist[it].get_type().find("lock") != -1 ||
    mypairlist[it].get_type().find("capital") != -1 || mypairlist[it].get_type().find("nonletter")!=-
    1)
                                        {
                                                tri_num = 0;
                                        }
                                        else
                                        {
                                                if (isright(mypairlist[it].get_char1()) &&
    isright(mypairlist[it].get_char2()))
                                                {
                                                        if (mypairlist[it].get_char1() !=
    mypairlist[it].get_char2() && prox(it))
                                                        {

    mypairlist[it].add_type("adj_right");
                                                                add_tri_item("Adjacent | rightside",
    mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
                                                        }
                                                        else if (mypairlist[it].get_char1() !=
    mypairlist[it].get_char2() && !prox(it))
                                                        {
                                                                if (prox2( it))
                                                                {

    mypairlist[it].add_type("second_right");
```

```
435.                                                                    add_tri_item( "Second |
    rightside", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
436.                                                                }
437.                                                                else if (prox3( it))
438.                                                                {
439.
    mypairlist[it].add_type("third_right");
440.                                                                    add_tri_item( "Third |
    rightside", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
441.                                                                }
442.                                                                else if (prox4( it))
443.                                                                {
444.
    mypairlist[it].add_type("fourth_right");
445.                                                                    add_tri_item("Fourth |
    righside", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
446.                                                                }
447.                                                                else
448.                                                                {
449.
    mypairlist[it].add_type("non_right");
450.                                                                    add_tri_item( "not adj |
    rightside", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
451.                                                                }
452.                                                        }
453.                                                }
454.                                                else if (!isright(mypairlist[it].get_char1()) &&
    !isright(mypairlist[it].get_char2()))
455.                                                {
456.                                                        if (mypairlist[it].get_char1() !=
    mypairlist[it].get_char2() && prox(it))
457.                                                        {
458.                                                                mypairlist[it].add_type("adj_left");
459.                                                                add_tri_item( "Adjacent | leftside",
    mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
460.                                                        }
461.                                                        else if (mypairlist[it].get_char1() !=
    mypairlist[it].get_char2() && !prox(it))
462.                                                        {
463.                                                                if (prox2( it))
464.                                                                {
465.
    mypairlist[it].add_type("second_left");
466.                                                                    add_tri_item( "Second |
    leftside", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
467.                                                                }
468.                                                                else if (prox3( it))
469.                                                                {
470.
    mypairlist[it].add_type("third_left");
471.                                                                    add_tri_item( "Third |
    leftside", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
472.                                                                }
473.                                                                else if (prox4( it))
474.                                                                {
475.
    mypairlist[it].add_type("fourth_left");
476.                                                                    add_tri_item( "Fourth |
    leftside", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
477.                                                                }
478.                                                                else
479.                                                                {
480.
    mypairlist[it].add_type("non_left");
```

```
481.                                                                    add_tri_item( "not adj |
    leftside", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
482.                                                            }
483.                                                    }
484.                                            }
485.                                            else
486.                                            {
487.                                                    if (mypairlist[it].get_char1() !=
    mypairlist[it].get_char2() && prox(it))
488.                                                    {
489.                                                            mypairlist[it].add_type("adj_rl");
490.                                                            add_tri_item( "Adjacent | l_r",
    mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
491.                                                    }
492.                                                    else if (mypairlist[it].get_char1() !=
    mypairlist[it].get_char2() && !prox(it))
493.                                                    {
494.                                                            if (prox2( it))
495.                                                            {
496.
    mypairlist[it].add_type("second_rl");
497.                                                                    add_tri_item("Second |
    l_r", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
498.                                                            }
499.                                                            else if (prox3( it))
500.                                                            {
501.
    mypairlist[it].add_type("third_rl");
502.                                                                    add_tri_item( "Third |
    l_r", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
503.                                                            }
504.                                                            else if (prox4( it))
505.                                                            {
506.
    mypairlist[it].add_type("fourth_rl");
507.                                                                    add_tri_item( "Fourth |
    l_r", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
508.                                                            }
509.                                                            else
510.                                                            {
511.
    mypairlist[it].add_type("non_rl");
512.                                                                    add_tri_item("not adj |
    l_r", mypairlist[it].get_h1(), mypairlist[it].get_h2(), mypairlist[it].get_dd(),
    mypairlist[it].get_uu(), mypairlist[it].get_ud(), tri_num);
513.                                                            }
514.                                                    }
515.                                            }
516.                                    }
517.                            }
518.                            print_pair();
519.                            print_keypair();
520.                            print_tri_list();
521.
522.                            print_st(gl_st);
523.                            print_stat(space_num, backspace_num, mi_du, mi_uu, ksorderl,
    ksorderr, skorderl, skorderr,task);
524.
525.
526.                            exit(0);
527.    }
528.
529.% text for different tasks
530.
531.    private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
532.                            textBox2->Text = "Photo-sharing sites such as Pinterest and
    Instagram are at the forefront of a new wave of social networks which showcase beautiful images
    uploaded by artists, brands and the public. Marketers are using the sites to drive social shopping
    and inspire people to collect and share pictures of their favourite products.\r\n\r\nFashion chain
```

```
    Topshop collaborated with pinboard site Pinterest in November to encourage shoppers to pin their
    favourite products – that is the equivalent of clicking Like on Facebook - from the retailer's
    website on to their own Pinterest pages, which are known as \"boards\". This helped shoppers
    create personalised Pinterest Christmas gift guides. The most pinned products were featured on the
    Topshop homepage and shoppers could enter their Christmas-themed Pinterest boards into a
    competition to win prizes at the store.\r\n\r\nThe chain also installed giant touchscreens in
    flagship stores in London and the US so shoppers could see the most popular pins. Popular items on
    display had swing tags attached stating that they were most pinned products.";
533.                             pictureBox2->Visible = true;
534.                             pictureBox3->Visible = false;
535.                             gl_st = gl_st +"Task # 1:\n";
536.                             task = 1;
537.     }
538.
539.     private: System::Void button3_Click(System::Object^  sender, System::EventArgs^  e) {
540.                             textBox2->Text = "Topshop's global marketing and communications
    director says the campaign aimed to get people to collect gift ideas and share them with family
    and friends in an attempt to create a social shopping platform. This is a much-touted development
    in retail where people will increasingly use social media to help them choose what to
    buy.\r\n\r\nShe believes Pinterest's appeal to a young, female audience is a good fit with
    Topshop's customers.\r\n\r\nUK country manager for Pinterest says the great attraction of the
    network is that it helps people plan what they want to achieve from organising a holiday to
    cooking a meal.The three most pinned topics on the network are for pictures related to travel,
    party planning and fashion.\r\n\r\n While Facebook is mainly for posting about what has just
    happened and Twitter is great for talking about the here and now, the attraction of Pinterest is
    its use for future plans.Because of how people use Pinterest, it is a tool which helps facilitate
    future planning - to plan what you are going to cook tomorrow night for instance.";
541.                            pictureBox3->Visible = true;
542.                             pictureBox2->Visible = false;
543.                             gl_st = gl_st + "Task # 2:\n";
544.                            task = 2;
545. }
546.
547.     private: System::Void button4_Click(System::Object^  sender, System::EventArgs^  e) {
548.                             textBox2->Text = "For example, when you are in the supermarket, you
    have the mobile app and you have created a pin board of meals with recipe pins and you can browse
    the aisles looking for ingredients. Pinterest is also powerful for discovering new products and
    prospective experiences. It is like Google, a search engine. It is an environment where you
    discover and do things, because all pins have a destination and it can work for a wide range of
    businesses.\r\n\r\nBrands can get involved by going to the Business Centre on the Pinterest
    website, where they sign up for a verified account, ensuring it links through to their official
    website.They can also add a Pin It button to their site.They can make use of different types of
    rich pins containing extra information.\r\n\r\n For instance, the newly - launched place pins
    include a map, address and phone number while article pins include headline, author and story
    description, helping pinners find and save stories that matter to them.There are also product
    pins, recipe pins and movie pins.";
549.                             pictureBox2->Visible = "False";
550.                             gl_st = gl_st + "Task # 3:\n";
551.                     task = 3;
552. }
553.
554.     private: System::Void button5_Click(System::Object^  sender, System::EventArgs^  e) {
555.                             textBox2->Text = "Meanwhile, Instagram has 160 million monthly
    active users globally, with 65 million photos uploaded every day and with a billion likes per day.
    The Facebook-owned company representer says users spend three times as long on Instagram as they
    do on Pinterest and twice as long as on Twitter.\r\n\r\nUK brands that have used Instagram include
    Burberry, which has posted live pictures of its fashion shows including London Fashion Week. The
    fashion retailer has grown its Instagram following organically to over one million. Moreover, Red
    Bull has also documented a cliff-diving competition in Wales through Instagram. And Jaguar has
    published a series of short films as part of an Instagram video campaign to promote its F-Type
    Coupe launch. Brands may find that pictures speak louder than words when it comes to scoring a hit
    with social shoppers.\r\n\r\nPublishers are also active on the photo-sharing sites. The director
    of partnerships at Random House in the US, says that many authors use Instagram to post teasers
    about upcoming books, and give a behind-the-scenes look at the life of a writer.";
556.                             gl_st = gl_st + "Task # 4:\n";
557.                     task = 4;
558. }
559.
560.     private: System::Void button6_Click(System::Object^  sender, System::EventArgs^  e) {
561.                             textBox2->Text = "Rome today is one of the most important tourist
    destinations of the world, due to the incalculable immensity of its archaeological and artistic
    treasures, as well as for the charm of its unique traditions, the beauty of its panoramic views,
    and the majesty of its magnificent villas (parks).\r\n\r\nAmong its most significant resources are
    the many museums - Musei Capitolini, the Vatican Museums and the Galleria Borghese - aqueducts,
```

```
    fountains, churches, palaces, historical buildings, and the Catacombs. It also contains numerous
    ancient sites, including the Forum Romanum, Trajan's Market, Trajan's Forum, the Colosseum, and
    the Pantheon, to name a few.\r\n\r\nRome contains a vast and impressive collection of art,
    sculpture, fountains, mosaics, frescos, and paintings, from all different periods. It first became
    a major artistic center during ancient Rome, with forms of important Roman art such as
    architecture, painting, sculpture and mosaic work. Metal-work, coin die and gem engraving, ivory
    carvings, figurine glass, pottery and book illustrations are considered to be minor forms of Roman
    artwork too.";
562.                                gl_st = gl_st + "Task # 5:\n";
563.                        task = 5;
564.}
565.
566.    private: System::Void button7_Click(System::Object^  sender, System::EventArgs^  e) {
567.                                textBox2->Text = "Rome later became a major center of Renaissance
    art, since the popes spent vast sums of money for the constructions of grandiose basilicas,
    palaces, piazzas and public buildings in general. The city was affected greatly by the baroque,
    and Rome became the home of numerous artists and architects, such as Bernini, Caravaggio,
    Carracci, Borromini and Cortona.\r\n\r\nRome was one of the centers of the Grand Tour, when
    wealthy, young English and other European aristocrats visited the city to learn about ancient
    Roman culture, art, philosophy and architecture. Rome hosted a great number of neoclassical and
    rococo artists, such as Pannini and Bernardo Bellotto.\r\n\r\nRome has a growing stock of
    contemporary and modern art and architecture. The National Gallery of Modern Art has works by
    Balla, Morandi, Pirandello, Carra, De Chirico, De Pisis, Guttuso, Fontana, Burri, Mastroianni,
    Turcato, Kandisky and Cezanne on permanent exhibition. It is one of Rome's most ambitious modern
    architecture projects alongside Renzo Piano's Auditorium Parco della Musica, Massimiliano Fuksas
    Rome Convention Center and Centro Congressi Italia EUR.";
568.                                gl_st = gl_st + "Task # 6:\n";
569.                        task = 6;
570.}
571.
572.    private: System::Void button8_Click(System::Object^  sender, System::EventArgs^  e) {
573.                        textBox2->Text = "\r\nPlease type two or three paragraphs long about any
    holiday experience that you've had.\r\n\r\nNote: Please free type - Don't copy.";
574.
575.
576.                        gl_st = gl_st + "Task # 7:\n";
577.                        task = 7;
578.}
579.
580.    private: System::Void button9_Click(System::Object^  sender, System::EventArgs^  e) {
581.                        textBox2->Text = "\r\nPlease type another two or three paragraphs long of any
    content you like.\r\nEg: song lyrics, opinion about a current issue, plans for next year …
    etc.\r\n\r\nNote: Please free type - Don't copy.";
582.                        gl_st = gl_st + "Task # 8:\n";
583.                        task = 8;
584.}
585.};
586.}
```

B. 2 Outlier Discarding and Data Scaling

```
1.  clear all;
2.  clc;
3.
4.  % outlier descarding
5.  vector=importdata('m.txt');
6.  C=vector;
7.  for i=1:size(vector,2)
8.          A=C(:,i);
9.          m = mean(A);
10.         sd = std(A);
11.         C=C( A<=(3*(m+sd )),:);
12.         m = mean(A);
13.         sd = std(A);
14.
15. end
16. fid = fopen('Mym.txt','wt');
17. for ii = 1:size(C,1)
18.     fprintf(fid,'%g|',C(ii,:));
```

```
    fprintf(fid,'\n');
end
fclose(fid);

%scalling
vector2=importdata('Mym.txt');
C2=vector2;
for i=1:size(vector2,2)
        A2=C2(:,i);
        b=scaledata(A2,0,1);
        C2(:,i)=b;
end
fid = fopen('Mymatrix.txt','wt');
for ii = 1:size(C2,1)
    fprintf(fid,'%g\t',C2(ii,:));
    fprintf(fid,'\n');
end
fclose(fid);

%chuncking
vector3=importdata('Mymatrix.txt');
chunck= floor((size(vector3,1))/20)
chunck_size=chunck;

mynum=1;
for i=1:20
        A2=vector3(mynum:chunck_size,:);
        mynum=mynum+chunck;
        chunck_size= chunck_size+chunck;
        b=mean(A2);
        T2(i,:)=b;
end
fid = fopen('chunks.txt','wt');
for ii = 1:size(T2,1);
    fprintf(fid,'%g\t',T2(ii,:));
    fprintf(fid,'\n');
end
fclose(fid);
```

B. 3 ACO

```
clear all;
clc;

%Initilization
NoA=55;      %num of ants
NoIT=100;    %num of itration
NoF=5;       %number of features for each ant
IP=1.0;      %initial phermon
my_accuracy=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];    %array of accuracy
alpha= 1.0;
beta= 0.1;
gama=0.95;
gama2=0.2;
my_gamma = 1;
accuracy_subset=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];  % array of ants best accuracy
maxit_ant=0;
maxit_feat=[0 0 0 0 0];
maxit_accur=0;
all_ant_nodes=[0 0 0 0 0;0 0 0 0 0; 0 0 0 0 0; 0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0
0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0; 0 0 0 0 0; 0 0 0 0 0;0 0 0 0 0;0 0 0 0
0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0; 0 0 0 0 0; 0 0 0 0 0;0 0 0
0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0; 0 0 0 0 0; 0 0
0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0; 0
0 0 0 0; 0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0
0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0];

```

```
%right format
b=importdata('m.txt');
fid = fopen('Mym.txt','wt');
for ii = 1:size(b,1)
    fprintf(fid,'%g,',b(ii,:));
    fprintf(fid,'\n');
end
fclose(fid);

b2=importdata('m2.txt');
fid2 = fopen('Mym2.txt','wt');
for ii = 1:size(b2,1)
    fprintf(fid,'%g,',b2(ii,:));
    fprintf(fid,'\n');
end
fclose(fid2);

% find accuracy of each feature
a=importdata('mym.txt');    %train data

a2=importdata('mym2.txt');    %test data

%for each feature
for jj=2:size(a,2)
    fid = fopen('Myfeat.txt','wt');
    %extract column
    for ii = 1:size(a,1)
        fprintf(fid,'%g,',a(ii,1));
        fprintf(fid,'%g,',a(ii,jj));
        fprintf(fid,'\n');
    end
    fclose(fid);

    fid2 = fopen('Myfeat2.txt','wt');
    %extract column
    for ii = 1:size(a2,1)
        fprintf(fid2,'%g,',a2(ii,1));
        fprintf(fid2,'%g,',a2(ii,jj));
        fprintf(fid2,'\n');
    end
    fclose(fid2);

    %LIBSVM format
    SPECTF = csvread('Myfeat.txt'); % read a csv file
    labels = SPECTF(:, 1); % labels from the 1st column
    features = SPECTF(:, 2:end);
    features_sparse = sparse(features); % features must be in a sparse matrix
    libsvmwrite('my.txt', labels, features_sparse);

    SPECTF2 = csvread('Myfeat2.txt'); % read a csv file
    labels2 = SPECTF2(:, 1); % labels from the 1st column
    features2 = SPECTF2(:, 2:end);
    features_sparse2 = sparse(features2); % features must be in a sparse matrix
    libsvmwrite('my2.txt', labels2, features_sparse2);

    %classification accuracy
    [y, x] = libsvmread('my.txt');

    [y2, x2] = libsvmread('my2.txt');

    % Libsvm options
    % -s 0 : classification
    % -t 2 : RBF kernel
    % -g : gamma in the RBF kernel

    model = svmtrain(y, x, sprintf('-s 0 -t 2 -g %g', my_gamma));

    % Display training accuracy
    [predicted_label, accuracy, decision_values] = svmpredict(y2, x2, model);

    my_accuracy(jj)=accuracy(1)/100;
```

```
92. end
93.
94. %feature Intilization
95. feature1=struct('phermon',IP,'accuracy',my_accuracy(2));     %my_accuracy(1)=0
96. feature2=struct('phermon',IP,'accuracy',my_accuracy(3));
97. feature3=struct('phermon',IP,'accuracy',my_accuracy(4));
98. feature4=struct('phermon',IP,'accuracy',my_accuracy(5));
99. feature5=struct('phermon',IP,'accuracy',my_accuracy(6));
100.feature6=struct('phermon',IP,'accuracy',my_accuracy(7));
101.feature7=struct('phermon',IP,'accuracy',my_accuracy(8));
102.feature8=struct('phermon',IP,'accuracy',my_accuracy(9));
103.feature9=struct('phermon',IP,'accuracy',my_accuracy(10));
104.feature10=struct('phermon',IP,'accuracy',my_accuracy(11));
105.feature11=struct('phermon',IP,'accuracy',my_accuracy(12));
106.feature12=struct('phermon',IP,'accuracy',my_accuracy(13));
107.feature13=struct('phermon',IP,'accuracy',my_accuracy(14));
108.feature14=struct('phermon',IP,'accuracy',my_accuracy(15));
109.feature15=struct('phermon',IP,'accuracy',my_accuracy(16));
110.feature16=struct('phermon',IP,'accuracy',my_accuracy(17));
111.feature17=struct('phermon',IP,'accuracy',my_accuracy(18));
112.feature18=struct('phermon',IP,'accuracy',my_accuracy(19));
113.feature19=struct('phermon',IP,'accuracy',my_accuracy(20));
114.feature20=struct('phermon',IP,'accuracy',my_accuracy(21));
115.feature21=struct('phermon',IP,'accuracy',my_accuracy(22));
116.feature22=struct('phermon',IP,'accuracy',my_accuracy(23));
117.feature23=struct('phermon',IP,'accuracy',my_accuracy(24));
118.feature24=struct('phermon',IP,'accuracy',my_accuracy(25));
119.feature25=struct('phermon',IP,'accuracy',my_accuracy(26));
120.feature26=struct('phermon',IP,'accuracy',my_accuracy(27));
121.feature27=struct('phermon',IP,'accuracy',my_accuracy(28));
122.feature28=struct('phermon',IP,'accuracy',my_accuracy(29));
123.feature29=struct('phermon',IP,'accuracy',my_accuracy(30));
124.feature30=struct('phermon',IP,'accuracy',my_accuracy(31));
125.feature31=struct('phermon',IP,'accuracy',my_accuracy(32));
126.feature32=struct('phermon',IP,'accuracy',my_accuracy(33));
127.feature33=struct('phermon',IP,'accuracy',my_accuracy(34));
128.feature34=struct('phermon',IP,'accuracy',my_accuracy(35));
129.feature35=struct('phermon',IP,'accuracy',my_accuracy(36));
130.feature36=struct('phermon',IP,'accuracy',my_accuracy(37));
131.feature37=struct('phermon',IP,'accuracy',my_accuracy(38));
132.feature38=struct('phermon',IP,'accuracy',my_accuracy(39));
133.feature39=struct('phermon',IP,'accuracy',my_accuracy(40));
134.feature40=struct('phermon',IP,'accuracy',my_accuracy(41));
135.feature41=struct('phermon',IP,'accuracy',my_accuracy(42));
136.feature42=struct('phermon',IP,'accuracy',my_accuracy(43));
137.feature43=struct('phermon',IP,'accuracy',my_accuracy(44));
138.feature44=struct('phermon',IP,'accuracy',my_accuracy(45));
139.feature45=struct('phermon',IP,'accuracy',my_accuracy(46));
140.feature46=struct('phermon',IP,'accuracy',my_accuracy(47));
141.feature47=struct('phermon',IP,'accuracy',my_accuracy(48));
142.feature48=struct('phermon',IP,'accuracy',my_accuracy(49));
143.feature49=struct('phermon',IP,'accuracy',my_accuracy(50));
144.feature50=struct('phermon',IP,'accuracy',my_accuracy(51));
145.feature51=struct('phermon',IP,'accuracy',my_accuracy(52));
146.feature52=struct('phermon',IP,'accuracy',my_accuracy(53));
147.feature53=struct('phermon',IP,'accuracy',my_accuracy(54));
148.feature54=struct('phermon',IP,'accuracy',my_accuracy(55));
149.feature55=struct('phermon',IP,'accuracy',my_accuracy(56));
150.
151.
152.% array of features
153.arr_feat=[feature1, feature2,feature3, feature4,feature5, feature6,feature7, feature8,
    feature9, feature10, feature11, feature12,feature13, feature14,feature15,
    feature16,feature17, feature18, feature19, feature20, feature21, feature22,feature23,
    feature24,feature25, feature26,feature27, feature28, feature29, feature30, feature31,
    feature32,feature33, feature34,feature35, feature36,feature37, feature38, feature39,
    feature40, feature41, feature42,feature43, feature44,feature45, feature46,feature47,
    feature48, feature49, feature50, feature51, feature52,feature53, feature54,feature55];
154.
155.% for all eatrations
156.for it=1 : NoIT
```

```
% for all ants

%randomise ants first selected features
ants=randperm(55);
ants=ants(1:55);

%features not used for ants
for i=1 :NoA
    not_used_feat(i,:)=[1 2 3 4 5 6 7 8 9 10  11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55];
    m=ants(1,i);
    not_used_feat(i,m)=0;
end

%for each ant
%denomenator
den=0;
for k=1 : size(arr_feat,2)
    den=den + ((arr_feat(1,k).phermon .^alpha).* (arr_feat(1,k).accuracy .^beta));
end


%transition rule
p=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
ants_subset= [0 0 0 0 0];

for i=1 : NoA
    for m= 1 : NoF            %number of features vesited
        max_p=0;
        max_feat=0;
        for j=1 : size(not_used_feat,2)
            if (not_used_feat(i,j)~= 0)
                nom=(arr_feat(1,j).phermon .^alpha).* (arr_feat(1,j).accuracy .^beta);   %nomenator
                p(1,j)=nom/den;
                if p(1,j)>max_p
                    max_p=p(1,j);
                    max_feat=j;
                    ants_subset(i,m)=max_feat;
                end
            end
        end
    not_used_feat(i,max_feat)=0;
    end
end

all_ants_nodes=ants_subset;

%choose best ant (subset)
best_ant_accuracy=0;
myant_set=[0 0 0 0 0];

my_a=importdata('mym.txt');    %train data
my_a2=importdata('mym2.txt');    %test data

for mm=1 : size (ants_subset,1)
    for mmm=1 : size(ants_subset,2)
    myant_set(mmm)=ants_subset(mm,mmm);
    end

    fid = fopen('selected_feature.txt','wt');
    %extract column
    for ii = 1:size(my_a,1)
        fprintf(fid,'%g,',my_a(ii,1));
        if (myant_set(1)~=0)
            fprintf(fid,'%g,',my_a(ii,(myant_set(1)+1))); %mm=1 is the lable
        end
        if (myant_set(2)~=0)
```

```
                fprintf(fid,'%g,',my_a(ii,(myant_set(2)+1)));
            end
            if (myant_set(3)~=0)
                fprintf(fid,'%g,',my_a(ii,(myant_set(3)+1)));
            end
            if (myant_set(4)~=0)
                fprintf(fid,'%g,',my_a(ii,(myant_set(4)+1)));
            end
            if (myant_set(5)~=0)
                fprintf(fid,'%g,',my_a(ii,(myant_set(5)+1)));
            end
            fprintf(fid,'\n');
        end
        fclose(fid);

        fid2 = fopen('selected_feature2.txt','wt');
        %extract column
        for ii = 1:size(my_a2,1)
            fprintf(fid2,'%g,',my_a2(ii,1));
            if (myant_set(1)~=0)
                fprintf(fid2,'%g,',my_a2(ii,(myant_set(1)+1))); %mm=1 is the lable
            end
            if (myant_set(2)~=0)
                fprintf(fid2,'%g,',my_a2(ii,(myant_set(2)+1)));
            end
            if (myant_set(3)~=0)
                fprintf(fid2,'%g,',my_a2(ii,(myant_set(3)+1)));
            end
            if (myant_set(4)~=0)
                fprintf(fid2,'%g,',my_a2(ii,(myant_set(4)+1)));
            end
            if (myant_set(5)~=0)
                fprintf(fid2,'%g,',my_a2(ii,(myant_set(5)+1)));
            end
            fprintf(fid2,'\n');
        end
        fclose(fid2);

        %LIBSVM format
        SPECTF = csvread('selected_feature.txt'); % read a csv file
        labels = SPECTF(:, 1); % labels from the 1st column
        features = SPECTF(:, 2:end);
        features_sparse = sparse(features); % features must be in a sparse matrix
        libsvmwrite('my_selected_fearures.txt', labels, features_sparse);

        SPECTF2 = csvread('selected_feature2.txt'); % read a csv file
        labels2 = SPECTF2(:, 1); % labels from the 1st column
        features2 = SPECTF2(:, 2:end);
        features_sparse2 = sparse(features2); % features must be in a sparse matrix
        libsvmwrite('my_selected_features2.txt', labels2, features_sparse2);

        %classification accuracy
        [y, x] = libsvmread('my_selected_fearures.txt');
        [y2, x2] = libsvmread('my_selected_features2.txt');


        % Libsvm options
        % -s 0 : classification
        % -t 2 : RBF kernel
        % -g : gamma in the RBF kernel

        model = svmtrain(y, x, sprintf('-s 0 -t 2 -g %g', my_gamma));

        % Display training accuracy
        [predicted_label, accuracy, decision_values] = svmpredict(y2, x2, model);

        accuracy_subset(1,mm)= accuracy(1)/100;

        if (best_ant_accuracy<accuracy(1)/100)
            best_ant_accuracy=accuracy(1)/100;
            best_ant=mm;
```

```
            best_antfeature=myant_set;
        end
    end


    % phermon evaporation rate
    evap_rate=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
    leng_subset=NoF;                % fixed
    for ii=1 : NoA
        acc_subset=accuracy_subset(1,ii);
        ph_evap=(gama.*(acc_subset/leng_subset)) + ((1-gama).*(NoA-leng_subset)/NoA);
        evap_rate(1,ii)=ph_evap;
    end

    % update phermon
    %phermone evaboration
    for ii= 1 : size(arr_feat,2)
        arr_feat(ii).phermon=(1-gama2)*arr_feat(ii).phermon;
    end

    %phermone update for every ant
    for i= 1 : size(all_ants_nodes,1)
        for j=1 : size(all_ants_nodes,2)
        arr_feat(all_ants_nodes(i,j)).phermon=(1-gama2)*arr_feat(all_ants_nodes(i,j)).phermon+evap_rate(i);
        end
    end

    if (maxit_accur<best_ant_accuracy)
    maxit_ant=best_ant;         %best itration solution
    maxit_feat=best_antfeature;
    maxit_accur=best_ant_accuracy;
    end

    maxit_ant
    maxit_feat

    fid = fopen('test.txt','a');
    fprintf(fid,'%g,',best_ant_accuracy);
    fprintf(fid,'%g,',maxit_accur);
    fprintf(fid,'%g,',best_ant);
    fprintf(fid,'%g,',maxit_ant);
    fprintf(fid,'%g,',best_antfeature);
    fprintf(fid,'\n');
    fclose(fid);

End
```

B. 4 SVMs Classification

```
clear all;
clc;

b=importdata('mym_train.txt');
fid = fopen('mformulty_train.txt','wt');
for ii = 1:size(b,1)
    fprintf(fid,'%g,',b(ii,:));
    fprintf(fid,'\n');
end
fclose(fid);

b2=importdata('mym_test.txt');
fid2 = fopen('mformulty_test.txt','wt');
for ii = 1:size(b2,1)
    fprintf(fid2,'%g,',b2(ii,:));
    fprintf(fid2,'\n');
end
```

```
fclose(fid2);

%LIBSVM format
SPECTF = csvread('mformulty_train.txt'); % read a csv file
labels = SPECTF(:, 1); % labels from the 1st column
features = SPECTF(:, 2:end);
features_sparse = sparse(features); % features must be in a sparse matrix
libsvmwrite('mymulty_train.txt', labels, features_sparse);

SPECTF2 = csvread('mformulty_test.txt'); % read a csv file
labels2 = SPECTF2(:, 1); % labels from the 1st column
features2 = SPECTF2(:, 2:end);
features_sparse2 = sparse(features2); % features must be in a sparse matrix
libsvmwrite('mymulty_test.txt', labels2, features_sparse2);

 [trainY trainX] = libsvmread('mymulty_train.txt');
 [testY testX] = libsvmread('mymulty_test.txt');

model = ovrtrain(trainY, trainX, '-c 10 -g 1');
[pred ac decv] = ovrpredict(testY, testX, model);
fprintf('Accuracyyy = %g%%\n', ac * 100);
```

B. 5 DTs Classification

```
clear all;
clc;
% training and pruning
training = importdata('non_time_data.txt');
labels   = importdata('non_time_lables.txt');
testing = importdata('non_time_test.txt');
tree = ClassificationTree.fit(training,labels)
tree2 = prune(tree,'level',2)
% prediction
prediction = predict(tree2, testing)
view(tree2)
view(tree2,'Mode','graph')
```

Appendix C

Results Details

C.1 Original Key-paring Method

C.1.1 Legitimate User Testing

User	Try	Feature Set								
		ALL	Hold1|Hold2	UU|DD|UD	Hold1|UD|Hold2	UU	DD	UD	H1	H2
User 1	Try1	1	1	1	1	1	1	1	1	1
	Try2	0	1	0	1	1	1	1	1	1
User 2	Try1	0	0	1	0	1	1	1	1	1
	Try2	0	0	1	0	1	1	1	1	1
User 3	Try1	0	0	1	0	1	1	1	0	1
	Try2	0	0	0	0	1	1	1	0	1
User 4	Try1	0	1	1	1	1	1	1	1	1
	Try2	0	1	0	0	1	1	1	1	1
User 5	Try1	0	1	1	1	1	1	1	1	1
	Try2	0	1	1	1	1	1	1	1	1
User 6	Try1	0	1	0	0	1	1	1	1	1
	Try2	0	1	1	1	0	0	1	1	1
User 7	Try1	0	1	1	1	1	1	1	1	1
	Try2	0	0	1	0	1	1	1	0	1
User 8	Try1	0	1	0	1	1	1	1	1	1
	Try2	0	1	0	0	1	1	1	1	1
User 9	Try1	0	1	0	0	1	1	1	1	1
	Try2	0	1	1	1	1	1	1	1	1
User 10	Try1	0	1	0	1	1	1	1	1	1
	Try2	0	1	0	1	1	1	1	1	1
User 11	Try1	0	1	0	0	0	0	0	1	1
	Try2	0	1	0	0	1	1	1	1	1
User 12	Try1	0	0	0	0	1	1	1	1	1
	Try2	0	0	0	0	1	1	1	0	1
User 13	Try1	1	1	1	1	1	1	1	1	1
	Try2	1	1	1	1	1	1	1	1	1

User 14	Try1	0	1	0	0	1	0	1	1	0
	Try2	0	1	0	1	1	0	1	1	0
User 15	Try1	1	1	1	1	1	1	1	1	1
	Try2	0	0	0	0	0	0	0	0	0

C.1.2 Imposter user testing

Victim	Try	Feature Set									Imposter
		ALL	Hold1|Hold2	UU|DD|UD	Hold1|UD|Hold2	UU	DD	UD	H1	H2	
User 1	Try1	0	0	1	0	1	1	1	1	1	User 2
	Try2	0	0	0	0	1	1	1	1	1	
User 2	Try1	0	0	1	0	1	1	1	0	1	User 3
	Try2	0	1	1	1	1	1	1	1	1	
User 3	Try1	0	0	0	0	1	1	1	1	1	User 4
	Try1	0	1	0	0	1	1	1	1	1	
User 4	Try2	0	0	0	0	1	1	1	1	1	User 5
	Try2	0	1	1	0	1	1	1	1	1	
User 5	Try1	0	1	1	0	1	1	1	1	1	User 6
	Try2	0	1	0	0	1	1	1	1	1	
User 6	Try1	0	0	0	0	1	1	1	1	1	User 7
	Try2	1	1	1	1	1	1	1	1	1	
User 7	Try2	0	1	1	1	1	1	1	1	1	User 8
	Try2	0	1	1	1	1	1	1	1	1	
User 8	Try1	0	1	1	1	1	1	1	1	1	User 9
	Try2	0	0	1	0	1	1	1	1	1	
User 9	Try2	0	1	0	0	1	0	0	1	1	User 10
	Try2	0	1	0	0	1	0	0	1	1	
User 10	Try1	0	1	0	1	1	1	1	1	1	User 11
	Try2	0	1	0	1	1	1	1	1	1	
User 11	Try2	0	1	0	0	1	1	1	1	1	User 12
	Try2	0	1	0	0	1	1	1	1	1	
User 12	Try2	0	1	1	1	1	1	1	1	1	User 13
	Try2	0	1	1	0	1	1	1	1	1	
User 13	Try1	0	1	0	1	1	1	1	1	1	User 14
	Try2	0	0	1	0	1	1	1	1	1	
User 14	Try1	0	1	1	1	1	1	1	1	0	User15
	Try2	0	1	0	1	1	0	1	1	0	
User15	Try1	0	0	0	0	0	0	0	0	0	User1
	Try2	0	0	0	0	0	0	0	0	0	

C.2 Extended Key-paring Method

C.2.1 One-vs-One

ACO	
Actual	Predicted
1	6
1	1
1	1
1	6
1	1
2	2
2	2
2	2
2	2
2	2
3	3
3	3
3	3
3	3
3	3
4	4
4	4
4	4
4	4
4	4
5	10
5	5
5	14
5	10
5	14
6	1
6	6
6	6
6	6
6	6
8	5
8	5
8	5
8	5
8	8
9	9
9	9
9	9
9	8

MANOVA	
Actual	Predicted
1	1
1	1
1	1
1	1
1	1
2	12
2	12
2	2
2	2
2	13
3	3
3	1
3	3
3	8
3	3
4	4
4	4
4	4
4	4
4	4
5	4
5	5
5	14
5	4
5	4
6	6
6	6
6	6
6	6
6	6
8	8
8	8
8	13
8	8
8	8
9	9
9	9
9	9
9	9

9	9
12	12
12	12
12	12
12	12
12	12
13	13
13	9
13	20
13	13
13	20
14	14
14	14
14	18
14	14
14	18
15	15
15	15
15	15
15	15
15	15
7	4
7	13
7	13
7	7
7	7
11	18
11	11
11	10
11	11
11	11
10	5
10	7
10	10
10	10
10	10
16	25
16	25
16	25
16	25
16	25
17	18
17	17
17	17

9	9
12	12
12	4
12	10
12	12
12	12
13	13
13	9
13	13
13	13
13	13
14	4
14	14
14	14
14	14
14	24
15	15
15	15
15	15
15	15
15	15
7	3
7	7
7	1
7	14
7	7
11	11
11	11
11	11
11	11
11	11
10	13
10	13
10	11
10	11
10	11
16	16
16	14
16	16
16	16
16	16
17	17
17	17
17	17

17	17
17	17
18	23
18	18
18	18
18	18
18	18
19	22
19	18
19	18
19	18
19	18
20	20
20	20
20	20
20	20
20	20
21	21
21	21
21	21
21	21
21	21
22	22
22	22
22	22
22	22
22	22
23	23
23	23
23	23
23	23
23	23
24	24
24	24
24	24
24	22
24	22
25	25
25	25
25	25
25	25
25	25

17	17
17	17
18	18
18	18
18	17
18	17
18	17
19	18
19	24
19	24
19	14
19	24
20	18
20	21
20	21
20	22
20	18
21	25
21	21
21	14
21	16
21	13
22	16
22	19
22	21
22	22
22	14
23	16
23	17
23	17
23	23
23	18
24	24
24	24
24	24
24	24
24	24
25	25
25	25
25	16
25	24
25	25

C.2.2 One-vs-Rest

a. ACO

User 1		User 2		User 3		User 4		User 5		User 6		User 7		User 8	
Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

User 9		User 10		User 11		User 12		User 13		User 14		User 15		User 16	
Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1

-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

User 17		User 18		User 19		User 20		User 21		User 22		User 23		User 24		User 25	
Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	1	-1	1	-1	1	-1	1	-1	1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1		-1		-1	-1	1	1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1		-1		-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1		-1		-1	-1	1	-1

b. MANOVA

User 1		User 2		User 3		User 4		User 5		User 6		User 7		User 8	
Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1
-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

User 9		User 10		User 11		User 12		User 13		User 14		User 15		User 16	
Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1

-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

User 17		User 18		User 19		User 20		User 21		User 22		User 23		User 24		User 25	
Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted	Actual	Predicted
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1

c. Predictions

ACO	MANOVA
1	1
1	-1
1	1
1	-1
1	-1
-1	-1
-1	1
-1	1
-1	-1
-1	-1
-1	1
-1	1
1	1
1	1
-1	1
1	1
1	1
1	1
-1	1
-1	1
-1	-1
-1	-1
-1	-1
1	-1
-1	-1
-1	-1
1	-1
-1	1
1	1
1	-1
-1	-1
1	-1
1	1
-1	1
1	-1
1	-1
1	-1
1	-1
1	-1
1	1
-1	-1
-1	1

-1	1
-1	1
-1	1
1	-1
1	1
1	1
-1	-1
-1	1
-1	1
-1	1
-1	1
-1	1
-1	1
-1	1
-1	1
-1	1
-1	1
1	1
1	1
1	-1
1	-1
1	-1
1	1
-1	1
-1	1
-1	1
-1	-1
-1	-1
1	1
1	-1
-1	-1
-1	-1
-1	-1
1	-1
1	-1
-1	-1
-1	-1
1	-1
-1	-1
-1	-1
1	-1
-1	1
-1	1
-1	-1

-1	-1
1	-1
1	-1
-1	-1
-1	-1
-1	-1
-1	-1
-1	-1
1	-1
-1	-1
-1	-1
1	-1
1	1
1	1
1	-1
1	-1
-1	1
-1	-1
1	-1
1	-1
-1	-1
-1	-1
-1	-1
-1	-1
1	-1
-1	-1
1	-1
1	-1
1	-1
1	-1
1	-1
-1	-1
-1	-1
1	-1
1	-1
-1	-1
-1	-1
-1	-1
-1	-1

C.3 Non-conventional Method

C.3.1 DTs

	Predicted class							
Actual class	**Sample 1**	**Sample 2**	**Sample 3**	**Sample 4**	**Sample 5**	**Sample 6**	**Sample 7**	**Sample 8**
1	1	1	1	1	17	24	20	1
2	2	2	2	2	2	2	11	16
3	16	3	16	3	3	3	3	22
4	4	16	4	4	4	4	3	4
5	5	5	5	5	5	5	5	8
6	6	6	6	6	6	6	6	4
7	19	7	7	7	7	7	19	11
8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9
10	10	10	10	10	10	10	10	10
11	11	2	11	11	11	11	11	2
12	12	12	12	12	12	12	12	12
13	13	13	13	13	13	13	13	13
14	14	14	14	14	9	14	14	14
15	15	12	15	15	15	15	15	15
16	3	5	16	16	14	3	3	22
17	17	20	17	17	20	17	19	20
18	20	19	7	18	1	20	20	18
19	18	7	17	19	17	17	18	18
20	20	5	17	17	20	17	20	7
21	21	21	21	21	21	21	21	21
22	23	22	22	22	22	22	1	4
23	23	23	23	23	23	23	23	23
24	24	24	24	24	24	24	24	24
25	25	25	25	25	25	25	25	25

C.3.2 SVMs

	Predicted class							
Actual class	**Sample 1**	**Sample 2**	**Sample 3**	**Sample 4**	**Sample 5**	**Sample 6**	**Sample 7**	**Sample 8**
1	1	1	1	1	1	1	1	1
2	11	11	10	11	11	24	1	11
3	3	3	16	4	3	16	3	4
4	4	14	3	4	14	8	14	4
5	1	5	5	25	1	25	25	5
6	6	6	6	6	6	6	6	9
7	7	7	7	7	7	7	7	7
8	8	8	8	8	23	8	14	8
9	14	9	9	9	9	9	9	9
10	13	25	25	13	13	24	25	13
11	11	17	11	11	11	11	17	11
12	7	20	19	25	20	25	1	18
13	24	13	13	13	13	13	13	13
14	14	14	14	14	14	14	14	14
15	1	20	15	15	15	15	15	15
16	16	16	16	16	16	16	16	3
17	17	20	19	6	20	1	1	17
18	15	20	15	25	1	25	15	15
19	7	20	19	25	17	25	15	15
20	7	7	19	25	17	25	7	7
21	21	9	21	24	10	21	21	21
22	22	22	8	8	22	8	22	22
23	23	23	23	23	23	23	23	23
24	24	24	24	24	24	24	24	24
25	15	20	12	25	20	25	25	18

C.4 Fusion

C.4.1 Feature-level Fusion

C.4.1.1 DTs

	Predicted class							
Actual class	**Sample 1**	**Sample 2**	**Sample 3**	**Sample 4**	**Sample 5**	**Sample 6**	**Sample 7**	**Sample 8**
1	17	1	1	5	17	5	17	5
2	2	21	2	2	2	2	2	2
3	3	14	16	7	3	3	3	3
4	4	4	4	4	4	23	4	23
5	5	5	16	5	16	5	5	8
6	3	6	6	4	6	6	6	22
7	7	7	7	23	7	7	7	24
8	8	8	8	8	8	8	7	8
9	21	9	9	9	9	9	9	9
10	10	10	10	10	10	10	12	10
11	11	11	11	22	11	11	11	11
12	12	12	12	12	12	12	12	10
13	13	13	13	13	13	13	13	13
14	3	14	14	14	9	14	14	23
15	15	12	15	15	15	15	15	10
16	16	16	5	16	14	16	16	4
17	17	17	18	17	17	17	17	17
18	18	18	18	12	1	17	18	5
19	7	19	19	19	19	19	19	19
20	20	20	20	20	20	20	20	20
21	21	21	2	21	21	21	2	21
22	22	22	22	22	22	22	22	22
23	14	18	23	23	23	23	23	23
24	24	24	24	24	24	24	24	24
25	25	25	25	25	25	25	25	25

C.4.1.2 SVMs

	Predicted class							
Actual class	**Sample 1**	**Sample 2**	**Sample 3**	**Sample 4**	**Sample 5**	**Sample 6**	**Sample 7**	**Sample 8**
1	20	1	17	1	1	1	1	1
2	2	2	2	11	2	2	17	11
3	3	3	16	3	3	3	3	3
4	23	14	3	4	4	8	4	14
5	5	5	10	5	5	10	5	13
6	6	6	6	12	6	6	6	9
7	7	7	7	7	7	7	7	7
8	8	8	8	8	23	8	14	8
9	14	9	9	9	9	9	9	9
10	13	15	10	10	10	24	5	13
11	11	11	11	11	11	11	17	11
12	7	15	10	15	18	20	19	19
13	24	13	13	13	21	13	13	13
14	14	14	14	14	14	14	14	14
15	20	15	10	15	18	20	19	19
16	16	16	16	16	16	16	16	3
17	11	15	10	6	18	6	18	6
18	18	18	10	15	17	20	19	15
19	7	15	10	15	17	19	18	19
20	7	7	20	20	20	19	7	7
21	21	21	21	24	21	9	21	21
22	22	22	22	22	22	8	22	22
23	23	23	23	23	23	23	23	23
24	24	24	24	24	24	24	24	24
25	20	15	10	25	25	25	19	19

C.4.2 Decision-level Fusion

C.4.2.1 Approach (A)

	Predicted class							
Actual class	**Sample 1**	**Sample 2**	**Sample 3**	**Sample 4**	**Sample 5**	**Sample 6**	**Sample 7**	**Sample 8**
1	6	24	19	25	13	22	3	1
2	13	19	9	13	14	25	5	16
3	10	19	9	9	1	25	6	15
4	9	9	21	12	6	7	8	16
5	17	12	14	5	5	17	25	15
6	10	19	2	14	9	10	13	14
7	8	20	17	5	8	23	25	22
8	6	8	8	8	25	8	8	8
9	9	9	9	8	16	16	9	9
10	17	3	2	18	24	25	3	16
11	5	12	23	5	10	22	8	9
12	17	15	1	19	18	24	23	22
13	13	3	3	10	19	16	13	1
14	15	20	25	8	19	14	2	16
15	10	2	10	12	18	12	13	7
16	16	16	16	16	16	16	16	16
17	17	23	23	22	19	12	22	12
18	17	18	12	1	19	23	17	22
19	19	22	19	2	20	19	15	4
20	14	20	22	2	20	6	21	25
21	21	21	21	21	21	21	21	21
22	12	22	20	22	22	20	22	13
23	2	23	23	23	23	20	23	23
24	24	18	23	25	22	23	12	22
25	20	25	1	20	11	7	16	15

C.4.2.2 Approach (B)

	Predicted class							
Actual class	**Sample 1**	**Sample 2**	**Sample 3**	**Sample 4**	**Sample 5**	**Sample 6**	**Sample 7**	**Sample 8**
1	1	1	1	1	17	24	20	1
2	2	2	2	2	2	2	11	16
3	16	3	16	3	3	3	3	22
4	4	16	4	4	4	4	3	4
5	5	5	5	5	5	5	5	8
6	6	6	6	6	6	6	6	4
7	19	7	7	7	7	7	19	11
8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9
10	10	10	10	10	10	10	10	10
11	11	2	11	11	11	11	11	2
12	12	12	12	12	12	12	12	12
13	13	13	13	13	13	13	13	13
14	14	14	14	14	9	14	14	14
15	15	12	15	15	15	15	15	15
16	3	5	16	16	14	3	3	22
17	17	20	17	17	20	17	19	20
18	23	19	7	18	1	20	20	18
19	18	7	17	19	17	22	18	20
20	20	5	17	17	20	17	20	7
21	21	21	21	21	21	21	21	21
22	23	22	22	22	22	22	1	4
23	23	23	23	23	23	23	23	23
24	24	24	24	24	24	24	24	24
25	25	25	25	25	25	25	25	25

C.4.2.3 Votes

Actual class	Voted class
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25

C.5 Arabic Language Experiment

C.5.1 Arabic Input

C.5.1.1 SVMs

	Predicted class							
Actual class	**Sample 1**	**Sample 2**	**Sample 3**	**Sample 4**	**Sample 5**	**Sample 6**	**Sample 7**	**Sample 8**
1	1	1	1	1	1	1	1	1
2	2	2	3	2	2	2	2	2
3	3	14	11	2	21	3	6	3
4	3	14	11	2	21	3	6	3
5	9	5	5	9	7	11	6	12
6	21	5	21	3	6	14	13	6
7	2	9	8	7	7	15	7	7
8	9	8	9	9	12	8	11	8
9	9	9	7	9	9	13	15	9
10	10	10	10	16	10	10	17	10
11	3	9	11	7	15	11	11	5
12	12	8	18	1	11	6	12	8
13	13	13	13	13	16	11	13	13
14	2	14	18	14	14	14	6	14
15	15	15	7	9	15	7	15	7
16	20	16	16	16	16	16	16	16
17	17	17	17	17	17	17	17	17
18	18	18	18	18	18	18	18	18
19	19	19	19	19	10	16	19	19
20	20	20	20	20	20	20	20	16
21	21	21	21	21	21	21	21	21

C.5.1.2 DTs

	Predicted class							
Actual class	**Sample 1**	**Sample 2**	**Sample 3**	**Sample 4**	**Sample 5**	**Sample 6**	**Sample 7**	**Sample 8**
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	5	21	14	17	1	9	5	5
4	5	21	14	17	1	9	5	5
5	5	5	5	6	3	8	5	5
6	1	6	6	6	1	1	13	3
7	15	10	2	7	7	7	16	2
8	8	8	21	5	8	14	13	14
9	9	9	9	9	9	9	9	9
10	17	17	17	21	10	10	17	10
11	21	11	11	1	12	11	15	1
12	5	18	18	18	14	8	6	5
13	13	13	13	13	20	13	13	2
14	15	14	14	16	19	14	3	14
15	9	9	9	7	18	15	15	15
16	19	16	16	16	19	16	16	16
17	17	17	17	17	17	17	17	17
18	18	18	18	18	18	12	18	18
19	19	19	19	3	19	19	9	20
20	3	20	13	21	19	16	19	13
21	21	21	21	21	21	8	9	6

C.5.2 English Input

C.5.2.1 SVMs

	Predicted class							
Actual class	**Sample 1**	**Sample 2**	**Sample 3**	**Sample 4**	**Sample 5**	**Sample 6**	**Sample 7**	**Sample 8**
1	1	15	1	6	8	4	1	1
2	12	20	2	9	15	8	8	12
3	4	10	3	3	3	1	10	3
4	4	4	4	12	15	8	2	6
5	1	3	7	4	4	1	9	2
6	12	4	6	16	2	17	3	20
7	3	9	1	7	6	3	10	7
8	12	17	17	8	4	8	15	10
9	15	9	9	9	14	5	2	18
10	4	8	11	8	13	3	10	10
11	1	19	19	2	3	3	12	1
12	15	15	12	12	9	7	6	2
13	12	13	13	13	13	13	13	13
14	6	11	11	6	11	14	20	20
15	15	16	16	8	2	17	1	10
16	16	16	16	16	16	16	13	16
17	17	17	17	17	17	17	17	17
18	18	18	18	13	18	18	13	18
19	19	7	19	19	19	19	19	19
20	19	20	20	20	20	20	15	20
21	21	21	21	21	21	21	9	21

C.5.2.2 DTS

	Predicted class							
Actual class	**Sample 1**	**Sample 2**	**Sample 3**	**Sample 4**	**Sample 5**	**Sample 6**	**Sample 7**	**Sample 8**
1	7	2	11	1	3	12	10	2
2	2	8	14	4	2	8	10	6
3	6	9	11	17	13	2	5	3
4	15	4	4	14	11	4	12	3
5	5	7	8	2	4	19	5	13
6	20	21	20	9	8	4	16	21
7	7	9	13	7	21	18	17	9
8	6	20	8	2	4	3	12	20
9	6	8	7	18	14	14	19	4
10	3	5	2	3	10	10	10	13
11	19	10	2	2	21	16	14	13
12	5	8	7	2	16	12	5	19
13	11	13	11	11	16	18	16	21
14	20	20	9	21	15	14	9	9
15	20	20	2	1	8	5	21	14
16	17	19	16	18	16	16	18	16
17	17	17	17	7	17	17	18	17
18	18	18	18	18	18	18	18	18
19	19	19	19	19	19	19	7	19
20	20	20	20	20	20	20	20	20
21	21	21	21	21	21	21	21	21

Appendix D

Published Material

The following papers have been published based on the contents of this Thesis:

- A. Alsultan and K. Warwick, 'User-friendly free-text keystroke dynamics authentication for practical applications', *Proc. IEEE International Conference on Systems, Man, and Cybernetics (SMC2013)*, Manchester, October, 2013, pp. 4658 -4663.
- A. Alsultan and K. Warwick, 'Keystroke dynamics authentication: a survey of free-text methods', *International Journal of Computer Science Issues*, vol. 10, no. 4, 2013, pp. 1-10.
- A. Alsultan, K. Warwick and H. Wei, 'Free-text Keystroke Dynamics Authentication for Arabic Language', *IET Biometrics*, 2016.
- A. Alsultan, H. Wei and K. Warwick, "Feature Selection: A Comparison Study in Free-text Keystroke Dynamics Authentication," *Proc. International Conference Human Computer Interaction and Artificial Intelligence*, Manchester, 2016.
- A. Alsultan, K. Warwick and H. Wei, "Non-Conventional Keystroke Dynamics for User Authentication," *Pattern Recognition Letters*, Accepted to Appear in 2017.

The following poster has been presented based on the contents of this Thesis:

- A. Alsultan, K. Warwick and H. Wei, 'The Use of Arabic Language in Free-text Keystroke Dynamics Authentication", in the 9th Saudi Students Conference in the UK, Birmingham, 2016.

The full versions of the papers/posters are provided in this appendix.

2013 IEEE International Conference on Systems, Man, and Cybernetics

User-Friendly Free-text Keystroke Dynamics Authentication for Practical Applications

Arwa Alsultan and Kevin Warwick
School of Systems Engineering
University of Reading
Reading, United Kingdom

Abstract—**This paper introduces a novel approach for free-text keystroke dynamics authentication which incorporates the use of the keyboard's key-layout. The method extracts timing features from specific key-pairs. The Euclidean distance is then utilized to find the level of similarity between a user's profile data and his/her test data. The results obtained from this method are reasonable for free-text authentication while maintaining the maximum level of user relaxation. Moreover, it has been proven in this study that flight time yields better authentication results when compared with dwell time. In particular, the results were obtained with only one training sample for the purpose of practicality and ease of real life application.**

Keywords-Keystroke Dynamics; Free-text; Authentication; Keyboard Key-layout; key-pair; Euclidian Distance

I. INTRODUCTION

During the past couple of decades, online services have become an essential tool for performing many tasks on a daily basis. Such services usually involve using a username and password in order to verify users. Unfortunately passwords are prone to social engineering and can be easy to crack by using various methods, such as the dictionary attack and even a brute force attack. Therefore, users are obliged to use extreme measures to safeguard their passwords, a procedure which includes remembering long and complex passwords in addition to the need for changing their passwords periodically. This causes frustration and apprehension for users, especially when a single user is most likely responsible for more than a hand-full of ID/passwords spread over multiple systems.

In consequence, the need for a more useable method for authentication has become a necessity. One of the alternative methods for ID/passwords is behavioral biometrics such as signature recognition, handwriting recognition, gait analysis etc.; all of which depend on the user's behavioral patterns, which makes it intuitive for the user and difficult to imitate by others. Unfortunately, all of these previously mentioned methods need highly reliable external hardware for measurement purposes, which is considered a critical drawback from the point of view of users, who are mostly concerned with the practicality of the system, in addition to the undesirable monetary costs of these devices.

Monitoring keystroke dynamics, on the other hand, is considered to be an effortless behavior-based method for authenticating users which employs the person's typing patterns for validating his/her identity. As mentioned in [1], keystroke dynamics is "not what you type, but how you type." In this approach, the user types in text, as usual, without any kind of extra work to be done for authentication. Moreover, it only involves the user's own keyboard and no other external hardware.

This paper considers a keyboard-layout based method to compare timing features of free text typing samples. The method classifies the text to five different key-pairs depending on the position of the two keys on the keyboard. The Euclidian distance between the timing features' vectors of each key-pair is used to find the level of similarity among the samples.

This particular structured key-pair based method for extracting features was followed in this study in order to increase the number of the di-graphs found and compared in both the training and the testing samples. The aim was to enhance the stability of its mean and standard deviation which is the main component of the user's timing vector. Furthermore, this method would aid in reducing the required computation time needed for comparing samples.

In this approach, we attempt to use the least amount of training samples possible. The main goal for this is the user's comfort and relaxation and hence the realization of a practically employable system. It is not adequate to relieve users from remembering long passwords if they will still have to type-in huge amount of text, multiple times when enrolling in the system. Hence here enrollment is a simple, relatively rapid process. Indeed this has been regarded as a vitally important aspect of our research.

The rest of this paper proceeds as follows. Section II introduces the Keystroke dynamics theory and describes some of the work previously carried out which is utilized in user authentication. Section III describes the method we have actually followed in this study. At first, we indicate the data collection technique followed. Then, we discuss the timing features we decided to include in the study. Next, we explain the concept of key-pairs and how it was applied in our experiment. After that, we point to the timing vector's creation and the distance calculation employed. In Section IV we present experimental results and make some comparisons with other recent studies in the same area. The final section concludes the topic and points out our research contributions and future work.

II. KEYSTROKE DYNAMICS

There are two basic classes of keystroke dynamics, namely: fixed-text and free-text. The fixed-text keystroke dynamics method is computed using a predefined text that the system has previously trained on and has to be delivered by the user at log-in time. On the other hand, the free-text keystroke method is considered easier for the user since it does not require memorizing any text due to the fact that the

978-1-4799-0652-9/13 $31.00 © 2013 IEEE
DOI 10.1109/SMC.2013.793

text used for enrolment does not have to be the same as the text used for log-in. In addition, free-text keystroke dynamics is used for enhancing security through continuous and nonintrusive authentication [2]. It is this latter procedure which has been considered in this paper.

Although free-text keystroke systems are much more complex than fixed-text systems, they can be applied in many useful settings to aid in real life situations in addition to the benefit they provide in balancing between security and usability [16] which directly fits with the objectives of this paper.

Basically, Keystroke dynamics are utilized in user's authentication by extracting timing features at the log-in session and comparing them with the timing features extracted at the enrolment session. These features include, among others: typing latency, keystroke duration [3], typing speed and shift key usage patterns [4]. If the extracted features are adequately similar, the user is authenticated and if not, the user might be denied access or at least asked to provide further identity information.

Undeniably, the timing features extracted from the two sessions will never be exactly the same since human behavior is often subject to change. Therefore, a threshold is regularly used to determine if the level of similarity actually attained in the authentication process is deemed to be satisfactory.

A lot of research has been carried out over the years to investigate the exploitation of keystroke dynamics for authentication purposes. Joyce and Gupta [5] used a statistical method that employs the absolute distances between the means of the signature data and test data; each of which consists of a fixed-text that includes username, password, first name, and last name.

Meanwhile Gunetti and Picardi [6] introduced an effective method for free-text authentication which was further explored by many other researchers. Their method was based on two measures: relative (R) measure and absolute (A) measure. These measures were used to calculate the degree of disorder and the absolute distance between two samples that share some n-graphs.

Another technique was introduced by Singh and Arya [7] for free-text authentication, in which, a keyboard grouping technique was used for creating timing vectors of flight times entered by the user at the enrolment session and log-in session. The keys were grouped based on their position on the keyboard, which was divided into 8 sections; two left and right halves and then each half divided into 4 lines representing the rows of the keyboard. The Euclidean distance was then used to calculate the similarity between the two vectors.

Park et al. [15] also considered benefiting from key pairs in keystroke dynamics. They first started by dividing all keystrokes into four features; left hand side, right hand side, spacebar and backspace bar. Then, they created di-graphs using combinations of these features. This resulted in sixteen di-graphs of features. After that, they compared two samples using these digraphs. Only diagraphs with more than ten appearances in the training and testing samples were used in the comparison. The last step was using the Kolmogorov-Smirnov test in order to decide if the test data actually belonged to the legitimate user.

Other researchers relied on pattern recognition classifying methods such as K-nearest neighbor; one example of which was the work done by Hu et al. [8]. They used the k-nearest neighbor approach together with the distance measurement proposed by Gunetti and Picardi [6] in order to classify the users' keystroke dynamics profiles.

Neural Networks have also been used for keystroke pattern classification; such as the research conducted by Raghu et al. [9] where they incorporated a three-layered back propagation neural network to verify the identity of users. It should be said that this method, along with those previously mentioned, involved substantial input from the user in order to allow the computer system the chance to learn the user's keystroke characteristics. Whilst such experimentation can be acceptable in the laboratory, this extensive initial input can in reality be very off putting, in a practical system, for the user.

III. METHODOLOGY

A. Data Collection

In our study, fifteen users participated in the experimentation. Each participant provided merely one training sample. There were nine females and six males. All the participants were in the age group between 20 and 60 years old. They had different levels of typing skills that varied between moderate and very good. Not all the people participating in the experiment were native speakers of English. Indeed this was a particular characteristic of our study. Included were, for example, native Pakistani, Czech and Saudi speakers.

During the data collection phase, or what we call here the enrolment phase, we asked the participants to type-in a paragraph consisting of five lines of text. The text included five short and well-known English quotes. Most of the words used in this text were short, simple and known words reliant on the study done by Giot et al. [10] which proved that these kinds of words give better results in keystroke dynamics systems. All the words were in lower case and we did not include any numbers or punctuation marks.

This 380 character long text was the only sample used for training the system, which was considered a very short training sample compared to previous studies. This added to the user-friendliness aspect of the system since the only instance of typing required for training the system was of reasonable length; which spared the user the annoyance of spending a long time in the enrolment phase.

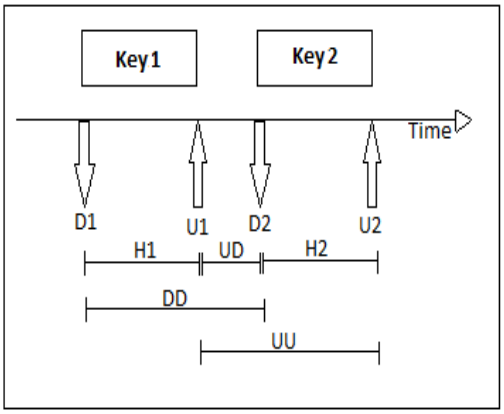

Figure 1. Keystroke features.

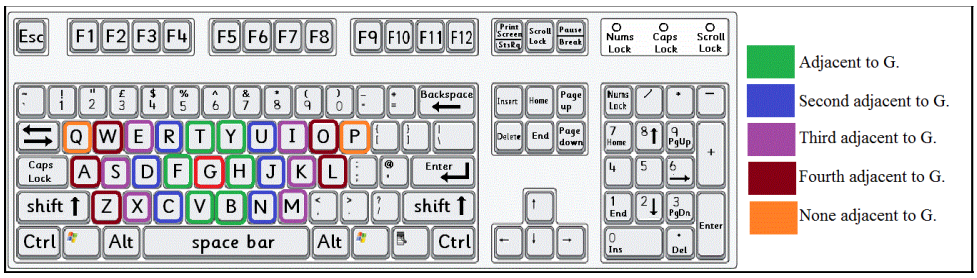

Figure 2. Key-pair classification example.

Users were directed to enter the samples in the most natural way they usually follow when typing. The user is allowed to enter carriage returns and backspaces if needed. The data collection was performed on a GUI program implemented using Java language.

B. Feature Extraction

After obtaining the users' raw data, extracting the keystroke features was performed [11]. These features were computed for every key and key-pair using two main values, specifically: the press time (D_n) and the release time (U_n) of each key (n) in milliseconds. In this research, five keystroke features were extracted from each key-pair as shown in Figure1:

1) Dwell time or Hold time: is the time a key is pressed until it is released. Consequently, each key-pair has two hold times:

a) Hold time for the first key (H1).

b) Hold time for the second key (H2).

2) Flight Time or Keystroke latencies: There are three types of latencies:

a) Down-Down (DD) or Press-Press (PP) time: is the interval time between two successive key presses.

b) Up-UP (UU) or Release-Release (RR) time: is the interval time between two successive key releases.

c) *Up-Down (UD) or Release-Press (RP) time:* is the interval time between a key release and the next key press. Unlike the other two keystroke latencies, Up-Down can be a negative value in the case that the next key is pressed before releasing the previous one, which can happen in the case of very fast typing speeds.

C. Key-pair Classification

In this research we introduce a novel free-text based approach which makes use of the keyboard's key-layout. This approach utilizes the keystroke features extracted between two keys (key-pair) which are pressed consecutively and have a relationship on the keyboard layout. This relationship depends mainly on the key positions on the keyboard. The keyboard used in this study was the English QWERTY keyboard since it is both the most common keyboard layout and the most popular one.

There are five types of key-pairs:

1) *Adjacent:* keys located next to each other on the keyboard.

2) *Second adjacent:* keys that are one key apart from each other.

3) *Third adjacent:* keys that are two keys apart.

4) *Fourth adjacent:* keys that are three keys apart.

5) *None adjacent:* keys that are more than three keys apart.

For more of an explanation, an example is illustrated in Figure 2. If we consider the key G, the related key-pairs are:

- F, T, Y, H, B, and V are adjacent keys to G.
- U, J, N, C, D, and R are second adjacent to G.
- I, K, M, E, S, and X are third adjacent to G.
- O, L, Z, A, and W are fourth adjacent to G.
- P and Q are none adjacent to G.

This key classification can be performed in the same way for all the key-pairs in the typed text.

An empirical example is given for all the key-pairs in the word "university" in table I. This example shows the process followed to break down a text into di-graphs and then classifying each digraph into one of the five types of key-pairs before using its timing data in the corresponding timing vector.

The key-pair based method was adapted as a way to escalate the stability of the timing vectors giving that it significantly boosts the number of key-pairs that can be found and compared in the training and testing samples. This is an obvious benefit of the suggested scheme as it utilizes a small amount of typing data in the best possible way. For example, the following training and testing data have only two similar di-graphs ("in" and "ng") whose typing times' can be compared in the authentication process.

Training data: "University of reading."

Testing data: "Systems engineering."

This is not the case when using the key-pair method as there are a larger number of instants for each kind of key-pair extracted from both the training and testing data; as shown in table II.

TABLE I. KEY-PAIRS IN THE WORD "UNIVERSITY"

Di-graphs	**un**	**ni**	**iv**	**ve**	**er**	**rs**	**si**	**it**	**ty**
Key-pair type	2nd adjacent	2nd adjacent	4th adjacent	3rd adjacent	Adjacent	2nd adjacent	Non Adjacent	3rd adjacent	Adjacent

TABLE II. TOTAL NUMBER OF KEY-PAIRS.

Key-pair type	**Adjacent**	**2nd adjacent**	**3rd adjacent**	**4th adjacent**	**Non adjacent**
Training data	3	7	2	1	3
Testing data	1	5	2	3	4

Each of the key-pairs is subject to the five features extraction described earlier. Pairs that involve spaces are discarded because of the work conducted by Singh and Arya [7] which inferred that a user normally experiences very unusual pauses before and after the space key which causes inconsistent typing behavior.

D. Capturing Timing Vectors

For each key-pair appearance in the text, the five timing features are extracted. Then, the mean of each timing feature is calculated for all the key-pair appearances. This is done for all the five types of key-pairs. From observing users behavior, we noticed that some users take small pauses for different reasons, such as moving their eyes to read the provided text, therefore any outlier data are discarded. We identify outlier data to be as much as three standard deviations above the mean [5].

After that, the timing vector is created and stored in the database. This timing vector consists of the means of the five features for all five kinds of key-pairs.

E. Finding Distance

At the log-in, or authentication, phase, the user is asked to enter another line of text that is different from the text entered at the enrolment phase. This text is approximately 75 characters long. Then, the system prepares the timing vector for the test data which includes the means of the five features extracted from the five types of key-pairs. Figure 3 (A) shows the similarity between the two timing vectors for one user whilst (B) shows the difference between the two vectors for two different users.

After that, the Euclidean distance is calculated between the log-in vector and the vector stored at the user's profile in the database. This distance has to be sufficiently small to give a good judgment that this is the legitimate user. Nevertheless, the distance will never be equal to zero because human behavioral characteristics are not always consistent [7]. Therefore, an acceptable value for the distance is determined; which if it is not exceeded, the user is accepted as genuine; and if otherwise, the user is denied access. This value or threshold, is decided based on each user's profile data [5]. After several trials, the threshold that we decided to use is described as the mean plus one and a half standard deviations. The threshold used was a local threshold i.e. it was calculated for each user based only on the training data which was used to build the timing victor in that user's profile.

IV. RESULTS AND DISCUSSION

Each participant was asked to enroll once in the system first; in which his/her timing profile was calculated and stored at the Database. This is the only sample that the user provides to the system as a training sample. And then each participant was requested to provide two testing samples. The testing process has been conducted in two manners:

1) Legitimate user testing: where each user timing profile is compared with their own two sets of testing data.

2) Imposter user testing: where each user timing profile is compared with two other random participants' testing data.

To infer the performance of the authentication method, two error rates are used [2]:

1) False Accept Rate (FAR): is the percentage of impostors who have successfully gained access to the system.

2) False Reject Rate (FRR): is the percentage of legitimate users who have been denied access to the system.

The results from using the five timing features were analyzed separately and results from using a combination of these features were also investigated. After examining the results, most of the features produced low FRR, but on the other hand, the FAR was not as satisfactory. This trade-off between FAR and FRR is unavoidable [16]. Since user relaxation is of higher priority, low FRR is more important in this study. This denotes that genuine users will face less access rejections which will relieve them from unfavorable stress while using the system, which meets the main objective of this study.

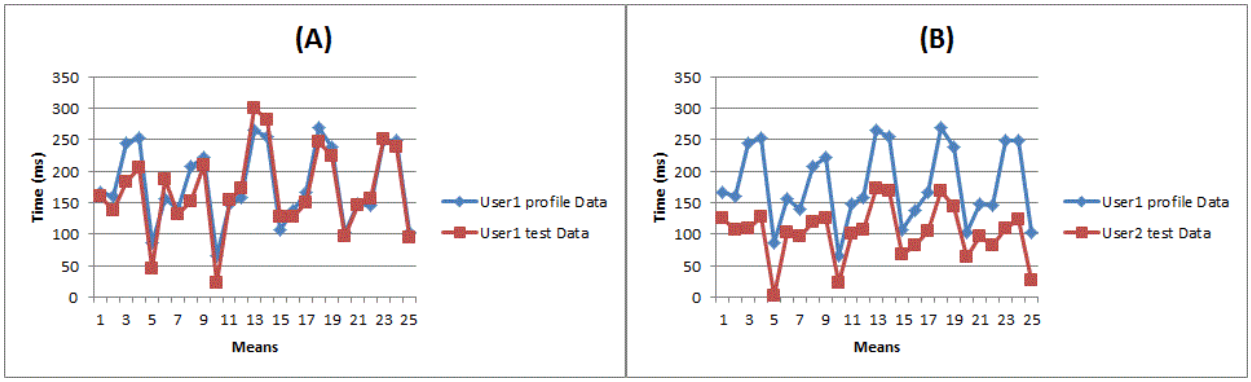

Figure 3. (A) The same user's timing vectors (B) Diferent users' timing vectors.

Moreover, it was found, by using an exhaustive search, that the Up-Down (UD) timing feature produced the best results in comparison to other individual features and it also gives a good balance between FAR and FFR. Furthermore, even better results were delivered when using the UD in combination with other features. For the sake of brevity, Table III only provides the error rates for the combination of timing features that includes the UD time.

The results from this research thus far are clearly comparable to other free-text studies such as the work done by Hempstalk et al. [12] which resulted in a 11.3% FAR and a 20.4% FRR and the study conducted by Xi et al. [13] which delivered a 9.43% FAR and a 24.7% FRR. However, both studies incorporated a relatively huge number of enrolment samples which adds up to 3000 samples from 19 participants and 765 samples from 205 participants respectively. This can be directly compared to one sample per participant for enrolment in our study, i.e. 15 samples from 15 participants.

TABLE III. ERROR RATES

Features	FAR	FRR
UD	25%	28%
H1 + UD + H2	28%	14%
DD + UD + UU	21%	17%
H1 + DD + UD + UU + H2	25%	25%

It is interesting in directly comparing our own results with those from previous free-text studies that our own FRR rates are pretty similar, indeed in some cases even better, despite the requirement for far fewer enrolment samples, however FAR rates are clearly much better in the other studies.

Whilst increased numbers of enrolment samples is clearly one reason that strongly affected the other results positively for FAR statistics, on the other hand, in this study user comfort was considered a main goal which accordingly limited the samples that each user was asked to provide. This is due to the fact that users are less likely to tolerate systems that require a large burden by having to go through long and multiple training sessions. Our own method is therefore, we believe, much more realistic from a practical perspective.

Another important aspect that adds to the practicality of this system is the text length used for both the training and testing. Even though, the shortness of the text used for training is highly desired by users, it is an evident reason for the substandard FAR. Increasing the sample size for both phases of this experiment will definitely intensify the performance of the system as noted by Curtin et al. in [14].

Nonetheless, an inevitable trade-off had to be experienced in this study as the user-friendliness of the system was of the highest priority. Hence, smaller numbers of training and testing samples were requested to be delivered by the users in both the enrolment and the authentication phases. This trade-off was reduced considerably by using key-pairs which work to enlarge the number of di-graphs that can be used for comparing samples during the course of authenticating users.

V. CONCLUSIONS

This paper examines the usefulness of using an original keyboard key-layout based method for keystroke dynamics authentication. The experiment produced reasonable results considering the fact that it uses free-text for authentication which gives a good balance between the system's security and the user's comfort. The UP-Down (UD) latency proved to be the best feature to use in such a system. Better performance was obtained using a combination of features that includes the UD time.

Looking at the results produced, it was not a surprise that the authentication performance is not perfect due to the nature of the experiment and the user's comfort priority. Nevertheless, using only one short training sample resulted in quite similar FRR results to other studies which require much more training samples than ours. Therefore, we have succeeded to create a simple yet practical system for authenticating users with the lowest possible amount of irritation for users.

There are much more that can be done to improve this approach. One of which is using a more sophisticated key-pair classification which takes into consideration the position of the hand on the keyboard. Another addition that might aid in increasing the performance of this method is selecting only key-pairs that have a minimum number of instants appearing in the training and testing data, which will help to further improve the stability of the timing vectors. Moreover, to produce more accurate results, data from more participants will be incorporated in the experiment. This is clearly ongoing research in which results thus far are extremely encouraging.

ACKNOWLEDGEMENTS

The authors wish to extend their gratitude to the participants who were involved in this experiment for the time they took out of their busy schedules to contribute in this study.

REFERENCES

[1] F. Monrose and A. Rubin, "Authentication via keystroke dynamics," in the 4th ACM conference on Computer and communications security, New York, USA, 1997, pp. 48 - 56.

[2] S. P. Banerjee and D. L. Woodard, "Biometric authentication and identification using keystroke dynamics: a survey," Journal of Pattern Recognition Research, vol. 7, no. 1, pp. 116-139, 2012.

[3] M. Karnan, M. Akila and N. Krishnaraj, "Biometric personal authentication using keystroke dynamics: a review," Applied Soft Computing, vol. 11, no. 2, pp. 1565-1573, 2011.

[4] E. Lau, X. Liu, C. Xiao and X. Yu, "Enhanced user authentication through keystroke biometrics," in Computer and Network Security, Massachusetts Institute of Technology, 2004.

[5] R. Joyce and G. Gupta, "Identity authentication based on keystroke latencies," Communications of the ACM, vol. 33, no. 2, pp. 168-176 ,1990.

[6] D. Gunetti and C. Picardi, "Keystroke analysis of free text," ACM Transactions on Information and System Security, vol. 8, no. 3, pp. 312-347, 2005.

[7] S. Sing and K.V. Arya, "Key classification: a new approach in free text keystroke authentication system," in Third Pacific-Asia Conference on Circuits, Communications and System, China, 2011, pp. 1-5.

[8] J. Hu, D. Gingrich and A. Sentosa, "A k-nearest neighbor approach for user authentication through biometric keystroke dynamics," in IEEE International Conference on Communications, Beijing, 2008, pp. 1556–1560.

[9] D. Raghu, C. H. R. Jacob and Y. V. K. D. Bhavani, "Neural network based authentication and verification for web based keystroke dynamics," International Journal of Computer Science and Information Technologies, vol. 2, pp. 2765-2772, 2011.

[10] R. Giot, A. Ninassi, M. El-Abed and C. Rosenberger, "Analysis of the acquisition krocess for keystroke dynamics," in International Conference of the Biometrics Special Interest Group, Germany, 2012, pp. 1-6.

[11] P. S. Teh, A. B. J. Teoh, T. S. Ong and H. F. Neo, "Statistical fusion approach on keystroke dynamics," in Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, Shanghai, 2007, pp. 918 – 923.

[12] K. Hempstalk, E. Frank and I. H. Witten, "One-class classification by combining density and class probability estimation," in European Conference on Machine Learning and Knowledge Discovery in Databases, Berlin, 2008, pp. 505-519.

[13] K. Xi, Y. Tang and J. Hu, "Correlation keystroke verification scheme for user access control," Computer Journal, vol. 54, no. 10, pp. 1632-1644, 2011.

[14] M. Curtin , C. Tappert, M. Villani, G. Ngo, J. Simone, H. St. Fort, and S.-H. Cha, "Keystroke Biometric Recognition on Long-text Input: a Feasibility Study", in IMECS, 2006.

[15] S. Park, J. Park and S. Cho, "User Authentication Based on Keystroke Analysis of Long Free Texts with a Reduced Number of Features", in the Second International Conference on Communication Systems, Networks and Applications, 2010.

[16] A. Alsultan and K. Warwick,"Keystroke Dynamics Authentication: A Survey of Free-text Methods', International Journal of Computer Science Issues, Vol. 10, no. 4, 2013.

Keystroke Dynamics Authentication: A Survey of Free-text Methods

Arwa Alsultan[1] and Kevin Warwick[2]

[1]School of Systems Engineering, University of Reading
Reading, Berkshire, UK

[2]School of Systems Engineering, University of Reading
Reading, Berkshire, UK

Abstract

Current computer systems depend greatly on authentication methods in order to provide sufficient protection to the data handled by these systems. Rather than using the common username and password scheme which suffers from many security and usability limitations, we investigate in this paper the use of keystroke dynamics as a more useable authentication alternative. We focus on the research done on free-text keystroke systems and its ability to provide continual identity verification during the whole time that the user is using the system.

Keywords: *free-text, keystroke dynamics, authentication, identification, performance, survey.*

1. Introduction

The use of computer systems has proliferated at an unforeseen rate. They are now used in almost all aspects of our lives. This is a strong reason to protect them against illegal intrusions. However many computer systems use the simple username/password scheme for authentication, even though it suffers from the security-usability trade-off dilemma. Passwords can be guessed using different methods such as social engineering, spyware, dictionary attack and mere brute force attacks. These are all reasons for the user to employ extreme measures to safeguard his/her computer by using long and complex passwords which are unfriendly and hard to memorize. It is therefore ideal to use an alternative authentication method that can be low-cost yet provide ease of use and transparency to the user in addition to security robustness.

Keystroke dynamics is a behavior biometric scheme that provides sturdy system protection while maintaining a high level of usability. In particular, using free-text keystrokes provides real-time identity verification by continuously monitoring the keyboard's activities. This is a very important, yet frequently ignored, part of the authentication process since it is fairly simple to establish a level of confidence about the user's identity at log-in time. However there is no guarantee that the user who was successfully authenticated is the same person who is still using the system. There is always a chance that the system was left unattended which is a golden opportunity for the attacker who is physically close to the machine to have access to it and, for example, alter some documents or send an e-mail on behalf of the original user.

In this method of authentication, it is not obligatory to memorize any text such as a password or a passphrase; instead authentication is conducted through finding the resemblance of the typing rhythm of a user, in a non-intrusive manner, regardless of the text typed.

One important fact in looking at research to date in free-text keystroke systems is that results from most studies are far from ideal, i.e. either the resulted accuracy is not satisfactory or it has a high accuracy level which was obtained under strictly controlled conditions, which is not at all representative of real-life situations. Thus, we aim in this paper to look at the various factors that might affect the authentication system performance in addition to covering the methods used for feature extraction and classification. Situations where free-text keystroke dynamics are best used are also discussed in this paper.

The rest of this paper proceeds as follows. Section two introduces keystroke dynamics theory and describes the differences between fixed-text and free-text systems. The third section lists some of the techniques followed for feature extraction while the section after that lists the methods used for classification. Performance measurement schemes are considered next. After that we list some of the factors affecting performance in free-text systems. A variety of applications that can benefit from free-text systems is given in the seventh section. Finally we discuss the level of protection that free-text systems can provide against some of the common security threats.

2. Keystroke Dynamics

Monitoring keystroke dynamics is considered to be an effortless behavioral based method for authenticating users which employs the person's typing patterns for validating his/her identity. As mentioned in [1], keystroke dynamics is "not what you type, but how you type." In this approach, the user types in text, as usual, without any kind of extra work to be done for authentication. Moreover, it only involves the user's own keyboard and no other external hardware. The original idea of using keystroke patterns for user identification purposes was originated from the idea of identifying the sender of Morse code on a telegraph machine, where operators have been able to identify the sender of a message by the rhythm, pace and syncopation of the received taps [2].

As early as 1980, researchers such as Gaines et al. [3] started to show interest in proving the hypothesis that typing patterns can be used as a mean of user

authentication. Experiments were conducted to find typing patterns that can be used effectively for authentication. Results from these tests showed that the similarity between typing samples from the same person is high with respect to the time delays it takes the user when typing one key or two successive keys. All of this early research though was only concerned with keystrokes generated by typing fixed words.

It wasn't until 1995 when Shepherd et al. [4] showed interest in continuous authentication. In 1997, the first organized attempt to use free-text keystroke system was conducted by Monrose and Rubin [1] where both fixed-text and free-text were used. The overall performance was not encouraging for free-text giving only 23% correct classification while fixed-text produced about 90% correct classification. This shows the complexity of using free-text systems compared with the fixed-text systems. Nevertheless, free-text systems have gone a long way since that experiment and much better results have been obtained using more sophisticated techniques.

There are two main phases that a user has to go through in order to be authorized by keystroke dynamic systems; namely: the enrolment phase and the log-in phase. The first phase has to do with collecting data about the user such as username and password in addition to capturing the user's typing behavior. The system gathers the keystroke times and extracts the timing features to create a template for each user's typing behavior. This template, also referred to as a user's profile, is stored in a database in correspondence to the user's other details.

The second phase takes place whenever the user needs to actually use the system. At that time, the system collects the user's keystroke times and then extracts the timing features in the same manner pursued in the enrolment phase. After that, the system performs feature matching with the user's template which is stored in the database. Next, based on the results of the matching process, one of two actions will take place: granting access to the user if the two sets of data are sufficiently similar or denying access to the user otherwise.

Two types of keystroke systems are used and discussed in the literature; they are: fixed-text and free-text keystroke systems. Fixed-text, also referred to as static, obliges the users to use only a predefined text to produce the typing samples. The predefined text varies in the research done in this area in the way that some have utilized the same shared password for all users [5] and others used different fixed text for each user such as using the user's name [6] or log-in IDs [7]. The main function of the fixed-text systems is applying it at log-in time in order to verify the user's identity at the beginning of the session only. This is done by forcing the user to retype their password a number of times at the enrolment phase in order to determine the user's typing rhythm for that specific password. This is considered a critical usability issue because of the amount of burden it adds on the user; still, the user needs to memorize the predefined text. Generally speaking, fixed-text keystrokes are mainly used for password hardening.

Free-text systems, also known as dynamic, don't restrict users to a particular text; on the contrary, they are given complete freedom to use any text of any length without any constraints. Unlike fixed-text, free-text systems will continue to collect the keystrokes, after successfully passing the log-in session, throughout the whole time that the user is logged-in for the reason of assuring the identity of the user during the full duration of that session. In free-text systems, the user's typing pattern is typically monitored during several days where he/she is performing regular typing tasks such as writing e-mails or typing word documents i.e. the enrolment phase is long yet transparent to the user. Even though, free-text and fixed-text systems are quite similar in the way that they both utilize the key press and release times to build a user behavior profile, they clearly differ in the way that the system is trained and applied.

All keystroke dynamics studies involve conducting five main experiment parts in the following order: recruiting participants, requesting a typing task to be done by the participants, collecting the keystrokes timing data, obtaining timing features from the raw keystroke data, training the classifier using part of the keystroke data and using the other part for testing the classifier [8]. We will go through the previous mentioned stages in order to compare and contrast what has been done in this area as reported in the current literature.

3. Feature Extraction and Profile Creation

The manner in which user data is collected in free-text keystroke systems is quite different from that of fixed-text systems in the way that a user is normally monitored for a period of time, a number of days for example. From all the typing data collected during this time, the system infers the typing pattern that the user typically follows which will be then stored as the user profile. The time it takes to type single letters or combinations of letters i.e. di-graphs, tri-graphs, even longer combinations is considered in free-text keystroke systems, yet there is a condition for including a particular letter or combination of letters in the template. It has to be typed often enough during the enrolment phase which will cause its mean and standard deviation to be statistically sound [9].

This implies that it is not necessary to include all letters and letter combinations, typed during the enrolment phase, in the template. Therefore, much research includes a pre-processing stage for removing noise from the data set. Extreme duration or latency values, i.e. very small or very large outliers, are discarded; for example: only the durations and the latencies of keys for which the standard deviation was below a predefined value were added to the user's template in [10, 1] while minimum and maximum values were fixed for the latencies that were used in [11].

Timing features are basically calculated using the press and release times of every key the user types and then processed in a specific way before being stored in the user's profile. Different methods were followed to carry out this part of the system, as shown in Table1. Here we

focus on some of the common methods used for feature extraction and profile creation in free-text keystroke systems.

First we go through some of the simple feature extraction techniques found in the literature. Profiles in [1, 23] consisted of the mean latency and standard deviation of each di-graph in addition to the mean duration and standard deviation of each individual key. While the profiles in [11, 10] only included the latencies' means and standard deviations for di-graphs that have occurred a minimum number of times. On the other hand, the down-down duration time of di-graphs was used in [12, 13, 14]. This was extended in [15] to include more n-graphs including di-graphs, tri-graphs, and other longer n-graphs.

Although, di-graph and tri-graph time has been used in plenty of research, Sim and Janakiraman [16] concluded from their several experiments that using di-graphs/tri-graphs is not a good discriminative between users when the actual typed words are not taken into consideration. This is because the context of the text that a particular letter is included in regulates the manner in which it is typed [9] i.e. the letter 't' has different duration in the word 'sentence' and 'question'. Therefore, di-graphs/tri-graphs are more effective for keystroke dynamics when using context-specific n-graphs.

A more structured feature extraction was followed in some research where the timing features were extracted for only a set of key pairs which helped to increase the number of the di-graphs that can be found and compared in both the training and the testing samples. This increases the stability of its mean and standard deviation, in addition to reducing the required computation time. This was done in [17] by dividing all keystrokes into four attributes: left hand side keys, right hand side keys, spacebar and backspace bar; then, creating 16 diagraphs using these attribute combinations.

A keyboard grouping technique was introduced in [18] for classifying the keys based on their location on the keyboard, which was divided into 8 sections; two left and right halves and then each half divided into 4 lines representing the rows of the keyboard. For example WM is represented as Left 2- Right 4. Moreover, only a fixed set of letters and two letter combinations were used in [9, 19]; these sets were chosen based on each letters frequency in the English language. Letters including E, A, T ... etc. and di-graphs including: AT, TH, HE … etc. are frequently found in English text, therefore, it is a good idea to use the mean and standard deviation of their duration and latencies in the user template which will increase its stability.

More complex features were also taken into consideration for the purpose of distinguishing users typing behavior. In addition to the usual key-press duration and di-graph's down-down (duration) and up-down (latency) times, other features were utilized in [19, 20, 21]; such as: typing speed, error rate, press-release ordering and the percentages of using special characters. Other features that capture the editing patterns of the user which includes the usage of specific keys i.e. Home, End, Backspace, Delete, Insert, shortcut keys , arrow keys … etc. were also used.

Another interesting feature was used in [22]; where all the commands executed in the first 10 minutes were collected. Although, it might seem irrelevant on first glimpse, an attacker is more likely to hurry to execute as many commands as he can on the victim's machine during the first few minutes. This shows an obvious change in the users habits which can be used to detect illegal intrusion.

4. Methods

After extracting the users' typing features and creating their profile templates has been completed, the classification process is performed to find the similarities and differences between the user's template stored at the enrolment phase and the sample provided during the session the system is being used. Similar to fixed-text systems, many methods have been used for classification in free-text keystroke systems; ranging from simple statistical methods to more complex pattern recognition and neural network algorithms. Moreover, an even more sophisticated combination of methods was used in some cases. This section highlights the major classification approaches used in the current literature. Please refer to Table1 for more details.

Simple statistical methods were used as a classification mean for typing behavior in several free-text keystroke systems studies. A variety of distance techniques have been used; Euclidean distance [18], weighted Euclidean distance [23], scaled Manhattan distance [9] and Bhattacharyya distance [24] were all utilized to find the level of similarity between samples. In addition, other statistical techniques were also used; decision trees were used in [22] while Kolmogorov-Smirnov Test (KS-test) was used in [13, 17].

One of the most cited free-text studies was that conducted by Gunetti and Picardi (P&G) [15] which depended on two measures, the first of which was the relative measure which was used to find the degree of disorder between the two samples. The second was the absolute measure which was used to calculate the absolute distance between the two samples. In both the relative and absolute measures only n-graphs occurring in both typing samples were considered. Even though the results were very good, the computational costs required to identify users was expensive because it needed to compare the test sample with all users' templates in the database which obviously makes it less scalable. Hu et al. [25] attempted to solve the scalability issue of P&G's method using the k-nearest classifier. In this approach, training samples were divided into clusters such that, every test sample was compared only with the samples of those users in the same cluster. Results for this modification revealed accuracy which compared well with that of P&G. Computation speed, on the other hand, proved to be 66.7% better.

A number of extensions have been carried out on P&G's method by Davoudi and Kabir. In [12] they combined P&G's method with a distance-calculating method that used histogram-based density estimation for each di-graph in order to find the probability density function of the di-graph's duration time. While in [26], they modified the relative distance in the P&G method by choosing the di-graph with the highest difference in duration between the two samples to compute the difference of its positions first. After that, it was removed from the two timing vectors, and then, the new vectors were sorted again. They also applied one further modification to P&G's method in [14] by adding a weight factor to the digraphs when computing the relative distance. This weight was defined as the ratio of the number of occurrences of this digraph and its standard deviation.

Table 1: Chronological list of free-text keystroke systems.

Study	***Features***	***Method***	***Subjects***	***Samples***	***Performance***
Monrose & Rubin [1]	Di-graph latency, key duration	Euclidian distance, probability score, weighted di-graph probability	31	-	23% accuracy
Gunetti & Ruffo [22]	Di-graph latency, executed commands	Decision tree	10	-	90% accuracy
Dowland et al. [11]	Di-graph latency	Mean, Standard deviation	4	-	50% accuracy
Gunetti & Picardi [15]	N-graph duration	Relative distance, absolute distance	205	765	0.005% FAR, 5% FRR
Villani et al. [20]	Di-graph latency & duration, key duration, typing speed, percentage of special characters, editing patterns	Euclidian distance, k-nearest neighbour	118	2360	99.8% - 44.2 % accuracy
Curtin et al. [19]	Di-graph latency & duration, key duration, typing speed, percentage of special characters, editing patterns	Euclidian distance, k-nearest neighbour	30	-	100% - 97% accuracy
Filho & Freire [30]	Di-graph latency	Simplified Markov chain model	15	150	41.6% - 12.7% EER
Janakiraman & Sim [24]	Di-graph latency, key duration	Bhattacharyya distance	22	-	100% - 70% accuracy
Buch et al. [34]	Di-graph latency & duration, percentage of special characters	Euclidian distance	36	650	100% - 98% accuracy
Hu et al. [25]	N-graph duration	Relative distance, absolute distance, k-nearest neighbour	36	36554	0.045% FAR, 0.005% FRR
Hempstalk et al. [21]	Di-graph latency, key duration, typing speed, error rate, P-R ordering	One-class classification	10	150	11.3% FAR, 20.4% FRR
Ahmed et al. [29]	Di-graph latency	Neural network	22	-	0.015% FAR, 4.82% FRR
Davoudi & Kabir [12]	Di-graph duration	Relative distance, absolute distance, histogram-based density estimation	21	315	0.015% FAR, 0.0025% FRR
Pilsung et al. [13]	Di-graph duration	Kolmogorov-smirnov Test	-	-	0.17% EER
Samura & Nishimura [23]	Di-graph latency & duration, key duration	Weighted Euclidian distance	112	-	67.5% - 81.2% accuracy
Bours & Barghouthi, [10]	Di-graph latency, key duration	Distance measure	25	-	79 – 348 keystrokes
Davoudi & Kabir [26]	Di-graph duration	Modified relative distance	21	315	0.08% FAR, 18.8% FRR
Davoudi & Kabir [14]	Di-graph duration	Weighted relative distance	21	315	0.07% FAR, 15.2% FRR
Park et al. [17]	Key-pair duration	Kolmogorov-smirnov Test	35	-	0.089% EER
Messerman et al. [38]	N-graph duration	Normalized relative distance	55	-	2.20% FAR, 1.84% FRR
Singh & Arya [18]	Key-pair latency	Euclidian distance	20	-	0.02% FAR, 0.04% FRR
Chantan et al. [28]	Di-graph duration	Bayes classifier	-	-	0% EER
Bakelman et al. [27]	Di-graph duration	K-nearest neighbour	20	200	4% EER
Bours [9]	Di-graph latency, key duration	Scaled Manhattan distance	25	-	182 keystrokes

Pattern recognition methods were also exploited in order to be used as a classification method for free-text keystroke authentication. For example: K-nearest neighbor was used in [27] and Bayes classifier was used in [28].

Ahmed et al. [29] used a feed forward multi-layer perceptron neural network system for the purpose of classifying users. Two neural networks were used; a behavior-modeling network and a detection network. The first used the di-graph's first and second keys press and release times to find the elapse time it took a user to press two successive keys. The second neural network used the di-graph's times and the matching output from the behavior-modeling network to estimate which user's typing patterns it represented.

5. Performance

Unfortunately, not only keystroke systems but all biometric authentication systems sometimes suffer from mistakes in the authentication decision. This is due to a number of reasons that has to do not only with the efficiency of the technique but also with the user himself or with his surroundings. First of all it is possible, yet not likely, that an imposter is mistakenly identified as the legitimate user if by chance the two persons typing patterns are close enough to the extent that the classification method fails to distinguish between them. Conversely, when one of the legitimate user's fingers slips off the keyboard and causes the typing pattern to change slightly, the user may not be successfully authenticated. Thus, it is important to have some metrics to exactly measure the error rate which will help to identify the performance level that can be expected and tolerated by that system's users.

A very simple way to measure the error rate was used in earlier studies; using the Accuracy measure which is the percentage of successfully authenticated attempts compared to the total number of completed attempts. This technique was adapted in [1, 27, 22].

The most frequently used error rates for inferring the performance of an authentication system are: the False Accept Rate (FAR), also referred to as the Imposter Pass Rate (IPR) and the False Reject Rate (FRR), also called the False Alarm Rate (FAR). The FAR is the percentage of impostors who have successfully gained access to the system while the FRR is the percentage of legitimate users who have been denied access to the system. These two error rates were used by the majority of free-test keystroke systems including [15, 21, 18].

Clearly, there is a trade-off between the FAR and FRR which can be controlled according to the level of security strictness required. FAR is required to be as low as possible in strictly secure applications while there is a compromise of having a higher FRR. Meanwhile, a higher FAR is acceptable in systems where security is not the major aim yet system usability has higher priority.

The other commonly used error rate is the Equal Error Rate (EER), also referred to as Cross-over Error Rate (CER), which is the value where FAR and FRR are equal. It was used in many methods such as [17, 30, 27] where lower EER values indicate a more secure system.

Due to the fact that free-text keystroke authentication is a continuous process, another metric which defines exactly how much time, in number of keystrokes, did it take the system to discover that an imposter had had access to the system has been proposed in some studies. This aims to detect the impostor as fast as possible, incorporating as few keystrokes as possible. This implies that an attacker would be detected before he can do more harm to the system. A penalty-reward technique was introduced in [9, 10] where a user was initially given the highest trust level prior to the user being successfully authenticated via a static authentication procedure. During the typing session, the user obtained a reward which he received in the form of an increase of his trust level when he typed in a manner sufficiently close to his typing template. Likewise, he obtained a penalty in the form of a decrease of his trust level when he typed in a manner far from his typing template. The system then locks-out a user if his/her trust level falls below a pre-determined threshold.

6. Factors Affecting Performance

There are many different performance measures used to determine the error rate in free-text keystroke systems, it is therefore often difficult to compare studies. This is also due to not having any form of standardization in the data collection process in these different experiments. Even though, the error rate in study A is lower than the error rate in study B, that does not necessarily mean that the method adapted in A is better than that used in B. Different factors may have a positive or a negative impact on the authentication process regardless of the actual method's functionality. Standardization of such factors requires information exchange amongst researchers which would offer an improved comparing mechanism between different algorithms. There are a lot of different factors to be considered in free-text keystroke systems; a detailed list of these factors is provided in this section.

Nevertheless, there are some solutions that can be used to standardize the factors involved. The first solution is using a widely available automated program for collecting data. A broad range of software is available commercially; for example: BehavioSec and KeystrokeID. Another solution involves the use of standardized databases which has been formerly created and published for the purpose of keystroke dynamics research. A list of some of the databases available online can be found in [31]. Using these solutions could not only standardize the data collection method, it could also decrease a duplication of effort among researchers.

6.1 Environment Controlling

There are two basic categories in the way experiments have been conducted in free-text keystroke studies.

Experiments have either been conducted in a controlled environment or in an uncontrolled one. In a controlled environment, users are asked to type on a specific machine which has built-in software for recording the keystrokes. Thus, the same external conditions are consistent for all users. The issue with this kind of arrangement is that it may not have the same characteristics as those encountered in realistic situations, therefore, the response may not be representative of a user's typical typing patterns.

In uncontrolled environments, on the other hand, users are asked to either download a program on their machines to collect their keystrokes [24, 30] or to use an online data collection form [15, 25]. This indicates that the data is collected wherever and whenever it is convenient to the user. Although, this method provides a realistic representation of normal circumstances for the user, each user's surroundings can be very different, which makes the data harder to analyze. This might be the reason for inconsistencies in the keystroke data provided by the users. A lot research done using free-text keystroke systems has so far been conducted in uncontrolled environments due to the desire to imitate the lifelike conditions of a real authentication system [15, 12, 28].

6.2 Keyboard Type

Using different brands of computer has a big impact on the user's typing pattern since the keyboard of different brands differs in key size and spacing between keys which is clearly a reason that users may type differently than normal [31]. Furthermore, different keyboards have different key pressing sensitivity levels which consequently may affect the timing data collected from the users. Using a laptop keyboard adds another variation which can also affect the typing behavior; because laptops provide the freedom of movement, users may use it in different positions such as on a bed or on a table.

Villani et al. [20] investigated the case of using different keyboards in free-text keystroke systems. One of their experiments was conducted using a desktop keyboard and another was performed on a lab top keyboard. A significant finding was produced in this study which can be summarized as: the system has a good chance of accurately identifying a user as long as he uses the same type of keyboard for training and testing. It is therefore important that researchers attempt to stick to using the same keyboard in order to maintain the same level of consistency throughout the data collecting process [19].

6.3 Entry Mode

Because free-text keystroke systems are used for long text, it makes more sense to allow the users to enter whatever text they prefer. Having said that, studies conducted have actually used two different methods for text entry in the experiments conducted for free-text authentication. The first technique allowed the users to type completely free text as they desired, such as: typing an e-mail or typing a report for work or an essay for school [15, 16]. The second approach required the users to type a specific long text from an article, in which the users needed to copy specific text into a section specified for text entry [19, 25].

In the research conducted by Villani et al. [20], participants were asked to be a part of several experiments with different conditions. One of these tests incorporated a copy-task in which the participants were asked to copy a predefined long text. Another included a free-text input where users were free to type-in arbitrary text. In this study, it was found that the accuracy of correctly authenticating a user decreased considerably when the user used different input modes in the training and testing phases. Moreover, it was also shown that the accuracy in free-text typing mode was higher than that in the copying-task mode. This can be explained by the frequent pauses that a user has to perform in order to look at the text during the copy-task which might cause the collected data to be inconsistent.

6.4 Text Length

One area that keystroke systems lack in is the amount of information that can be obtained. The only data that can be collected while the user is actually typing is the time each key is pressed and released, from which only little information can be inferred, including the time interval between each two consecutive keys and the duration time for each key press. In addition the data is often not stable since it changes based on the environment surrounding the experiment or based on the state of mind of the user at the time. As a result, to reduce the effect of such instabilities, much research has shown more interest in using short free-text [e.g. 28, 1, 18]. Realistically though, it is not enough to use short texts to analyze keystrokes since it does not offer an adequate amount of information to distinguish between users. Consequentially, using longer sample texts is considered a better alternative [11, 15, 16].

Moreover, Curtin et al. [19] provided evidence that using long-texts increases the chance of having more repetitions of the same di-graph in the training and testing samples which will, consequently, increase the stability of its mean and standard deviation significantly. The only problem with using long texts systems is that the training phase unavoidably needs more time. In their experiment, Curtin et al. investigated the accuracy of identifying users when typing long-texts under the condition that training and testing texts were different in length. The accuracy from different text/same length experiments was better than that from different text/different length experiments. Therefore, improving authentication accuracy can be achieved via standardization of the feature measurements i.e. the text size in this case.

6.5 User's Experience

The user's health and state of mind are a very crucial part of the authentication process using keystroke dynamics. The user's typing skills and level of comfort while using a keyboard are additional characteristics that have a clear impact on the user's typing behavior. The more skillful the user is, the more stable his/her fingers are located on

the keyboard and the more familiar he is with the position of each character on the keyboard. This will result in a more consistent typing pattern all through.

Samura and Nishimura [23] conducted a study that examined keystroke dynamics for long free-texts. The experiment participants were divided into three groups based on their typing speed, specifically the number of letters typed in a 5 minute period. This study indicated that the best recognition accuracy was obtained from the group which typed fastest.

6.6 Monitoring Mode

A free-text keystroke system is a continual process of identity verification which is taking place during the course of the whole time a user is using the system. This can be done in either a continuous manner or a periodic manner. Continuous authenticating is done, in real time, every time a key is clicked on the keyboard [9]. Although this method provides strong imposter detection, it is computationally expensive. Periodic authentication, alternatively, is repeated every time a certain text is entered [24]. This is a less strict method, security wise, yet it is computationally cheaper. Moreover, waiting until a specific text is entered may cause the system to wait for long periods of time if this particular text does not occur frequently enough in the typed text; which will represent a security threat for the system.

A periodic verification scheme that included the use of interruptions was utilized in [27]. In this research, the identity of the user was only verified after text breaks e.g. user leaving the PC for a coffee break. The system only captured the first burst of input after each pause in order to analyze it. The method does though reduce the frequency of authentication checks which is a key reason for reducing the false alarm rate in addition to decreasing the computational cost.

6.7 Words Choice

As mentioned, some free-text systems depend on periodic authentication where the authentication process is actually performed every time a particular text is entered. It is clear that choosing a specific piece of text is crucial for training and testing the system. It might be thought that using familiar English words may realize more consistent typing patters. However this has been shown to be wrong by Janakiraman and Sim [24].

In their research, Janakiraman and Sim introduced a new "goodness" measure which was suggested to be used to calculate the universality, accuracy and expectancy of a word used for free-text keystroke authentication. Universality is a measure to identify if a word is one of the words commonly used by users or not. Accuracy measures how unique a word is. Lastly, expectance is used to calculate the average number of keystrokes typed before that word actually appears in the text. Unexpectedly, using the goodness measure, the result of this experiment revealed that non-English words, such as: 'tmr' which is an abbreviation of 'tomorrow' used in online chats, are better than English words for identification and verification purposes.

6.8 Number of Training Samples

When considering the training phase in fixed-text keystroke systems, it is hard to ignore the time required for training the system by retyping the password again and again. This is not an issue in free-text keystroke systems where the user's data is collected while performing daily tasks. This implies that the free-text method is more practical in real life situations and easier to use since it causes less burden for the user. For example, 15 samples were collected from the participants over a two weeks period in [27]; each sample was 400 characters long of whatever the user needed to type at the time. This demonstrates that even though the samples were long, they were collected transparently to the user.

From the experiment results conducted by Gunetti et al. [32] it was found that the accuracy of the system generally escalated when the number of samples in the user's profile was increased. Meanwhile an effective mechanism for profile enhancement was suggested in [18] where the user's profile was expanded, during the typing session, by adding new key-pairs timing data attained from text entered by the user after being authenticated.

7. Applications

Although more than a quarter of a century has passed since keystroke authentication was first researched, it has not yet been applied much in the security field. In addition to the security that keystroke authentication systems can provide by locking-up the workstation when an imposter is detected at any point of time during which the system is used, a wide variety of other applications can also benefit from such authentication schemes. The applications, listed in this section, are examples of some situations where free-text keystroke authentication is more applicable than fixed-text systems.

7.1 Different Languages Authentication

Most of the work done on keystroke dynamics has concentrated on using the same language for training and testing the system. Gunetti et al. [32] though gave empirical proof that free-text typing patterns could be used to authenticate the user even when the test samples were written in different languages to that of the samples in the user's profile. Evidently, this only works when the two languages share a significant number of di-graphs. So, languages like English and Italian which have largely the same alphabet can be used for this kind of authentication but English and Arabic, for example, cannot be used because they have a completely different set of letters.

The data used in this study was provided by Italian speakers each of whom provided two samples typed in Italian and another two samples typed in English. From the experimental results, about 10% mistakes in identification occurred when the test sample was in a

language different to that of the user's profile. Better performance was obtained when the user's profile contained samples in both languages. The error rate was even smaller when the test sample was in the same language as that which dominated the samples in the user's profile. By experimenting with different combinations of template samples and test samples, it was clear that samples provided by the same person while typed in different languages were more similar than samples provided by different persons while typed in the same language. An average performance of 1.61% FRR and 3.23% FAR was achieved in total. Thus, keystroke authentication for different language texts, is possible, though more difficult than the case where all samples are in the same language.

7.2 Old Profile Authentication

Most of the studies conducted in the free-text keystroke authentication field have had only a few months gap between the time the training samples were collected and the time in which the test samples were gathered. Gunetti et al., however, showed in [33] that a typing profile could still be used to identify a user, even though, it has been created a long time before the test samples are provided and investigated. Their original experiment involved 30 participants whom were asked to provide 15 samples each. The samples consisted of whatever the users choose to type. One and a half years later, the same 30 volunteers were asked again to provide another two free-text samples. It was discovered that even after such a long period of time their keystroke dynamics system was still able to identify users with an average accuracy of a 1.67% FAR and a 11.67% FRR.

7.3 Intrusion Detection

The continual authentication scheme that the free-text method provides is a very effective intrusion detection method. It is mainly used to notice any warnings with regards to irregularities in the typing patterns of a specific user. Moreover, free-text keystroke systems are used for active monitoring of the system which can aid in finding any intrusion quickly and reliably. One important issue that has to be addressed here is the generation of false alarms in continual keystroke-based authentication systems. It might cause frequent and rapid system halts with much annoyance for the users when they falsely occur. Therefore, Gunetti et al. [32] suggested using it combined with other authentication methods in order to reduce the false alarm rate.

7.4 Online Marketing

Free-text keystroke systems can also be utilized for identifying users over the internet. This is done by capturing a user's typing patters on their first visit to the website and then it can be used to identify returning users [32]. This data can be used to determine user preferences and interests which can be employed for marketing purposes. This approach, on the other hand, has many privacy issues regarding the amount of information that users are happy to hand-in to the websites they visit.

7.5 Cybercrime Investigating

User tracking through typing patterns can also help in cybercrime and investigating illegal electronic movements of anonymous users. Using free-text keystroke schemes was suggested for network forensics in [29] through attacker profiling which is conducted by collecting his/her typing patterns when surfing websites on the internet. This profile, collected for each user, can be used as a digital fingerprint gathered from the cybercrime scenes. This is considered as passive fingerprinting because it can be created without the knowledge of the attacker which can be extremely beneficial in fraud or identity theft cases where attackers are completely oblivious that they are being monitored. The issue with such a digital fingerprint is that it must be built progressively which requires a lot of internet service providers to collaborate and work together in facing such threats.

7.6 Identification and Authentication

Keystroke dynamics systems are used for two different purposes. Firstly: identification, which is a way of determining the user's identity when no data is available about their identity beforehand. In this method, a test sample is matched with all the templates stored in a database. The system assigns the user to the identity of the person whose template is the most similar to the test sample. The second purpose is authentication which is used to verify the identity of the user. The user supplies his identity and the system takes on the responsibility of making sure that the user is who he/she claims to be. The test sample in this case is only compared with the user's template in the database.

The complexity of performing identification is clearly higher than that of authentication since it includes comparing the test sample with all available templates which may be a very large undertaking in large scale systems. Identification also requires a larger amount of data i.e. longer text. From the definition of both methods, fixed-text keystrokes system is used mostly for authentication since it employs a password that is considered a mean for providing the user's identity. Free-text keystrokes system, on the other hand, is used for both identification and authentication [e.g. 34, 15].

7.7 User's Emotion Detection

Since free-text keystroke systems gather a lot of data from the user during the whole time he/she is using the computer, this data can also be used to infer the emotional state that the user is going through during the typing process. This has been employed in [35] to determine what the user is feeling during every day free typing. Feelings like frustration, focus, anger, stress, relaxation, excitement and tiredness were derived from the user's typing behavior. Extracting the emotional state that the user is going through in a particular period of time that the user is using the system has many benefits for intelligent

computers. It helps the system make the right decisions regarding the best interaction method to practice with the user. The issue with using keystrokes for user emotion detection is that it can cause an invasion into the user's typing experience. For example, in [35], the user was required to determine his emotion every 10 minutes in order to train the system to identify his emotions automatically.

8. Security Issues

In this section we discuss the security level that free-text keystroke authentication systems provide. A list of the most common threats is provided here along with the degree of safety that free-text keystroke systems deliver against these dangers [36].

1. Shoulder surfing and user mimicking: is an attack in which the attacker monitors the victim typing, during the typing process, in order to try imitating his/her typing behavior. Even though there is little possibility of an attacker successfully mimicking a user typing pattern in fixed-text keystrokes, it is even harder to do so in free-text systems. Since it requires the attacker to observe the user's behavior for the whole time the user is logged-in, it is very rare that an attacker can actually imitate all the aspects of the user's typing behavior.

2. Spyware: is software downloaded into the victims' computer without their consent which is used to record information about them. Spyware is perhaps the biggest threat to keystroke dynamic authentication systems because it can record exactly the time each key is pressed and released. This can be used by the attacker to simulate the legitimate user's typing behavior. Nevertheless, it is still a hard task for the attacker to undertake in the case of the huge amount of data that free-text systems need to analyze.

3. Social engineering: is manipulating the user in order to obtain his/her private information. Tricking the victim to reveal his typing pattern is though not possible using telephone calls or face to face meetings. Yet, phishing e-mails can be used to trick the user to type some text which can be used to extract the victim's typing patterns. But even then, the attacker has to get hold of a sufficient amount of keystrokes to be able to actually simulate the victim's free-text typing patterns.

4. Guessing: is trying to guess the way that a victim types. There are simply too many different ways that a user might normally follow when typing. Therefore, guessing the typing behavior of another person is almost impossible in free-text keystroke dynamics.

9. Conclusions

Free-text keystroke dynamics is a non-intrusive method, since it only uses the behavioral data that users convey during regular typing tasks. In addition to that, it is relatively inexpensive; the only required hardware is the keyboard. However the most important benefit that free-text keystroke systems provide is that the typing patterns can still be used for authenticating users even after the authentication phase has passed. In addition, free-text authentication provides a valuable balance between security and usability which is highly desirable in the businesses world.

One concern about free-text keystrokes is that it tends to be instable in the sense that it might be influenced by the user state or by environmental conditions. Indeed some level of instability might occur without any obvious cause. Therefore, free-text authentication is probably best used as a part of a multi-factor authentication scheme [28, 37] that provides a higher level of security.

Generally, it is obvious that keystroke dynamics works more accurately for fixed-text compared with free-text. Therefore, it might be a good practice for free-text tests to take into consideration the actual words that the user is typing, in addition to the key hold time the di-graph's duration and latency times.

Moreover, determining the best method to follow to achieve the best authentication accuracy is not a straightforward task. Due to the variation of conditions that might be affecting the study participants, environment or procedure, the comparison between two or more methods is not always accurate. Therefore, a standardization mechanism has to be established to assure that factors affecting performance are in agreement in all the studies and hence can be properly compared.

Lastly, it is clear that the idea of using keystroke dynamics is not only restricted to the traditional keyboard, it can be conveyed to many other mechanisms like ATM machines and cell phones, which will then provide better every day protection for the standard user.

References

[1] F. Monrose and A. Rubin, "Authentication via Keystroke Dynamics", in the Fourth ACM Conference on Computer and Communication Security, 1997.
[2] M. Karnan, M. Akila and N. Krishnaraj, "Biometric Personal Authentication Using Keystroke Dynamics: a Review", Applied Soft Computing, Vol. 11, No. 2, 2011, pp. 1565–1573.
[3] R. Gaines, W. Lisowski, S. Press and N. Shpiro, "Authentication by Keystroke Timing: some Preliminary Results", Rand Corporation, Rep. R-256-NSF, 1980.
[4] S. J. Shepherd, "Continuous Authentication by Analysis of Keyboard Typing Characteristics", in European Convention on Security and Detection, 1995, pp. 111-114.
[5] R. Giot, M. El-Abed, B. Hemery and C. Rosenberger, "Unconstrained Keystroke Dynamics Authentication with Shared Secret", Computers and Security, Vol. 30, 2011, pp. 427-445.
[6] J. Garcia, "Personal Identification Apparatus", U.S. Patent 4621334, 1986.
[7] M. S. Obaidat and B. Sadoun, "Verification of Computer Users Using Keystroke Dynamics", in IEEE Transactions on Systems, Man and Cybernetics - Part B: cybernetics, 1997, Vol. 27.
[8] K. S. Killourhy, "A Scientific Understanding of Keystroke Dynamics", PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, US, 2012.

[9] P. Bours, "Continuous Keystroke Dynamics: a Different Perspective Towards Biometric Evaluation", Information Security Technical Report, Vol. 17, 2012, pp. 36-43.
[10] P. Bours and H. Barghouthi, "Continuous Authentication Using Biometric Keystroke Dynamics", in The Norwegian Information Security Conference, 2009.
[11] P. S. Dowland, H. Singh and S. M. Furnell, "A Preliminary Investigation of User Authentication Using Continuous Keystroke Analysis", in the IFIP 8th Annual Working Conference on Information Security Management and Small Systems Security, 2001.
[12] H. Davoudi and E. Kabir, "A New Distance Measure for Fee Text Keystroke Authentication", in the14th International CSI Computer Conference, 2009, pp. 570-575.
[13] K. Pilsung, P. Jooseong, P. Sunghoon , Y. Joonha and C. Sungzoon, "Keystroke Dynamics Analysis Based on Long and Free Text", in Fall Korean Industrial Engineering Conference, 2009.
[14] H. Davoudi and E. Kabir, "Modification of the Relative Distance for Free Text Keystroke Authentication", in the 5th International Symposium on Telecommunications, 2010.
[15] D. Gunetti and C. Picardi, "Keystroke Analysis of Free Text", ACM Transactions on Information System Security, Vol.8, No.3, 2005, pp. 312-347.
[16] T. Sim and R. Janakiraman, "Are Digraphs Good for Free-text Keystroke Dynamics?", in the IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp.1-6.
[17] S. Park, J. Park and S. Cho, "User Authentication Based on Keystroke Analysis of Long Free Texts with a Reduced Number of Features", in the Second International Conference on Communication Systems, Networks and Applications, 2010.
[18] S. Sing and K.V. Arya, "Key Classification: a New Approach in Free Text Keystroke Authentication System", in Third Pacific-Asia Conference on Circuits, Communications and System, 2011, pp. 1-5.
[19] M. Curtin , C. Tappert, M. Villani, G. Ngo, J. Simone, H. St. Fort, and S.-H. Cha, "Keystroke Biometric Recognition on Long-text Input: a Feasibility Study", in IMECS, 2006.
[20] M. Villani, C. Tappert, G. Ngo, J. Simone, H. St. Fort and S. Cha, "Keystroke Biometric Recognition Studies on Long-text Input Under Ideal and Application-oriented Conditions", in the IEEE Computer Society Workshop on Biometrics, 2006.
[21] K. Hempstalk, E. Frank and I. H. Witten, "One-class Classification by Combining Density and Class Probability Estimation", in the European Conference on Machine and Learning and Principles and Practice of Knowledge Discovery in Database, 2005, pp.505-519.
[22] D. Gunetti and G. Ruffo, "Intrusion Detection through Behavioral Data", in the Third International Symposium on Advances, 1999, pp. 383-394.
[23] T. Samura and H. Nishimura, "Keystroke Timing Analysis for Individual Identification in Japanese Free Text Typing", in ICROS-SICE International Joint Conference, 2009.
[24] R. Janakiraman and T. Sim, "Keystroke Dynamics in a General Setting," in Advances in Biometrics, 2007, Vol. 4642, pp. 584–593.
[25] J. Hu, D. Gingrich and A. Sentosa, "A K-nearest Neighbor Approach for User Authentication through Biometric Keystroke Dynamics", in IEEE International Conference on Communications, 2008, pp. 1556-1560.
[26] H. Davoudi, E. Kabir, "User Authentication Based on Free Text Keystroke Patterns", in the 3rd Joint Congress on Fuzzy and Intelligent Systems, 2010.
[27] N. Bakelman, J. V. Monaco, S. H. Cha, and C. C. Tappert, "Continual Keystroke Biometric Authentication on Short Bursts of Keyboard Input", in Pace University CSIS Research Day, 2012.
[28] C. Chantan, S. Sinthupinyo and T. Rungkasiri, "Improving Accuracy of Authentication Process via Short Free Text Using Bayesian Network", International Journal of Computer Science Issues, Vol. 9, No. 3, 2012.
[29] A. A. E. Ahmed, I. Traore and A. Almulhem, "Digital Fingerprinting Based on Keystroke Dynamics", in the Second International Symposium on Human Aspects of Information Security and Assurance, 2008.
[30] J. R. M. Filho and E. O. Freire, "On the Equalization of Keystroke Timing Histogram", Pattern Recognition Letters, Vol. 27, No.13, 2006, pp. 1440-1446.
[31] S. P. Banerjee and D. L. Woodard, "Biometric Authentication and Identification Using Keystroke Dynamics: a Survey", Journal of Pattern Recognition Research, Vol. 7, 2012, pp. 116-139.
[32] D. Gunetti, C. Picardi, and G. Ruffo, "Keystroke Analysis of Different Languages: a Case Study", in the Advances in Intelligent Data Analysis, 2005, Vol. 3646, pp. 133–144,
[33] D. Gunetti, C. Picardi, and G. Ruffo, "Dealing with Different Languages and Old Profiles in Keystroke Analysis of Free Text", in the Advances in Artificial Intelligence, 2005, Vol. 3673, pp. 347–358.
[34] T. Buch, A. Cotoranu, E. Jeskey, F. Tihon and M. Villani, "An Enhanced Keystroke Biometric System and Associated Studies", in Pace University CSIS Research Day, 2008.
[35] C. Epp, M. Lippold and R. L. Mandryk, "Identifying Emotional States Using Keystroke Dynamics", in the Conference on Human Factors in Computing Systems, 2011, pp. 715-724.
[36] D. Shanmugapriya and G. Padmavathi, "A Survey of Biometric Keystroke Dynamics: Approaches, Security and Challenges", International Journal of Computer Science and Information Security, Vol. 5, No. 1, 2009.
[37] O. S. Adeoye, "Evaluating the Performance of Two-factor Authentication Solution in the Banking Sector", International Journal of Computer Science Issues, Vol. 9, No. 2, 2012.
[38] 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," in the International Joint Conference on Biometrics, 2011, pp.1-8.

Arwa Alsultan is pursuing a PhD degree in Computer Science from the School of Systems Engineering at the University of Reading, Reading, Berkshire, UK. She completed her Master's degree in Computer Science from the Computer and Information Science College at the King Saud University, Riyadh, SA in 2010. She works as a lecturer at the IT Department in the Computer and Information Science College at the King Saud University, Riyadh, SA.

Kevin Warwick is Professor of Cybernetics at the University of Reading. His research interests are in Artificial Intelligence, Robotics, Biomedical Engineering and Control Systems. He has D.Sc. degrees from both Imperial College London and the Czech Academy of Sciences. He has published over 500 research papers and is perhaps best known for his experimentation with implant technology.

IET Biometrics

Research Article

IET Journals
The Institution of Engineering and Technology

Free-text keystroke dynamics authentication for Arabic language

ISSN 2047-4938
Received on 29th October 2015
Revised on 11th January 2016
Accepted on 8th February 2016
doi: 10.1049/iet-bmt.2015.0101
www.ietdl.org

Arwa Alsultan[1] ✉, *Kevin Warwick*[2], *Hong Wei*[1]

[1]*School of Systems Engineering, University of Reading, Reading RG6 6AH, UK*
[2]*Vice Chancellors Office, Coventry University, Priory Street, Coventry CV1 5FB, UK*
✉ *E-mail: a.f.a.alsultan@pgr.reading.ac.uk*

Abstract: This study introduces an approach for user authentication using free-text keystroke dynamics which incorporates text in Arabic language. The Arabic language has completely different characteristics to those of English. The approach followed in this study involves the use of the keyboard's key-layout. The method extracts timing features from specific key-pairs in the typed text. Decision trees were exploited to classify each of the users' data. In parallel for comparison, support vector machines were also used for classification in association with an ant colony optimisation feature selection technique. The results obtained from this study are encouraging as low false accept rates and false reject rates were achieved in the experimentation phase. This signifies that satisfactory overall system performance was achieved by using the typing attributes in the proposed approach, while typing Arabic text.

1 Introduction

The ongoing quest to find a technique to protect sensitive data and computer systems from harmful imposters, whilst maintaining ease of use, is an important challenge in the field of information security. This paper focuses on a novel method that verifies the identities of users based on their unique typing rhythms in Arabic language. Keystroke dynamics is considered to be an effortless behaviour-based method for user authentication which employs the person's typing patterns for validating his/her identity. As mentioned in [1], keystroke dynamics is 'not what you type, but how you type'. In this approach, the user types in text, as usual, without any extra work to be done for authentication. Moreover, it only involves the user's own keyboard and no other external hardware.

Keystroke dynamics is based on timing features that compute time lapses between two actions on the keyboard such as key press and key release. In this paper, we investigate the use of such timing features with Arabic input. We consider a keyboard-layout-based method to compare timing features of free-text typing samples. A large feature set is deployed in this research for the purpose of trying to find the best representative features of the typing patterns in human behaviour using the least amount of training, specifically while typing in Arabic.

Keystroke dynamics have been studied comprehensively using English input. Other languages have not yet received the same attention as English has in the research literature to date. Languages such as Italian, which share the same alphabet with English, have been included in some research on keystroke dynamics, e.g. [2]. To our knowledge, there has been no reported research that has utilised Arabic input in keystroke dynamics authentication. Use of handwriting identification in Arabic writing has however been reported on in the Arabic-related literature [3]. Therefore, in this paper we are attempting to incorporate Arabic input in keystroke dynamics user authentication. The Arabic language has completely different characteristics to those of English, thus using typing patterns for Arabic input to authenticate users is an important contribution of this paper.

Arabic and English languages are very different to each other. Whereas English is a Germanic language from the Indo-European language family, Arabic is a Semitic language belonging to the Afro-Asiatic language family [4]. Arabic has 28 letters which are completely different from the English alphabet. Moreover, Arabic text is written from right to left which is a unique characteristic for merely Arabic, Urdu and Hebrew scripts [5]. In addition, there is no distinction between upper and lower cases in Arabic. Punctuation rules are much looser than those in English and less commonly used [6].

In this paper, decision trees (DTs) and support vector machines (SVMs) associated with ant colony optimisation (ACO) are used to classify the typing samples collected from participants.

The rest of this paper proceeds as follows. Section 2 briefly introduces keystroke dynamics theory and discusses similar prior research in the area of keystroke dynamics user authentication. Section 3 describes the method developed in this paper, in which we discuss the timing features included in this paper. In Section 4, we present our experimental results and consider the data space under investigation. Discussion about our results and some comparisons with previous studies are also included in this section. The final section concludes the topic and points out our research contributions and future work.

2 Related work

There are two basic classes of keystroke dynamics: namely, fixed-text and free-text [7]. The fixed-text keystroke dynamics method uses the typing pattern of the user while entering a predefined text. This text has been previously used to train the system and is delivered by the user at log-in time. Contrariwise, the free-text keystroke method is considered easier for the user as it overcomes the problem of memorising the text, something that the fixed-text keystrokes method suffers from [8]. As its name suggests in the free-text keystrokes method, the text used for enrolment does not have to be the same as the text used for log-in. Moreover, free-text keystroke dynamics is used for enhancing security through continuous and non-intrusive authentication [9]. Thus, the latter method is the one that has been considered in this paper as it can be applied in many useful settings to aid in real-life situations in addition to the benefit it provides in balancing between security and usability [10].

Keystroke dynamics is utilised in user authentication by extracting timing features at the log-in session and comparing them with the timing features extracted at the enrolment session. These features

IET Biom., 2016, Vol. 5, Iss. 3, pp. 164–169

include, among others: typing latency [11], keystroke duration [1], typing speed and shift key usage patterns [12]. Another feature which requires a specific keyboard for its measurement is typing pressure [13]. If the extracted features are adequately similar, the user is authenticated and if not the user might be denied access or at least asked to provide further identity information.

A large amount of research has been carried out over the years to investigate how keystroke dynamics can aid user authentication. Joyce and Gupta [11] used a statistical method that employs the absolute distances between the means of the signature data and test data; each of which consists of a fixed-text that includes username, password, first name and last name.

Gunetti and Picardi [14] meanwhile introduced an effective method for free-text authentication which was further explored by many other researchers. Their method was based on two measures: a relative (R) measure and an absolute (A) measure. These measures were used to calculate the degree of disorder and the absolute distance between two samples that share some *n*-graphs, i.e. *n*-characters-long letter combinations.

Another technique was introduced by Singh and Arya [15] that considered benefiting from key-pairs for free-text authentication. In that research, a keyboard grouping technique was used for creating timing vectors of flight times entered by the user. The keys were grouped based on their position on the keyboard, which was divided into 8 sections; two left and right halves and then each half divided into 4 lines representing the rows of the keyboard. The Euclidean distance was then used to calculate the similarity between the two vectors.

Other researchers relied on pattern recognition classifying methods such as the work done by Hu *et al.* [16]. They used the *k*-nearest neighbour approach together with the distance measurement proposed by Gunetti and Picardi [14] in order to classify the users' keystroke dynamics profiles. Neural networks have also been used for keystroke pattern classification such as the research conducted by Raghu *et al.* [17], in which they incorporated a three-layered back propagation neural network to verify the identity of users.

It should be indicated that the previously mentioned studies, involved only English input from the user. Whilst such experimentation is very important, there is clearly a lack of language variation used in such systems.

The work done by Gunetti *et al.* [2] is one of the very few research involving languages other than English in keystroke dynamics. In that study, a combination of the two measures developed by Gunetti and Picardi [14] is also used to assess the similarities between the typing patterns using the duration time of the di-graphs in samples typed in both English and Italian. Italian was used since the two languages, i.e. English and Italian, share a considerable number of di-graphs (key-pairs). From experimenting with different combinations of template samples and test samples, the best results were achieved when the user's profile contains samples in both languages yet the performance increases when the language of the testing samples is the dominant language in the samples forming the user profile.

Another study directly relevant to the research reported in this paper was that by Samura and Nishimura [18], in which they conducted a study that examined keystroke dynamics for long free-texts in Japanese language. In this paper, hold time and flight time of Japanese language-specific keystrokes were used as timing features. To compare the test and training timing vectors, a weighted Euclidean distance was used. However, though this paper was applied to Japanese language, the keyboard used was an English standard keyboard. Subjects carried out the typing process by entering the alphabet letters (in English) corresponding to the Japanese characters.

3 Methodology

3.1 Key-pair formation

This research examines the use of Arabic input in the novel approach we introduced in [19] for free-text keystroke dynamics authentication. This approach specifically makes use of the keyboard's key-layout. The technique employs the keystroke features extracted between two keys (key-pair) that are pressed consecutively and have a relationship on the keyboard layout. This relationship depends mainly on the key position of each character on the keyboard in relation to the other characters. Moreover, these relationships can vary depending on the location of the two keys with respect to the overall keyboard layout. The keyboard used in this paper was the standard Arabic keyboard since it is the most commonly used Arabic keyboard [20].

There are five categories for key-pair relationships:

(i) Adjacent: keys located next to each other on the keyboard.
(ii) Second adjacent: keys that are one key apart from each other.
(iii) Third adjacent: keys that are two keys apart.
(iv) Fourth adjacent: keys that are three keys apart.
(v) None adjacent: keys that are more than three keys apart.

Fig. 1 illustrates the key relationship concept when considering the key 'ل'. This is just an example, as the relationship formation can be performed in the same way for all the key-pairs in the typed text.

Each of these relationship categories can fall into one of the following overall locations:

(i) Both keys are on the right-hand side of the keyboard.
(ii) Both keys are on the left-hand side of the keyboard.
(iii) The two keys are located on different sides of the keyboard, i.e. the first key is located on the right-hand side while the second key is on the left or vice versa.

For further explanation, an example is provided in Fig. 2. One colour represents the right-hand side section of the keyboard, whereas the other colour represents the left-hand side. Other characters that are not located in the main part of the keyboard, e.g. the num-pad keys, arrows and the function keys are excluded from the key-pair formation; they are therefore shown in white.

As an example, we demonstrate the key-pairs forming the text 'نتلائم' which is entered from right to left as the following sequence: 'م ئ ا ل ت ن'. The key-pairs are:

- 'ن ت': Adjacent/RightSide.
- 'ت لا': SecondAdjacent/DifferentSide.

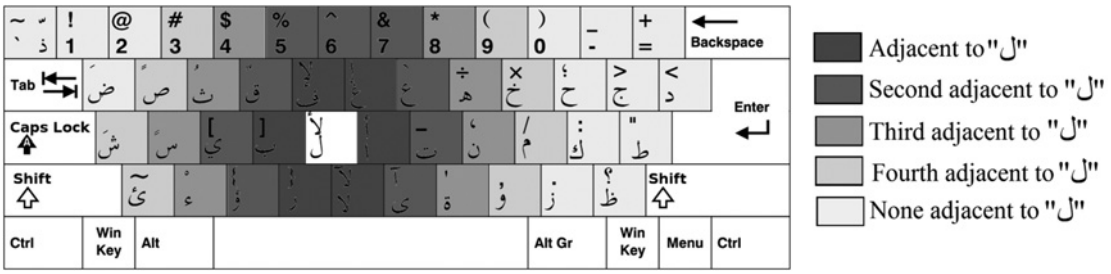

Fig. 1 *Key-pair relationship formation*

Fig. 2 *Overall key location*

- 'لا ى': FourthAdjacent/LeftSide.
- 'ى م': NonAdjacent/DifferentSide.

This process is pursued to break the text down into key-pairs and then classify each key-pair typed on the main part of the keyboard. In total, there are 15 different combinations of key-pairs that any two keys can be classified into.

Given that the key-pairing method significantly boosts the number of key-pairs that can be found and compared in the training and testing samples, it was adopted as a way to increase the soundness of the mean and standard deviation of the timing features. This will help to increase the stability of the timing vectors. This is an obvious benefit of the suggested scheme as it utilises a small amount of typing data in the best possible way. Therefore, it succeeds in authenticating users based on the smallest amount of training possible.

For example, the following training and testing data have only two similar key-pairs ('ال' and 'ير') to which typing times can be compared in the conventional keystroke dynamics authentication process, i.e. without the use of key-pairs [14]. However, this is not the case when using the key-pair method as there are more instances of each key-pair extracted from both the training and testing data.

Training data: 'قرد الغابة صغير'.
Testing data: 'حوت البحر كبير'.

Pairs that involve spaces were discarded because of the work conducted in [15], which provided evidence that a user normally experiences unusual pauses before and after pressing the space key, thereby leading to inconsistent typing behaviour. Moreover, key-pairs that included the backspace key were also excluded for similar reasons.

3.2 Feature definition

Once the key-pairs have been obtained from the users' raw data, the keystroke features are extracted [21]. These features were computed for every key and key-pair using two main values, specifically: the press time (D_n) and the release time (U_n) of each key (n) in milliseconds. In this research, five keystroke features were extracted from each key-pair as shown in Fig. 3:

(i) Dwell time or keystroke duration or hold time: is the time for which a key is pressed until it is released. Consequently, each key-pair has two hold times:
  (a) Hold time for the first key (H1).
  (b) Hold time for the second key (H2).

(ii) Flight time or keystroke latencies: there are three types of latencies:
  (a) Down–Down (DD) or Press–Press time: is the interval time between two successive key presses.
  (b) Up–UP (UU) or Release–Release time: is the interval time between two successive key releases.
  (c) Up–Down (UD) or Release–Press time: is the interval time between a key release and the next key press.

3.3 Feature subset selection and classification

Five timing features were defined for each key-pair appearance in the text. This was done for all 15 types of key-pairs. Therefore, the overall number of timing features was 75 (5 timing features × 15 key-pairs). Table 1 lists all the 75 features extracted from all key-pairs. The feature abbreviations listed in this table combine the key-pair category and the timing feature, for example: 'AR-H1' stands for: Adjacent/RightSide-Hold1 and so on.

Having such a large feature set in its entirety adds more computational cost in addition to raising the complexity of the classification process [22]. Therefore, it is necessary to incorporate a feature subset selection mechanism.

Feature subset selection is considered as an optimisation problem, in which the space of all possible features is scrutinised to recognise the feature or set of features that produce optimal or near-optimal performance, i.e. those that minimise the classification error [7]. ACO proved to be a good candidate for achieving that goal [19].

SVMs have been chosen as a classifier in this research as it is one of the most successful classification techniques [23]. Moreover, for

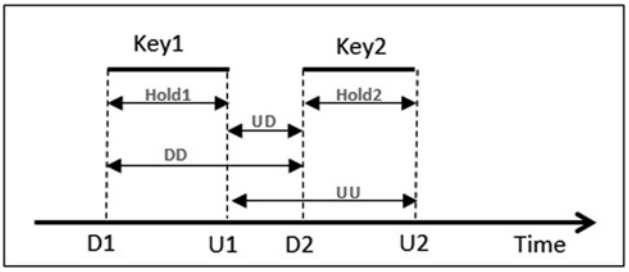

Fig. 3 *Keystroke timing features*

Table 1 Feature set

Key-pair category	Feature set				
Adjacent/RightSide	AR-H1	AR-H2	AR-DD	AR-UU	AR-UD
Adjacent/LeftSide	AL-H1	AL-H2	AL-DD	AL-UU	AL-UD
Adjacent/DifferentSide	AD-H1	AD-H2	AD-DD	AD-UU	AD-UD
SecondAdjacent/RightSide	SR-H1	SR-H2	SR-DD	SR-UU	SR-UD
SecondAdjacent/LeftSide	SL-H1	SL-H2	SL-DD	SL-UU	SL-UD
SecondAdjacent/DifferentSide	SD-H1	SD-H2	SD-DD	SD-UU	SD-UD
ThirdAdjacent/RightSide	TR-H1	TR-H2	TR-DD	TR-UU	TR-UD
ThirdAdjacent/LeftSide	TL-H1	TL-H2	TL-DD	TL-UU	TL-UD
ThirdAdjacent/DifferentSide	TD-H1	TD-H2	TD-DD	TD-UU	TD-UD
FourthAdjacent/RightSide	FR-H1	FR-H2	FR-DD	FR-UU	FR-UD
FourthAdjacent/LeftSide	FL-H1	FL-H2	FL-DD	FL-UU	FL-UD
FourthAdjacent/DifferentSide	FD-H1	FD-H2	FD-DD	FD-UU	FD-UD
NonAdjacent/RightSide	NR-H1	NR-H2	NR-DD	NR-UU	NR-UD
NonAdjacent/LeftSide	NL-H1	NL-H2	NL-DD	NL-UU	NL-UD
NonAdjacent/DifferentSide	ND-H1	ND-H2	ND-DD	ND-UU	ND-UD

IET Biom., 2016, Vol. 5, Iss. 3, pp. 164–169

comparison purposes, DTs were also used in this experiment. DTs were chosen as a rival classifier because the technique follows a completely different mechanism to that of SVMs.

None the less, when using SVMs in classification, feature subset selection was in place in order to reduce redundancy among the features [24]. Contrariwise, feature subset selection was impeded in the building process of the DT where all redundant features were removed [25].

4 Experiment, results and discussion

4.1 Data space

A total of 21 users participated in this paper for data collection. All participants were native Arabic language speakers. They had different levels of typing skills that varied between moderate and very good.

During data collection, the participants were asked to perform two typing tasks. The tasks involved copying text that consisted of around 180 characters. The text employed was an excerpt from an Arabic online newspaper. The text included, in addition to letters, numbers and punctuation marks.

In this research, we used keystrokes produced by typing free-text to authenticate users. Free-text refers to the utilisation of any text for first training and then testing. Importantly the two texts do not have to be the same, as opposed to the use of predefined text in the fixed-text keystroke method, in which the enrolment text and log-in text must be identical [26].

Though the tasks included text that was chosen for the users to type, it is still considered free-text as the text used for training is not related at all to that used for testing [8]. Therefore, based on the definition of free-text [14], all text used in this paper was free-text but with a different method for sourcing the text. In fact, the results produced by the experiments carried out in [27] illustrates that using either un-copied or copied text has no effect on the results of free-text keystroke systems. Copied text was however provided for the participants to ease the process of data collection.

Users were directed to enter the samples in the most natural way possible, i.e. the same way they usually follow when typing. They were also allowed to enter carriage returns and backspaces if needed. Data collection was performed by a graphical user interface (GUI) programme implemented using the C++ language. The application was downloaded on the users' personal machines to maximise their comfort, on the basis that they are more familiar with their own machine and its surroundings. Therefore, they were able to feel more at ease, and thus, to perform the typing tasks in a manner closer to that of their real typing behaviour. Thus, an uncontrolled environment was adapted due to the fact that the data was collected wherever and whenever it was convenient to the user, thereby as much as possible providing a realistic representation of the normal conditions for the user.

Moreover, on observing the data collected in this experiment, a number of outliers were detected. Outlier data was identified to be as much as three standard deviations above or below the mean, as was suggested in [11]. These particularly very large or very small data points were discarded from the final data as it was considered that they represented noise that might affect the overall system performance.

In addition, it was seen as preferable to normalise the data before handing it to the machine learning technology [28]. Therefore, all the data was normalised to be between [0,1] to add a sense of uniformity to the data as otherwise attributes in greater numeric ranges might have dominated those in smaller numeric ranges [29].

The final step of data pre-processing involved creating the timing vector and storing it in the database as the user's profile. This process was carried out by combining the text from the two tasks and, then, dividing it into eight equal parts. Each one of the eight parts was considered as a single typing sample, the features from which were extracted and its mean calculated and stored separately. Therefore, eight samples per subject were included in the analysis phase for classifier training and testing.

Though there were 15 key-pairs, from which 75 timing features were captured, there were not enough instances that appeared in the used text for some of the key-pairs which made it unfeasible to include them in the final feature set, the omitted key-pairs were: NonAdjacent/RightSide, ThirdAdjacent/LeftSide, FourthAdjacent/LeftSide and NonAdjacent/LeftSide. In total 20 timing features were excluded from the final feature set; resulting in the inclusion of only 55 features in the final feature vector.

4.2 Experiment and results

After creating users' profiles, feature subset selection was performed using ACO [30] for each user's data. The selected features were then passed to the SVMs machine learning mechanism in order to be used as the basic data for differentiating between classes. The radial basis kernel multiclass SVMs classification process [31] was implemented on MATLAB with the aid of the LIBSVM library [32].

The DT technique, on the other hand, is capable of performing feature selection in the tree building phase [25]. Therefore, this was fed with the complete set of features. The Statistics toolbox in MATLAB was used to fit the tree and predict the class of each of the test data.

Classification, for both classifiers, was carried out through cross-validation which is a statistical sampling technique that aims to ensure that every example from the original dataset has the same chance of appearing in the training and testing set. We followed the leave-one-out cross-validation protocol which is a special case of the well-known n-fold cross-validation [33].

N-fold cross-validation divides the data up into n chunks and trains n times, treating a different chunk as the test sample each time; such that for each of n experiments, it uses $n-1$ folds for training and the remaining one for testing. Leave-one-out cross-validation is exactly the same except that all chunks contain only a single sample.

In our experiment, eight samples were used to perform eight cross-validation experiments. Seven of the samples were treated as the training sample set and the remaining sample was regarded as the testing sample. In each experiment, a different sample was selected to act as the test data.

Furthermore, two error rates were used to infer the performance: namely, false accept rate (FAR) and false reject rate (FRR). FAR is the percentage of impostors who have successfully gained access to the system, whereas FRR indicates the percentage of legitimate users who were denied access to the system [7]. In both cases, it is therefore desirable for these figures to be as low as possible.

SVMs have a slight performance advantage over DTs, as shown in Table 2. This is due to the fact that SVMs are more advanced in distinguishing between classes in similar situations [34]. The classification is done in SVMs by performing optimisation to find the separating hyperplane [35], whereas classification in DTs is purely based on rules [36]. It is worth noting that by using a more sophisticated subset selection mechanism such as ACO this has also contributed to the SVM's superior results [37]. The subset selection in DTs, on the other hand, is relatively primitive compared with ACO as it is carried out internally in the tree building stage [25].

The features selected by the ACO were AR-H2, TD-H1, TD-H2, TD-DD and FD-H2. Meanwhile, the features selected by DTs were: TR-UD, TD-H1, TD-DD and FD-DD. The two subsets are different except for two features: TD-H1and TD-DD. It is noted that the features selected by DTs are dominated by latency features whilst the features selected by ACO are dominated by duration features.

Table 2 System performance using Arabic input

	FAR	FRR
DT	0.205	0.512
SVMs	0.169	0.423

Table 3 System performance using English input

	FAR	FRR
DT	0.281	0.702
SVMs	0.245	0.613

This partially explains the lower error rate produced by SVMs classification. As found in [38], duration time appears to be a more reliable method to capture a user typing pattern compared with latency in systems concerned with user authentication.

4.3 Discussion and comparison of Arabic and English

In this paper, we performed the same experiment using English input on the same 21 subjects. The results of the English input experiment are shown in Table 3. Similar to Arabic input, SVMs proved to outperform DTs as they produced lower error rates. SVMs used features selected by ACO in the classification stage, these features were: AL-H1, AL-H2, AD-UU, AD-UD and SR-H2. Meanwhile, with DTs subset selection was performed in the tree building stage. The features chosen by DTs in the classification were: SL-UU, TL-DD, FL-UD, FD-DD and ND-DD. The features subset selected by the ACO consisted of both duration times and latency times, while the features selected by DTs were exclusively latency times. As with Arabic input, this contributed to the superiority of the SVMs system performance as duration time has been shown to produce better system performance [38].

It is worth mentioning that we noted a difference between the key-pairs in English and Arabic. In Arabic the number of key-pairs on the right-hand side of the keyboard is more than those on the left as most of the commonly used letters are on the right-hand side of the keyboard. English, on the other hand, has a greater number of the most used letters (letters such as *e*, *t*, *a*, *s* and *r*) on the left-hand side of the keyboard, thus key-pairs from that side are larger in number than those on the right-hand side. Owing to that, the key-pairs that were used to create the users' Arabic profile have some differences to those used for creating the user's English profile, as mentioned in Section 4.1. The key-pairs included in the English experiment can be found in previous research [19].

Moreover, Arabic input results were generally better than those based on English input due to the fact that all of the subjects chosen to be part of the experiments of this paper were native Arabic speakers and ten of the 21 subjects do not type in English regularly. As Arabic speakers are used to typing in Arabic, their typing skills have developed in Arabic and they are more familiar with an Arabic keyboard and how to operate it. This provides their typing with enough uniformity to be used to correctly identify the users based on their typing patterns. In addition, subjects who are not familiar with English typing have less familiarity with an English keyboard and typing in that language. Therefore, the absence of English experience has caused non-English natives to lack the typing consistency needed to identify users based on their typing patterns [39].

The same key-pairing approach has been used in our previous research for user authentication using English text [19], in which the SVMs/ACO method resulted in an FAR of 0.001 and an FRR of 0.504. The performance of that experiment outperformed the English experiment we performed in this paper for two reasons. First, the features selected in that experiment were: SR-H2, FL-H2, FD-H2, ND-H1 and ND-UD. Thus, the selected feature subset is dominated by duration times; this contributed to the superiority of the previous study system performance as duration time has better distinguishing capabilities in keystroke dynamics [38]. Moreover, nine of the 21 participants in the previous English experiment were native English speakers and the rest were very familiar with English and were used to typing in English on a daily basis. This clearly improves the consistency of the typing patterns of such users compared with the subjects in this new study, in which the majority were not familiar with English typing.

Both experiments provided results demonstrating the effect that the most commonly used language has on system performance. Lower error rates are achieved when the system uses the native language for most of the users or a language that most of the users are familiar with. In the experiment performed in this research, the use of Arabic language was shown to achieve higher performance as all of the subjects involved in the experiment were either native to Arabic or more familiar with Arabic. In contrast, in the previous research, the English experiment yielded good results due to the participants being either native to English or more familiar with English typing.

To compare the results we found from Arabic native speakers with results from English native speakers, we conducted a further experiment. In this test, we collected English data from eight English natives and analysed it in a manner similar to that used to analyse the Arabic data. Employing SVMs/ACO, system performances of 0.089 FAR and 0.125 FRR were achieved using English input. This result is considered satisfactory and it demonstrates that using the native language of the users affects the system performance positively. This agrees with the conclusion we made using Arabic native speakers' data. Yet more investigation is needed to achieve better understanding of the English/Arabic native speakers' typing differences and similarities.

5 Conclusions

In this paper, we examined the usefulness of applying free-text keystroke dynamics user authentication on Arabic text by an original keyboard's key-layout-based method. This key-pairing approach works by classifying every two characters typed consecutively based on their relation to each other and their overall location on the keyboard. For each key-pair, five timing features were extracted to be used in the user's feature vector.

SVMs and DTs were employed to classify individuals based on the proposed timing features. The experiment produced good results considering the fact that it used free-text for user authentication. Moreover, it accomplished user authentication based on the smallest amount of training possible. The FAR and FRR rates were both satisfactory with the FAR being the slightly better of the two.

This paper proved that the proposed method has been successful in authenticating users based on their Arabic typing. The method was originally created to be used with English typing, yet it has crossed over to Arabic input successfully. Moreover, in the comparative study SVMs produced lower error rates compared with DTs. Duration times also proved to contribute more in increasing the system performance when compared with the latency times.

In addition, experimenting with two languages showed that the user's familiarity with a certain language has a high impact on the user's typing patterns in that language. This considerably affects the system performance as lower error rates are produced from systems incorporating a language native or familiar to the users.

There is much more that can be done to improve this approach, one example of which is to expand the typing features to include other non-timing features such as typing speed, error rate and special keys usage patterns. Experimenting with different classification methods might also contribute positively to the overall system performance. Moreover, experimenting with other languages, which have a different alphabet to English such as Chinese or Thai can be carried out to understand how they compare with English and Arabic.

6 References

1 Monrose, F., Rubin, A.: 'Authentication via keystroke dynamics'. Proc. Fourth ACM Conf. on Computer and Communications Security, 1997, pp. 48–56

2 Gunetti, D., Picardi, C., Ruffo, G.: 'Keystroke analysis of different languages : a case study'. Proc. Advances in Intelligent Data Analysis 2005, vol. VI, no. 2, pp. 133–144

3 Awaida, S.M., Mahmoud, S.A.: 'State of the art in off-line writer identification of handwritten text and survey of writer identification of Arabic text', *Educ. Res. Rev.*, 2012, **7**, (20), pp. 445–463
4 Katzner, K.: 'The languages of the world' (Routledge Ltd., London, 1975)
5 Cooper, R.L., Olshtain, E., Tucker, G.R., *et al.*: 'The acquisition of complex English structures by adult native speakers of Arabic and Hebrew', *Lang. Learn.*, 1979, **29**, (2), pp. 255–275
6 Versteegh, K.: 'The Arabic language' (Edinburgh University Press, Edinburgh, 2014, 2nd edn.)
7 Karnan, M., Akila, M., Krishnaraj, N.: 'Biometric personal authentication using keystroke dynamics: a review', *Appl. Soft Comput.*, 2011, **11**, (2), pp. 1565–1573
8 Banerjee, S.P., Woodard, D.L.: 'Biometric authentication and identification using keystroke dynamics: a survey', *J. Pattern Recognit. Res.*, 2012, **7**, pp. 116–139
9 Bours, P., Mondal, S.: 'Performance evaluation of continuous authentication systems performance evaluation of continuous authentication systems', *IET Biometrics*, 2015, **4**, (4), pp. 1–7
10 Alsultan, A., Warwick, K.: 'Keystroke dynamics authentication: a survey of free-text methods', *Int. J. Comput. Sci. Issues*, 2013, **10**, (4), pp. 1–10
11 Joyce, R., Gupta, G.: 'Identity authentication based on keystroke latencies', *Commun. ACM*, 1990, **33**, (2), pp. 168–176
12 Lau, E., Liu, X., Xiao, C., *et al.*: 'Enhanced user authentication through keystroke biometrics'. Proc. Computer and Network Security, 2004
13 Lv, H.R., Lin, Z.L., Yin, W.J., *et al.*: 'Emotion recognition based on pressure sensor keyboards'. Proc. IEEE Int. Conf. Multimedia and Expo, 2008, pp. 1089–1092
14 Gunetti, D., Picardi, C.: 'Keystroke analysis of free text', *ACM Trans. Inf. Syst. Sec.*, 2005, **8**, (3), pp. 312–347
15 Singh, S., Arya, K.V.: 'Key classification: a new approach in free text keystroke authentication system'. Proc. Third Pacific-Asia Conf. on Circuits, Communications and System, 2011, pp. 1–5
16 Hu, J., Gingrich, D., Sentosa, A.: 'A K-nearest neighbor approach for user authentication through biometric keystroke dynamics'. Proc. IEEE Int. Conf. on Communications, 2008, pp. 1556–1560
17 Raghu, D., Jacob, C.R., Bhavani, Y.V.K.D.: 'Neural network based authentication and verification for web based key stroke dynamics', *Int. J. Comput. Sci. Inf. Technol.*, 2011, **2**, (6), pp. 2765–2772
18 Samura, T., Nishimura, H.: 'Keystroke timing analysis for individual identification in Japanese free text typing'. Proc. ICCAS-SICE, 2009, pp. 3166–3170
19 Alsultan, A., Warwick, K., Wei, H.: 'Feature subset selection for free-text keystroke dynamics' (John Wiley & Sons, NJ, USA, 2015), in press
20 Malas, T.M., Taifour, S.S., Abandah, G.A.: 'Toward optimal Arabic keyboard layout using genetic algorithm'. Proc. Ninth Int. Middle Eastern Multiconference on Simulation and Modeling (MESM 2008), 2008, pp. 50–54
21 Teh, P.S., Teoh, A.B.J., Ong, T.S., *et al.*: 'Statistical fusion approach on keystroke dynamics'. Proc. Third Int. IEEE Conf. on Signal-Image Technologies and Internet-Based System, 2007, pp. 918–923
22 Saeys, Y., Inza, I., Larrañaga, P.: 'A review of feature selection techniques in bioinformatics', *Bioinformatics*, 2007, **23**, (19), pp. 2507–2517
23 Burges, C.J.C.: 'A tutorial on support vector machines for pattern recognition', *Data Min. Knowl. Discov.*, 1998, **2**, (2), pp. 121–167
24 Shanmugapriya, D., Padmavathi, G.: 'An efficient feature selection technique for user authentication using keystroke dynamics', *Int. J. Comput. Sci. Netw. Secur.*, 2011, **11**, (10), pp. 191–195
25 Russell, S., Norvig, P.: 'Artificial intelligence: a modern approach' (Prentice-Hall, Englewood Cliffs, 2010, 3rd edn.)
26 Teh, P.S., Teoh, A.J., Yue, S.: 'A survey of keystroke dynamics biometrics', *Sci. World J.*, 2013, pp. 1–24
27 Killourhy, K.S., Maxion, R.A.: 'Free vs. transcribed text for keystroke-dynamics evaluations'. Proc. ACM Int. Conf. Proceeding Series, 2012, pp. 1–8
28 Hsu, C.W., Chang, C.C., Lin, C.J.: 'A practical guide to support vector classification'. Technical Report, Department of Computer Science, National Taiwan University, 2003
29 Kantardzic, M.: 'Data mining: concepts, models, methods, and algorithms' (2011, 2nd edn.)
30 Jensen, R.: 'Performing feature selection with ACO', *Swarm Intell. Data Min.*, 2006, **34**, pp. 45–73
31 Ambwani, T.: 'Multi class support vector machine implementation to intrusion detection'. Proc. the Int. Joint Conf. on Neural Networks, 2003, vol. 3, pp. 2300–2305
32 Chang, C.C., Lin, C.J.: 'LIBSVM: a library for support vector machines'. 2001, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
33 Refaeilzadeh, P., Tang, L., Liu, H.: 'Cross-validation', *Encycl. Database Syst.*, 2009, pp. 532–538
34 Caruana, R., Niculescu-Mizil, A.: 'An empirical comparison of supervised learning algorithms'. Proc. 23th Int. Conf. on Machine Learning, 2006, pp. 161–168
35 Fletcher, T.: 'Support Vector Machines Explained', pp. 1–19, 2009. Available at http://www.sutikno.blog undip.ac.id/files/2011/11/SVM-Explained. pdf., accessed 06 June 2013
36 Safavian, S.R., Landgrebe, D.: 'A survey of decision tree classifier methodology', *IEEE Trans. Syst. Man Cybern.*, 1991, **21**, (3), pp. 660–674
37 Dorigo, M., Stützle, T.: 'Ant colony optimization' (Massachusetts Institute of Technology, Cambridge, Massachusetts, 2004)
38 Robinson, J.A., Liang, V.M., Chambers, J.A.M., *et al.*: 'Computer user verification using login string keystroke dynamics', *IEEE Trans. Syst. Man Cybern. A, Syst. Humans*, 1998, **28**, (2), pp. 236–241
39 Thorson, H.: 'Using the computer to compare foreign and native language writing processes: a statistical and case study approach', *Mod. Lang. J.*, 2000, **84**, (2), pp. 155–169

Feature Selection: A Comparison Study in Free-text Keystroke Dynamics Authentication

Arwa Alsultan *, Hong Wei *, Kevin Warwick **
* School of Mathematical, Physical and Computational Sciences, University of Reading, Reading, UK
** Vice Chancellors Office, Coventry University, Coventry, UK

Corresponding Author:

Arwa Alsultan,
Department of Computer Science, School of Mathematical, Physical and Computational Sciences, University of Reading, Reading, RG6 6AH, UK
Email: a.f.a.alsultan@pgr.reading.ac.uk

ABSTRACT

In this paper, two kinds of typing features are extracted for free-text keystroke dynamics authentication. These features are: timing features and non-conventional features. Feature selection is a vital process in the success of applying free-text keystroke dynamics in individuals' authentication. In this study, Ant Colony Optimization (ACO), an optimization algorithm, is implemented to perform feature subset selection. The selected features are used for user authentication by Support Vector Machines (SVMs). A comparison is made between the selected feature sets of the two sets of features. The selected subset of non-conventional features succeeded in producing a better overall system performance. The non-conventional features provided the appropriate balance between imposters' accessing the system (false accept rate) and legitimate users denied access to the system (false reject rate) in this study. It is believed that the selected features represent users' typing pattern to a great extent.

Keyword:

Free-text

Keystroke dynamics

Feature selection

ACO

SVMs

1. INTRODUCTION

Keystroke dynamics is an effortless behavior-based method for authenticating users, which employs the person's typing patterns for validating his/her identity. As mentioned in [1], keystroke dynamics is "not what you type, but how you type." In this approach, the user types- in text, as usual, without any extra work to be done for authentication. Moreover, it only involves the user's own keyboard and no other external hardware. These criteria make keystroke dynamics an excellent alternative or addition to the more conventional ID/password authentication scheme.

There are two basic classes of keystroke dynamics, namely: fixed-text and free-text [2]. The fixed-text keystroke dynamics method uses the typing pattern of the user while entering a predefined text. This text has been previously used to train the system and is delivered by the user at log-in time. Contrariwise, the free-text keystroke method is considered easier for the user as it overcomes the problem of memorizing the text, something that fixed-text keystrokes suffers from. As its name suggests, in free-text keystrokes, the text

used for enrolment does not have to be the same as the text used for log-in. Moreover, free-text keystroke dynamics is used for enhancing security through continuous and nonintrusive authentication [3]. Thus, the latter method is the one that has been considered in this paper as it can be applied in many useful settings to aid real life situations in addition to the benefit it provides in balancing between security and usability [2].

Existing research in the literature of keystroke dynamics primarily has focused on exploiting timing features for authentication purposes. These features include: typing latency [4], and keystroke duration [1]. Other research has included different kinds of typing features such as typing speed and shift key usage patterns [5], and error rate [6].

In this study we consider in depth both the timing features and the non-conventional features and investigate how to apply feature subset selection to both feature sets. This is done for the purpose of comparing the two feature sets and their ability to differentiate between users. For that purpose, Ant Colony Optimization (ACO) is utilized for feature selection in both features sets. Moreover, Support Vector Machines (SVMs) is used for classification.

The feature subset selection mechanism is considered a very important step in applications such as keystroke dynamics authentication as it aids in choosing features that most represent the user's typing patterns and eliminate any redundancy or noise that can affect the authentication process negatively. ACO was utilized in keystroke dynamics in several studies. An example of the studies utilizing ACO, together with other feature selection techniques, is the one performed in [7]. In addition to ACO, Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) were applied to the data before feeding it into a back propagation neural network (BPNN) classifier. Based on feature reduction rate and classification accuracy, this study proved that ACO yields better performance than PSO and GA.

Moreover, while ACO, PSO and GA were all used in [8] for feature subset selection, the Extreme Learning Machine (ELM) was chosen to be the learning method. Supportive of the conclusions found in [7], this work demonstrated that ACO results in the best feature subset selection with ELM.

The rest of this paper proceeds as follows. Section 2 introduces the feature sets used in this study. It also describes the Ant Colony Optimization (ACO) idea and how it works. In Section 3 we present experimental results with discussion, in which the data space is indicated. The final section concludes the topic and points out our research contributions and future work.

2. RESEARCH METHOD

In this section the features extracted for authentication are discussed. In addition, the feature subset selection algorithm, i.e. ACO, is also explained.

2.1. Keystroke Features

Both timing features and non-convenational typing features are discussed in this section.

2.1.1. Timing Features

The timing features used in this study are extracted between two keys (key-pair) that are pressed consecutively and have a relationship on the keyboard layout. This relationship depends mainly on the key

position of each character on the keyboard in relation to the other characters. Moreover, these relationships can vary depending on the location of the two keys with respect to the overall keyboard layout.

There are five categories for key-pair relationships:

1. Adjacent: keys located next to each other on the keyboard.
2. Second adjacent: keys that are one key apart from each other.
3. Third adjacent: keys that are two keys apart.
4. Fourth adjacent: keys that are three keys apart.
5. None adjacent: keys that are more than three keys apart.

An example is provided in Figure 1 demonstrating the key relationship concept; while considering the key 'G'.

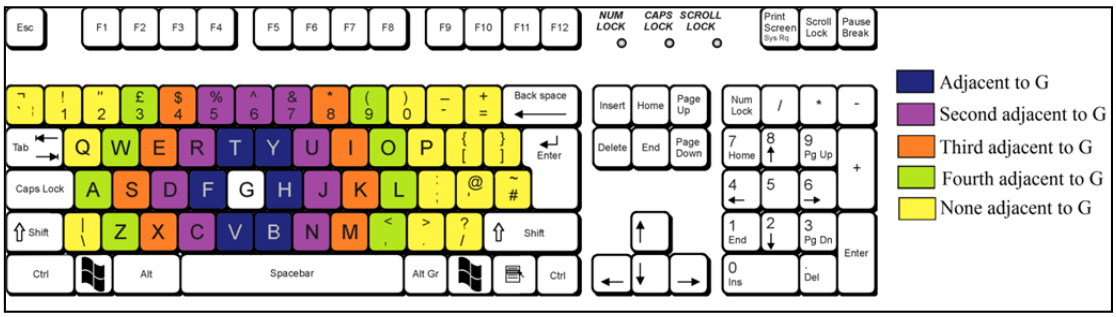

Figure 1. Key-pair classification

Each of these categories can fall into one of the following overall locations:

1. Both keys are on the right hand side of the keyboard.
2. Both keys are on the left hand side of the keyboard.
3. The two keys are located on different sides of the keyboard, i.e. the first key is located on the right hand side while the second key is on the left or vice versa.

The text is broken down into di-graphs, or key-pairs, and each di-graph typed on the main part of the keyboard is classified using the above categories and locations. In total, there are fifteen different combinations of key-pairs that any two keys can be classified into. Based on the previously explained technique, the key-pair "vr" is categorized as: SecondAdjacent/LeftSide.

Once the key-pairs have been obtained from the users' raw data, the keystroke features are extracted [9]. These features were computed for every key and key-pair using two main values, specifically: the press time (D_n) and the release time (U_n) of each key (n) in milliseconds. These features are (as shown in Figure 2):

1. Hold time (also called Dwell time or keystroke duration): is the time a key is pressed until it is released. Consequently, each key-pair has two hold times:
    a. Hold time for the first key (H1).
    b. Hold time for the second key (H2).
2. Keystroke latencies (also called Flight Time): There are three types of latencies:
    a. Down-Down (DD) (also called Press-Press (PP) time): is the interval time between two successive key presses.

b. Up-UP (UU) (also called Release-Release (RR) time): is the interval time between two successive key releases.

c. Up-Down (UD) (also called Release-Press (RP) time): is the interval time between a key release and the next key press.

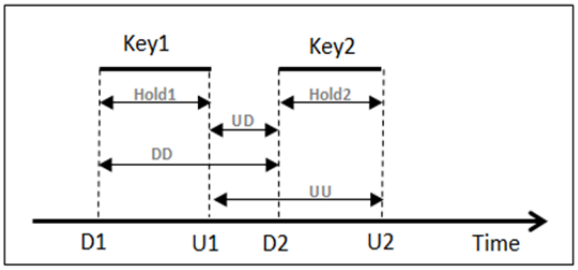

Figure 2. Timing features for a key-pair

Thus, five timing features were defined for each key-pair appearance in the text. This was done for all fifteen types of key-pairs. Therefore, the overall number of timing features was 75 (5 timing features * 15 key-pairs). Table 1 lists all the 75 features extracted from all key-pairs. The features abbreviations listed in the table combine the key-pair category and the timing feature, for example: "AR-H1" stands for: Adjacent/RightSide-Hold1 and so on.

Table 1. Overview of the timing features

Key-pair Category	Feature Set				
Adjacent/RightSide	AR-H1	AR-H2	AR-DD	AR-UU	AR-UD
Adjacent/LeftSide	AL-H1	AL-H2	AL-DD	AL-UU	AL-UD
Adjacent/DifferentSide	AD-H1	AD-H2	AD-DD	AD-UU	AD-UD
SecondAdjacent/RightSide	SR-H1	SR-H2	SR-DD	SR-UU	SR-UD
SecondAdjacent/LeftSide	SL-H1	SL-H2	SL-DD	SL-UU	SL-UD
SecondAdjacent/DifferentSide	SD-H1	SD-H2	SD-DD	SD-UU	SD-UD
ThirdAdjacent/RightSide	TR-H1	TR-H2	TR-DD	TR-UU	TR-UD
ThirdAdjacent/LeftSide	TL-H1	TL-H2	TL-DD	TL-UU	TL-UD
ThirdAdjacent/DifferentSide	TD-H1	TD-H2	TD-DD	TD-UU	TD-UD
FourthAdjacent/RightSide	FR-H1	FR-H2	FR-DD	FR-UU	FR-UD
FourthAdjacent/LeftSide	FL-H1	FL-H2	FL-DD	FL-UU	FL-UD
FourthAdjacent/DifferentSide	FD-H1	FD-H2	FD-DD	FD-UU	FD-UD
NonAdjacent/RightSide	NR-H1	NR-H2	NR-DD	NR-UU	NR-UD
NonAdjacent/LeftSide	NL-H1	NL-H2	NL-DD	NL-UU	NL-UD
NonAdjacent/DifferentSide	ND-H1	ND-H2	ND-DD	ND-UU	ND-UD

2.1.2. Non-conventional Features

The non-conventional features used here include two types of typing features, namely: semi-timing features and editing features. Both categories are explained in this section.

2.1.2.1. Semi-timing Features.

Semi-timing features are different from the standard timing features, as the time calculation followed in this category is slightly different from that of the regular timing features. These features have a collective property, as all of them are calculated during longer periods of time. The features included in this category are:

1. Word-per-Minute (WPM): measures the user's average typing speed.
2. Negative Up-Down percentage (negUD): measures the percentage of negative Up-Down actions detected in the user's typing stream. Negative Up-Down is due to the overlap happening between two successive keys being typed. This particular typing behavior is found in the typing stream of users who have the tendency to press the second key before releasing the first one. Figure 3 illustrates a negUD caused by two-key sequences overlapping.
3. Negative Up-Up percentage (negUU): measures the percentage of negative Up-Up actions detected in the user's typing stream. Negative Up-Up occurs when the typist tends to release the second key before releasing the first key. Negative UU only happens when there is a negative UD between the two successive keys.

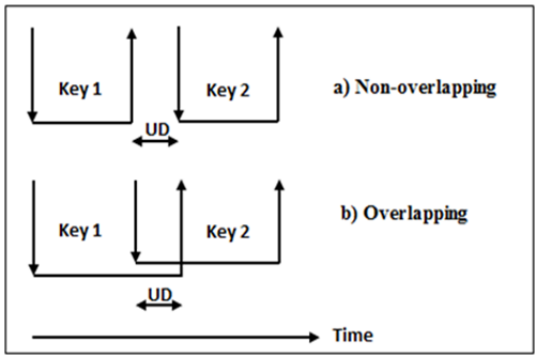

Figure 3. Negative UD caused by overlapping keystroke events

2.1.2.2. Editing Features.

Editing features does not give any attention to the time a user spends typing, rather it considers the way a user goes about the process of typing. The features included in this category are:

1. Error rate: captures the percentage of times a user performs a typing error.
2. Capital letters incorporation practices: this subcategory is concerned with the specific ways a capital letter is included in the user's typing stream. This is usually done using either Caps Lock key or shift key. It has been noted that if a user has the habit of using the CapsLock key, then he or she will hardly ever use the shift key for capitalizing letters, and vice versa. Therefore, using these two attributes simultaneously might lead to a better understanding of the user's editing habits. Thus, the following features are derived:
    a. CapsLock usage: calculates the percentage of the CapsLock keys being used to produce capital letters in a given typing task.
    b. Shift Key usage: this subcategory has two different aspects of user's habits. The first shift key usage attribute is the right/left shift key choice. Some users use strictly the right shift or

strictly the left shift and others alternate between the two [5]. The second attribute is the order of which the shift/letter keys are released. The shift key is always pressed before the letter key if the user is intending to produce a capital version of that letter. However, there are two orders that users go about when releasing those keys, they either release the letter key before releasing the shift key or they release the letter key after releasing the shift key. Based on the previous observations, four different features that combine the two aspects of shift key usage are used:

i. Percentage of Right Shift released after letter (RSA).
ii. Percentage of Right Shift released before letter (RSB).
iii. Percentage of Left Shift released after letter (LSA).
iv. Percentage of Left Shift released before letter (LSB).

Table 2 gives an overview of all the nine non-conventional typing features used.

Table 2. Overview of the non-conventional typing features.

Category	Features
Semi-Timing Features	WPM
	negUD
	negUU
Editing Features	Error Rate
	CapsLock Usage
	RSA
	RSB
	LSA
	LSB

2.2. Ant Colony Optimization (ACO)

Having a large feature set means more computational time in addition to raising the complexity of the classification process. Therefore, it is necessary to incorporate a feature subset selection mechanism in cases where the feature set needs more examining [10]. Feature subset selection works by scrutinizing all possible features to recognize the feature or set of features that produce optimal or near-optimal performance, i.e. those that minimize classification error [11].

ACO is used in this study as it is one of the most successful mechanisms of swarm intelligence used for feature subset selection. It is an optimization technique that was inspired by the indirect communication between real ants using chemical pheromones that they leave on trails, which permits them to find the shortest path between the nest and the food supply [12].

The transition rule for any ant 'm' that allows it to decide on including the i^{th} feature at any time ‘t’ in the solution is influenced by two aspects: the heuristic desirability (η_i) and the level of pheromone (τ_i) [13]. Often a classifier performance is used as heuristic information for feature selection.

The probabilistic transition rule is calculated in (1) as follows:

$$P_i^k(t) = \begin{cases} \frac{\tau_{i(t)}{}^{\alpha} * \eta_i{}^{\beta}}{\sum_{j \epsilon h^k} \tau_{j(t)}{}^{\alpha} * \eta_i{}^{\beta}} & \text{if } i \epsilon h^k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Where h^k is the set of feasible features that can be added to the partial solution; the two parameters α and β are used to control the relative importance of the pheromone value and heuristic information.

The process of pheromone evaporation on all nodes is activated after all ants have completed their solutions. The evaporation rate is shown in (2).

$$\Delta\tau_i^k(t) = \begin{cases} \varphi * \frac{C\left(S^k(t)\right)}{|S^k(t)|} + (1-\varphi) * \frac{(N - |S^k(t)|)}{N} & \text{if } i \epsilon S^k(t) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Where Sk(t) is the feature subset found by ant k at iteration t, and |Sk(t)| is its length while C(Sk(t)) is the classifier performance for that ant at that iteration. N is the total number of features in the data set and the parameter φ controls the relative weight which dictates the importance of the classifier performance and the feature subset length.

Afterwards, each ant k deposits a specific quantity of pheromone on each node that it has navigated. Equation (3) shows the pheromone update for all ants, which also includes the effect of evaporation:

$$\tau_i(t) = (1-\rho) * \tau_i(t) + \sum_{k=1}^{m} \Delta\tau_i^k(t) \quad (3)$$

Where ρ is the pheromone trail decay coefficient which ranges from 0 to 1 and m is the number of ants.

The stopping criterion for feature selection has been targeted in numerous ways such as using a fixed number of features, in which the user defines the minimum and maximum limit for the feature subset length [14]. The overall process of ACO feature selection is shown in Figure 4.

3. EXPERMENTS, RESULTS AND ANALYSIS

The data space used in this study is indicated in this section. Moreover, the expermintal results are also discussed in this section together with a discussion about the outcomes of the experiment.

3.1. Data Space

A total of twenty-five users participated in this study. During data collection, the participants were asked to perform eight typing tasks each of which consisted of around 1000 characters. The text included both upper and lower case letters in addition to numbers and punctuation marks. Furthermore, the data was acquired in different sessions as the users were requested to complete each of the eight tasks in a separate session. Users were directed to enter the samples in the most natural way possible, i.e. the same way they usually follow when typing.

The data collection was performed on a GUI program implemented using the C++ language. The application was downloaded on the users' personal machines to maximize their comfort as they are more familiar with their own machine and its surroundings. Therefore, they were able to feel more at ease, and thus to perform the typing tasks in a manner closer to that of their real typing behavior.

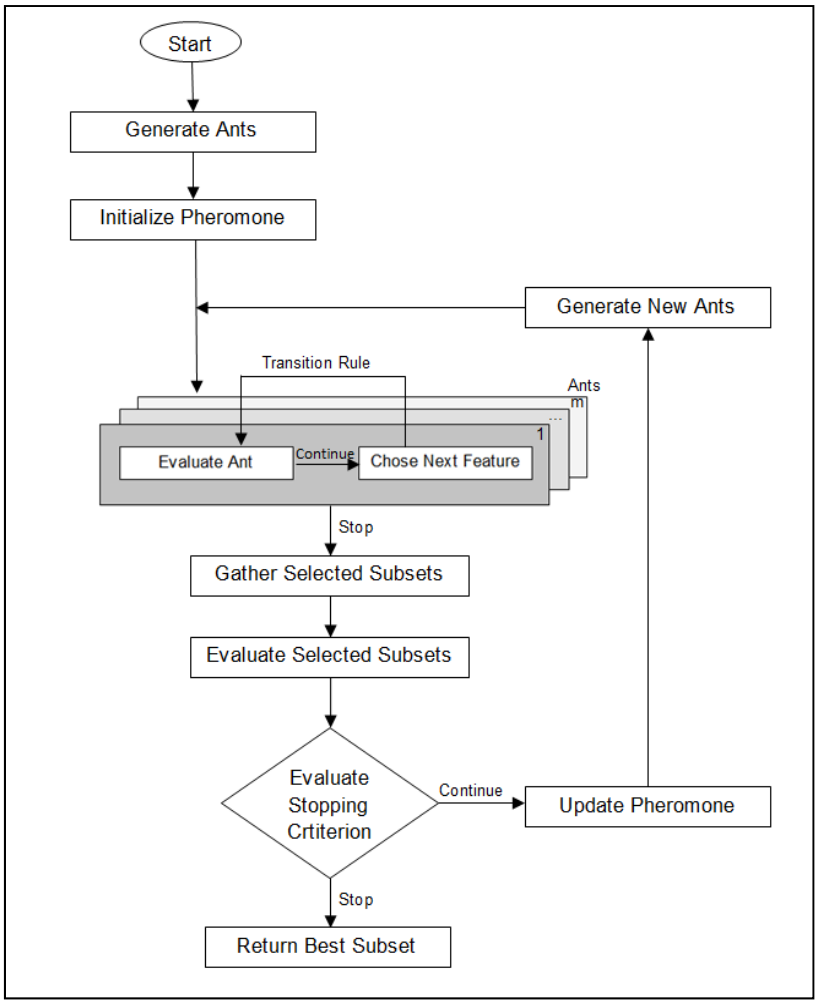

Figure 4. ACO feature selection process.

Although there were 75 timing features captured from each user's typing stream, there were only 55 having enough instances appearing in the used text. Therefore 20 timing features were excluded from the final feature set. The discarded features are shaded in Table I. Thus, a total of 55 timing features were considered in the experiments.

Outlier elimination was performed on the timing features data set. Outlier data has been identified to be as much as three standard deviations above or below the mean as suggested in [4]. These particularly very large or very small data points were discarded from the final data as they were deemed to represent noise that might affect the overall system performance.

In addition, timing data was normalized before being handed to the machine learning technology [15]. Therefore, all the data was normalized between [0,1] to add a sense of uniformity to the data as attributes in greater numeric ranges might otherwise have dominated those in smaller numeric ranges [16].

The final step of data pre-processing for the timing features involved creating the timing vector and storing it in the database as the user's profile. This was done by computing the mean of each timing feature and storing it in the timing features vector. This process was carried-out by considering each one of the eight typing tasks as a single typing sample.

For non-conventional typing features, there was no need for outlier discarding as the features did not rely on a time factor that might add noise in the form of too large or too small time lags. Moreover, no scaling was needed as all non-conventional features were percentages in the [0, 1] range. Lastly the non-conventional typing features were calculated and stored in the non-conventional features vector at the database as the user's profile. Similar to the timing features, each one of the eight typing tasks was considered as a single typing sample.

3.2. Experiment and Results

In the experimentation stage, ACO was implemented by generating a total number of ants which is equal to the number of features. A total of 55 and 9 ants were generated for timing and non-conventional features, respectively. In addition, an initial pheromone of 1.0 was used and as the importance of the heuristic information, i.e. classification rate, was more than that of the pheromone level in the transition rule, we used α=1.0 and β=0.1. For both timing and non-conventional features, the features selected by the ACO were fed into SVMs in order to be classified. A Radial Basis Kernel multiclass SVMs was used [17]. The classification process was implemented on MATLAB with the aid of the LIBSVM library [18].

Both feature subsets were classified through cross-validation which is a statistical sampling technique that aims to ensure that every example from the original dataset has the same chance of appearing in the training and testing set [19]. N-fold cross-validation divides the data up into n chunks and trains it n times, treating a different chunk as the test sample each time; such that for each of n experiments, it uses n-1 folds for training and the remaining one for testing.

In our experiment, eight samples were used to perform eight cross-validation experiments. Seven of the samples were treated as the training sample set and the remaining sample was regarded as the testing sample. In each experiment, a different sample was selected to act as the test data.

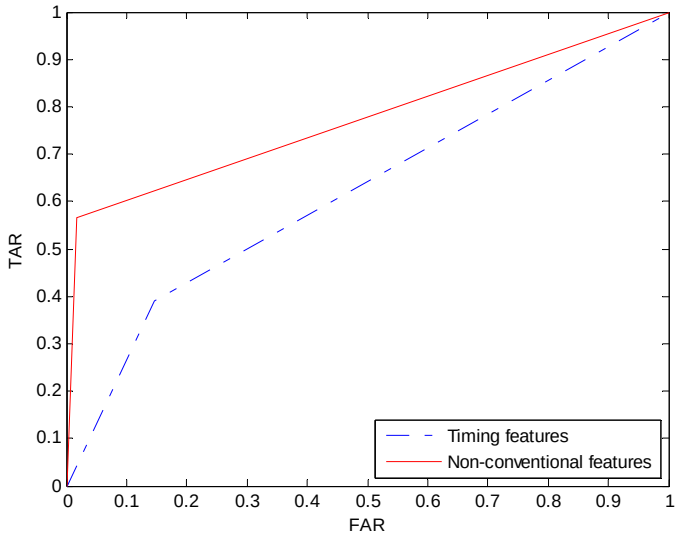

Figure 5. ROC curve of the timing features and the non-conventional features.

Furthermore, two error rates were used to infer the performance, namely: False Accept Rate (FAR) and False Reject Rate (FRR). FAR is the percentage of impostors who have successfully gained access to the

system whereas FRR indicates the percentage of legitimate users who were denied access to the system [20]. The error rates resulting from the timing features and the non-conventional features are shown in Table 3.

Table 3. Timing features and non-conventional features error rates

Features	FAR	FRR
Timing	0.147	0.61
Non-conventional	0.018	0.435

Conclusively, non-conventional features have a considerable performance advantage over SVMs (as shown in Figure 5). They produced a higher accuracy system as the ROC is plotted closer to the upper left corner of the diagram.

3.3. Discussion

Using all features with no feature selection did not produce particularly good results as the level of noise was larger in such high features dimensionality [10]. However, when looking more closely at the features selected to be utilized in both feature sets (shown in Table 4), we noticed that the majority of the selected timing features were duration features. Furthermore, the non-conventional features subset is dominated by editing features. This denotes the significance of these two kinds of features in the area of user authentication by free-text keystroke dynamics.

Table 4. Timing features and non-conventional selected subsets

Timing features	Non-conventional features
SR-H2 | FL-H2 | FD-H2 | ND-H1 | ND-UD	negUD | Error Rate | RSB | LSA | LSB

The FAR produced by non-conventional features is very good, but the FRR is not as satisfactory. The FAR and FRR figures in the case of non-conventional features denotes the fact that there is hardly any imposters accessing the system. Yet there were a number of legitimate users that were denied access. In timing features, on the other hand, both FAR and FRR are not satisfactory as a considerable number of imposters were granted access and more than half of the legitimate users were denied access.

These results support the belief that non-conventional features represent human typing patterns more precisely compared with timing features in free-text keystroke systems. Non-conventional features appear to have a strong relationship between input values and target values, in this data set. A strong input-target relationship is formed when knowledge of the value of an input improves the ability to predict the value of the target which helps in understanding the characteristics of the target [21].

CONCLUSION

In this paper, a comparison between timing features and non-conventional features has been presented to improve the overall understanding of free-text keystroke dynamics authentication. A subset of each feature set was selected based on ACO. The subsets were then used for user classification with the aid of SVMs.

It has been noted that non-conventional typing features have outperformed the timing features as the subset selected from non-conventional features provided a better recognition rate in this study. The utilization of the non-conventional features subset produced a system that have a better balance between the number of legitimate users denied access and the number of imposters accessing the system.

Feature subset selection is considered crucial in the process of machine learning, particularly in the field of keystroke dynamics authentication. ACO, as one of the most successful mechanisms of swarm intelligence used in optimization, can perform the feature selection in a more natural way.

There is much more that can be done to improve this study. One example of which is to investigate other feature selection techniques and classification methods. Experimenting with more key-pairing methods for timing features and more non-conventional features could also be explored to contribute to the overall system performance.

ACKNOWLEDGEMENTS

The authors wish to extend their gratitude to the participants who were involved in this experiment for the time they took out of their busy schedules to contribute in this study.

REFERENCES

[1] F. Monrose and A. Rubin, "Authentication via keystroke dynamics," in *4th ACM conference on Computer and communications security*, 1997, pp. 48–56.

[2] A. Alsultan and K. Warwick, "Keystroke Dynamics Authentication : A Survey of Free-text Methods," *International Journal of Computer Science Issues*, vol. 10, no. 4, pp. 1–10, 2013.

[3] S. P. Banerjee and D. L. Woodard, "Biometric Authentication and Identification using Keystroke Dynamics : A Survey," *Journal of Pattern Recognition Research*, vol. 7, no. 1, pp. 116–139, 2012.

[4] R. Joyce and G. Gupta, "Identity authentication based on keystroke latencies," *Communications of the ACM*, vol. 33, no. 2, pp. 168–176, Feb. 1990.

[5] E. Lau, X. Liu, C. Xiao, and X. Yu, "Enhanced User Authentication Through Keystroke Biometrics," in *Computer and Network Security*, 2004.

[6] K. Hempstalk, E. Frank, and I. H. Witten, "One-class classification by combining density and class probability estimation," in *the European Conference on Machine and Learning and Principles and Practice of Knowledge Discovery in Database*, 2005, pp. 505–519.

[7] M. Karnan and M. Akila, "Personal Authentication Based on Keystroke Dynamics Using Soft Computing Techniques," in *Second International Conference on Communication Software and Networks*, 2010, pp. 334–338.

[8] D. Shanmugapriya and G. Padmavathi, "An Efficient Feature Selection Technique for User Authentication using Keystroke Dynamics," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 11, no. 10, pp. 191–195, 2011.

[9] P. S. Teh, A. B. J. Teoh, T. S. Ong, and H. F. Neo, "Statistical Fusion Approach on Keystroke Dynamics," *2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, pp. 918–923, Dec. 2007.

[10] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, Oct. 2007.

[11] M. Karnan, M. Akila, and N. Krishnaraj, "Biometric personal authentication using keystroke dynamics: A review," *Applied Soft Computing*, vol. 11, no. 2, pp. 1565–1573, Mar. 2011.

[12] M. Dorigo, A. Colorni, and V. Maniezzo, "The Ant System: An Autocatalytic Optimizing Process," *Tech. Rep. 91-016, Université Libre de Bruxelles, Milano, Italy*, 1991.

[13] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical Computer Science*, pp. 243 – 278, 2005.

[14] S. Nemati, M. E. Basiri, N. Ghasem-Aghaee, and M. H. Aghdam, "A novel ACO–GA hybrid algorithm for feature selection in protein function prediction," *Expert Systems with Applications*, vol. 36, pp. 12086–12094, 2009.

[15] C. W. Hsu, C. C. Chang, and C. J. Lin, "A practical guide to support vector classification," *Tech. Rep., Department of Computer Science, National Taiwan University*, 2003.

[16] R. A. Irizarry, B. Hobbs, Y. D. Beazer-barclay, K. J. Antonellis, U. W. E. Scherf, and T. P. Speed, "Exploration , normalization , and summaries of high density oligonucleotide array probe level data," *Biostatistics*, vol. 4, no. 2, pp. 249–264, 2003.

[17] K. Warwick and R. Craddock, "An introduction to Radial Basis Functions for system identification a comparison with other neural network methods .," in *35th IEEE International Conference on Decision and Control*, 1996, no. December, pp. 464–469.

[18] C. Chang and C. Lin, "LIBSVM : A Library for Support Vector Machines," pp. 1–39, 2013.

[19] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-Validation," *Encyclopedia of Database Systems*, pp. 532–538, 2009.

[20] D. Gunetti and C. Picardi, "Keystroke analysis of free text," *ACM Transactions on Information and System Security*, vol. 8, no. 3, pp. 312–347, Aug. 2005.

[21] B. D. Ville and P. Neville, *Decision trees for analytics using SAS enterprise miner*. SAS Institute, 2013.

BIBLIOGRAPHY OF AUTHORS

Arwa Alsultan	Arwa Alsultan is pursuing a PhD degree in Computer Science from the School of Mathematical, Physical and Computational Sciences at the University of Reading, Reading, Berkshire, UK. She completed her Master's degree in Computer Science from the Computer and Information Science College at the King Saud University, Riyadh, SA in 2010. She works as a lecturer at the IT Department in the Computer and Information Science College at the King Saud University, Riyadh, SA.
Hong Wei	Hong Wei received her first and Master degrees from Tianjin University, China, in 1983 and 1986, respectively, and PhD degree from Birmingham University, UK, in 1996. She is author/co-author of over 60 refereed research papers and two text books. Her research areas cover biometrics, image-based pattern classification, and computer vision.
Kevin Warwick	Kevin Warwick was born in Coventry. He received the B.Sc. degree from Aston University, Birmingham, U.K. in 1979, the Ph.D. and D.Sc. degrees from Imperial College, London, U.K. in 1982 and 1994 respectively and the D.Sc. degree from Czech Academy of Sciences, Prague, Czech Republic. He is Deputy Vice-Chancellor (Research) at Coventry University. His research areas are artificial intelligence, biomedical engineering and robotics. He has over 600 publications to his name. He has received Honorary Doctorates from nine Universities. He received The Future of Health Technology Award in the Massachusetts Institute of Technology, Cambridge, MA, USA; the IEE Senior Achievement Medal in 2004; the Mountbatten Medal in 2008; the Golden Eurydice in 2009; and the Ellison-Cliffe Medal in 2011.

Pattern Recognition Letters

Authorship Confirmation

Please save a copy of this file, complete and upload as the "Confirmation of Authorship" file.

As corresponding author I, Arwa Alsultan, hereby confirm on behalf of all authors that:

1. This manuscript, or a large part of it, has not been published, was not, and is not being submitted to any other journal.
2. If presented at or submitted to or published at a conference(s), the conference(s) is (are) identified and substantial justification for re-publication is presented below. A copy of conference paper(s) is(are) uploaded with the manuscript.
3. If the manuscript appears as a preprint anywhere on the web, e.g. arXiv, etc., it is identified below. The preprint should include a statement that the paper is under consideration at Pattern Recognition Letters.
4. All text and graphics, except for those marked with sources, are original works of the authors, and all necessary permissions for publication were secured prior to submission of the manuscript.
5. All authors each made a significant contribution to the research reported and have read and approved the submitted manuscript.

Signature Arwa Alsultan Date 15/09/2015

List any pre-prints:

Relevant Conference publication(s) (submitted, accepted, or published):

Justification for re-publication:

Graphical Abstract (Optional)

To create your abstract, type over the instructions in the template box below.
Fonts or abstract dimensions should not be changed or altered.

Leave this area blank for abstract info.

Research Highlights (Required)

To create your highlights, please type over the instructions in the template box below:

- Non-conventional features are able to authenticate users using free-text keystrokes
- Non-conventional features produce lower error rates compared with timing features
- Decision trees produce better system performance compared with SVMs

Pattern Recognition Letters
journal homepage: www.elsevier.com

Non-Conventional Keystroke Dynamics for User Authentication

Arwa Alsultan[a]*, Kevin Warwick[b] and Hong Wei[a]

[a]*Department of Computer Science, School of Mathematical, Physical and Computational Sciences, University of Reading, Reading RG6 6AH, UK*
[b]*Vice Chancellors Office, Coventry University, Priory Street, Coventry CV1 5FB, UK*

ABSTRACT

This paper introduces an approach for user authentication using free-text keystroke dynamics which incorporates the use of non-conventional keystroke features. Semi-timing features along with editing features are extracted from the users' typing stream. Decision trees were exploited to classify each of the users' data. In parallel for comparison, Support Vector Machines (SVMs) were also used for classification in association with an Ant Colony Optimization (ACO) feature selection technique. The results obtained from this study are encouraging as low False Accept Rates (FAR) and False Reject Rates (FRR) were achieved in the experimentation phase. This signifies that satisfactory overall system performance was achieved by using the typing attributes in the proposed approach. Thus, the use of non-conventional typing features improves the understanding of human typing behavior and therefore, provides significant contribution to the authentication system.

Keywords: Keystroke dynamics authentication; Free-text; Non-conventional features; Decision trees; SVMs; ACO.

* Corresponding author. Tel.: +44 (0) 118 378 7565; fax: +44 (0) 118 975 1994; e-mail: A.F.A.Alsultan@pgr.reading.ac.uk

1. Introduction

The ongoing quest to find a technique to protect sensitive data and computer systems from harmful imposters, whilst also maintaining ease of use, is an important challenge in the field of computer and information security. Because the ID/password pair, the most common method for authentication, frequently fails to deliver an adequate balance between security and user-friendliness, more sophisticated methods have to be used. This is due to the ID/password pair being prone to social engineering, cracking and other forms of exploitation. Therefore, users are compelled to use extreme measures to safeguard their passwords, a procedure which includes remembering long and complex passwords in addition to the need for changing their passwords periodically [1] which causes them to endure great amounts of frustration and apprehension.

This research focuses on an alternative to the ID/password that verifies the identities of users based on their unique typing rhythms. This method provides a sufficient balance between practicality and safety, without requiring any additional hardware. Keystroke dynamics is considered to be an effortless behavior-based method for user authentication which employs the person's typing patterns for validating his/her identity. As was mentioned in [2], keystroke dynamics is "not what you type, but how you type." In this approach, the user types in text, as usual, without any extra work to be done for authentication. Moreover, it only involves the user's own keyboard and no other external hardware.

Keystroke dynamics is normally based on timing features that compute time lapses between two actions on the keyboard such as key press and key release. In this study, however, we investigate the use of non-conventional keystroke features in the authentication of users. Features such as typing speed, error rate, and shift key usage are utilized to find typing patterns that can be used to distinguish between individuals. Non-conventional features are considered during long free text input as they are extracted using calculations that spread along extended typing time.

These non-conventional features are important due to the lack of sufficient measurements that conventional keystroke dynamics present. Conventional keystroke data, in a very different way to other biometrics (e.g. image processing), captures very little information [3]. This information consists of the data that can be extracted from two consecutive keystrokes such as: the elapsed time between the release of the first key and the press of the second (digraph latency) and the amount of time each key is held down (keystroke duration) [2]. The majority of research, carried-out earlier in this area, focused only on these conventional features.

To enlarge the amount of information that can be extracted from a user input and therefore assemble better indications about his/her typing behavior, we focus our studies on non-conventional typing features that can be extracted collectively during long text input, in which more information is available. Long free text input is experienced daily in a manner that can be used to achieve continuous authentication [4].

Although there are many applications of keystroke biometrics used with fixed short text such as password hardening [5], there are scenarios where long free text input is more suited. For example: identification of one-of-many users who all have access to the resources in a work environment, the subject is identified when using any easily accessed desktop by his/her typing behavior of an e-mail or any other document. Another potential application for such long free text is verifying the identity of students taking online quizzes or tests.

Most of the work done in the field of keystroke dynamics authentication focuses primarily on timing features while ignoring other typing behavior such as editing patterns. Even previous studies that have included some non-timing features have not delivered the significance of these features in the way that they still focused on the importance of the conventional timing features, in the authentication process [6,7]. For that reason, we were motivated to explore the area of non-conventional typing features in order to concentrate on their distinctive ability to distinguish between individuals. A more in depth study on the effect of using various non-conventional feature subset sizes, which is to our knowledge not covered in the literature, has also been conducted.

In our work decision trees and Support Vector Machines (SVMs) are used to classify the typing samples collected from participants. Also Ant Colony Optimization (ACO) is utilized to select features that contribute more to the system in the case of SVMs, as decision trees are capable of performing feature selection in the tree building phase [8].

The rest of this paper proceeds as follows. Section 2 briefly introduces keystroke dynamics theory and discusses similar prior research in the area of keystroke dynamics user authentication. Section 3 describes the method developed in this study, in which we discuss the specific non-conventional features included in the study. In Section 4 we present our experimental results and consider the data space under investigation. Discussion about our results and some comparisons with previous studies are also included in this section. The final section concludes the topic and points out our research contributions and future work.

2. Keystroke Dynamics

Keystroke dynamics is categorized into two basic classes, namely: fixed-text and free-text [9]. The fixed-text keystroke dynamics method uses the typing pattern of the user when entering a predefined text. The same text has been previously used to train the system and is delivered by the user at log-in time. In contrast, the free-text keystroke method is considered easier for the user as it overcomes the problem of memorizing the text, something that the fixed-text method suffers from. As its name suggests, in free-text keystrokes, the text used for enrolment does not have to be the same as the text used for log-in. Moreover, free-text keystroke dynamics is used for enhancing security through continuous and nonintrusive authentication [10]. Thus, this research uses the typing behavior of free-text to resemble real-world situations, which allows users the freedom of not having to remember any text in order to go through the authentication process.

Keystroke dynamics is utilized in users' authentication by extracting typing features at the log-in session and comparing them with the typing features extracted at the enrolment session. These features include, among others: typing latency[11], keystroke duration [2], typing speed [11], shift key usage patterns [12] and typing pressure [13]. If the extracted features are adequately similar, the user is authenticated and if not the user is denied access.

Keystroke features extraction is usually performed after obtaining the users' raw data [14]. Among the data, timing features are popularly used and they are computed using two main values, specifically: the press time and the release time of each key, in milliseconds. These features are: Hold time, Down-

Down, UP-UP and Up-Down time. Most previous studies have typically employed more than one of these features [15].

Other non-conventional features, which are mainly used in free-text keystroke dynamics, were also considered in few studies. These features make use of extra information that can be obtained collectively during the training process. Unique patterns were produced after observing users for a longer period of time. Attributes such as the error rate and editing patterns have been found to give a fair idea about a user's typing behavior [9].

A large amount of research has been carried-out for quite some time to investigate how keystroke dynamics can aid user authentication in general. Specifically, we look here at some of the research that focuses on the extraction of non-conventional keystroke features, utilizing them in different ways.

The research conducted by Hempstalk et al. [16] included, in total, eight features in the typist dataset. Most of these features were based around the typing speed or error rate. The typing speed features included: average words-per-minute (WPM) rate, peak WPM and trough WPM, whilst error rate features included: backspaces, paired backspaces and average backspace block length.

In the research conducted by Villani et al. [3] long-text-input features were extracted. The feature set mainly consisted of percentages of key presses of many of (what were referred to as) special keys. Some of these percentage features were intended to capture the users' preferences for using certain keys or key groups. For instance some users do not capitalize or use much punctuation, which is a distinctive trait of their typing behavior.

Other percentage features were planned to acquire the user's text editing patterns. As an example, there are many ways to locate a specific key, such as using other keys, i.e. Home, End and Arrow keys, or using mouse clicks. There is also a large number of ways to delete, such as Backspace, Delete keys and Edit-Delete. Inserting and moving of words and characters can be done in different ways too, such as: Insert, shortcut keys, or Edit-Paste.

Shift-key patterns were incorporated in Bartlow and Cukic's research [17]. A password designed to enforce shift-key behavior consisting of 12 randomly generated characters was employed. The feature vector collected for each input sequence included many shift-related features. Examples of such features are: the average, standard deviation, maximum, minimum and total of the hold time for right shifts and left shifts. It also included the average, standard deviation, maximum, minimum and total of the delay time for right shifts and left shifts.

Based on the literature, only a few studies have taken into consideration non-conventional typing features such as features associated with editing patterns. Therefore, we are focusing, in this study, on these features to try and find consistent typing patterns that can be utilized for recognizing the particular typist.

3. Methodology

3.1. Feature Definition

A great deal of the research done in the keystroke dynamics field has been focused mainly on the timing features extracted from the user's typing stream. These features compute the time lapses between performing two actions on the keyboard such as calculating the time it takes a person to press a certain key, i.e. the Hold time, which can be done by subtracting the release time from the press time of that key. Latency time is computed in a similar way but the two actions are performed on two different keys pressed successively rather than both actions being performed on one key in the case of the Hold time. It is calculated by finding the time difference between the press time of the first key and the press time of the second key, in the case of Down-Down time. The Up-Up time and Up-Down time are also computed in similar manner [9].

In this research, we are striving to explore new features. Non-conventional features step away from the conventional methods which rely on computing the time lapses between performing two actions on the keyboard. Instead, non-conventional features focus on the overall typing patterns that a user follows during input that extends over a relatively long period of time. It considers the percentage of performing certain actions (in relation to the total number of actions), i.e. general typing actions or editing actions, which leads to understanding the user's typing behavior. Better perception of human typing patterns is particularly easier to capture while typing long free text in which more information can be extracted. We consider two types of typing features, namely: semi-timing features and editing features. We will briefly describe each category in this section as follows:

3.1.1. Semi-Timing Features

Different from the standard timing features used in most of the literature, we incorporate features that have been extracted using some form of time calculation. The time calculation followed in this category however, is slightly different from that of the regular timing features. These features have a collective property to them, as most of them are calculated during longer periods of time.

The first feature is the Word-per-Minute (WPM) feature which, as the name suggests, measures the user's average typing speed [16]. The total typing time is calculated from the very first key press until the very last key release and this is used in the final calculation of the WPM. The number of words are totaled and then divided by the total typing time in minutes; this is shown in Equation (1). Of course, this feature will easily distinguish between slow and fast typists. Nonetheless, it is not enough to find the difference between individuals who are close in typing speed.

$$\text{WPM} = \frac{\text{Number of words}}{\text{Total typing time in minutes}} \quad (1)$$

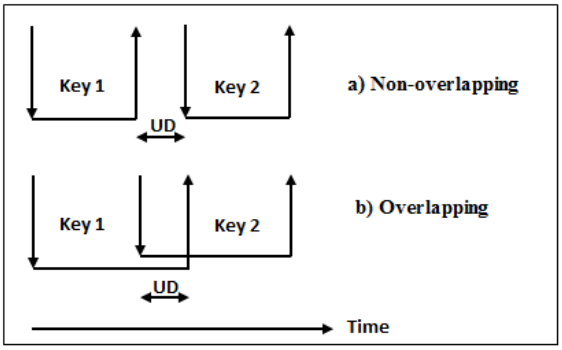

Fig. 1. Negative UD caused by overlapping keystroke events.

An interesting characteristic that can be found in some user's typing behavior is the number of negative Up-Down (negUD) actions detected in their typing stream. The negative Up-Down is due to an overlap happening between two successive keys being typed. This particular typing behavior is found in the typing stream of users who have the tendency to press the second key before releasing the first one. While most timing features are always positive because they represent the sequence determining

the keyboard output, the Up-Down feature, can be negative in some cases that might involve fast typists [3].

Figure 1 illustrates two different two-key sequences showing the Up-Down time in a non-overlapping situation and in an overlapping one. A keystroke is represented as a horizontal line with the down arrow marking the press and the up arrow indicating the release time. In part (a), a positive Up-Down time was produced from non-overlapping keystroke events and in part (b), a negative Up-Down time was produced from overlapping keystroke events where the first key was released after the second was pressed.

Some studies found it challenging to deal with negative UD time [18]. Here we are using it to our advantage by finding the percentage of negative Up-Down instances for each user. As mentioned in [19], a negative value of UD implies time reduction or faster pressing while positive values imply time addition or slower pressing between two sequences of keystrokes. We found that some users have absolutely no negative UDs whilst others have a fair amount, which was consistent in all the typing tasks they produced. This gives a good indication that comparing the percentage of negative UDs can be a good method to assist in user recognition. NegUD is computed as the percentage of the number of negative UD appearances and the total number of key-pairs, i.e. two keys typed consecutively, this is shown in the following equation:

$$\text{negUD} = \frac{\text{Number of negative UDs}}{\text{Total number of keypairs}} \quad (2)$$

A very similar typing behavior that has been, to our knowledge, hardly ever referred to in the literature is the negative Up-Up (negUU) time, which occurs when the typist tends to release the second key before releasing the first key. This characteristic happened with a few of our volunteers who participated in the data collection. Moreover, a negative UU only happens when there is a negative UD between the two successive keys. However if there happens to be a negative UD this does not mean that there is definitely a negative UU as shown in Figure 2.

Having said that, negative UU has the property of occurring less frequently, but if it does, there is a high possibility that it is a particular characteristic that an individual possesses intuitively. Thus there is a very good chance that it can be a good measure to employ in order to recognize that particular typist.

Similar to the previous feature, negUU is calculated as:

$$\text{negUU} = \frac{\text{Number of negative UUs}}{\text{Total number of keypairs}} \quad (3)$$

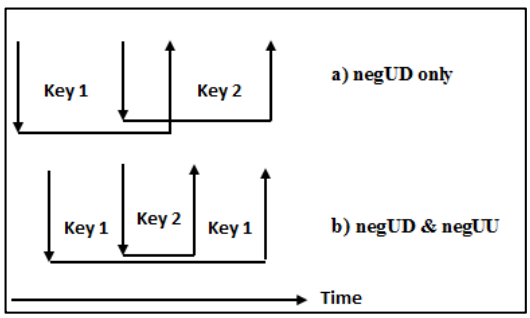

Fig. 2. Cases of negative UD only and negative UD/negative UU.

3.1.2. Editing Features

The second category of features does not give any attention to the time a user spends typing, rather it considers the way a user goes about the process of typing. Characteristics such as how frequently a user commits typing errors and how he/she edits text are studied here.

The error rate is the first feature in this category and it captures the percentage of times a user performs a typing error and corrects it [16]. This is simply calculated by dividing the number of times that a user commits an error, i.e. presses the backspace button, by the total number of letters typed, as follows:

$$\text{Error rate} = \frac{\text{Number of errors}}{\text{Total number of letters}} \quad (4)$$

The next five features are closely related as they all associate with the way a user incorporates capital letters in typing. Including a capital letter is done either by using the CapsLock key on the keyboard or by using a shift key together with the letter intended to be capitalized. We noted that if a user has the habit of using the CapsLock key, then he will hardly ever use the shift key for capitalizing letters, and vice versa. Therefore, using these two attributes simultaneously might be a good clue to understand the user's editing habits.

The first measure is CapsLock key usage which calculates the percentage of the CapsLock keys being used to produce capital letters in a given typing task. This is simply computed using the following equation:

$$\text{CapsLock usage} = \frac{\text{Number of CapsLocks}}{\text{Total number of keys}} \quad (5)$$

Shift key usage is a bit more complicated than it might appear to be as there are two different aspects in which users differ when it comes to shift key usage. The first shift key usage attribute is the right/left shift key choice. Some users use strictly the right shift or strictly the left shift whilst others alternate between the two [7]. The second attribute is the order of which the shift/letter keys are released. The shift key is always pressed before the letter key if the user is intending to produce a capital version of that letter. However, there are two orders that users go about when releasing those keys, they either release the letter key before releasing the shift key or they release the letter key after releasing the shift key. This behavior proved to be quite consistence throughout the different typing tasks for most users.

Based on the previous observations we suggest four different features that combine the two aspects of shift key usage. The percentage of each of the following was utilized; for the right shift key: Right Shift released After letter (RSA), Right Shift released Before letter (RSB); and for the left shift key: Left Shift released After letter (LSA), Left Shift released Before letter (LSB). They are calculated using Equation (6).

$$S = \frac{\text{Number of x}}{\text{Total number of shifts}} \quad (6)$$

Where: x= right shifts released after letter, incase S=RSA;
x= right shifts released before letter, incase S= RSB;
x = left shifts released after letter, incase S= LSA;
x = left shifts released before letter, incase S= LSB.

4. Experiment, Results and Discussion

4.1. Data Space

A total of thirty users participated in this study for data collection. Participants had different levels of typing skills that varied between moderate and very good.

During data collection, the participants were asked to perform eight typing tasks. The tasks involved copying text that consisted of around 1000 characters. The text was an excerpt from the Guardian newspaper. The text included both upper and lower case letters in addition to numbers and punctuation marks. Although the tasks included text that was chosen for the users to type, it is still considered free-text as the text used for training is not related at all to that used for testing [20].

Users were directed to enter the samples in the most natural way possible, i.e. the same way they usually follow when typing. Users were allowed to enter carriage returns and backspaces if needed. The data collection was performed by a GUI program implemented using the C++ language. The application was downloaded on the users' personal machines to maximize their comfort as they are more familiar with their own machine and its surroundings. Therefore, they were able to feel more at ease, and thus, to perform the typing tasks in a manner closer to that of their real typing behavior.

A feature vector, containing the nine features used in this study, was created and was stored in the database as the user's profile. This process was carried out by considering each one of the eight typing tasks as a single typing sample, the features from which were extracted separately. Therefore, eight samples per subject were included in the analysis phase for classifier training and testing.

4.2. Experiment and Results

Decision trees have been chosen as a classifier in this research as they are strictly nonparametric and do not require assumptions regarding the distributions of the input data [21]. Furthermore, decision trees handle nonlinear relations between features and classes [22].

Classification was carried-out through cross-validation as the number of samples was not sufficient enough to perform a regular training/testing process. Cross-validation is a statistical sampling technique that aims to ensure that every example from the original dataset has the same chance of appearing in the training and testing set. We followed the leave-one-out cross-validation protocol which is a special case of the well-known n-fold cross-validation [23].

N-fold cross-validation divides the data up into n chunks and trains n times, treating a different chunk as the test sample each time; such that for each of n experiments, it uses n-1 folds for training and the remaining one for testing. Leave-one-out cross-validation is exactly the same except that all chunks contain only a single sample.

In our experiment, eight samples were used to perform eight cross-validation experiments. Seven of the samples were treated as the training sample set and the remaining sample was regarded as the testing sample. In each experiment, a different sample was selected to act as the test data.

The Statistics toolbox in Matlab was used to fit the tree and predict the class of each of the test data. Moreover, the tree structure, i.e. the order in which attributes were chosen to be tested at each node, differs each time when a different training set was selected.

Furthermore, two error rates were used to infer the performance, namely: False Accept Rate (FAR) and False Reject Rate (FRR). FAR is the percentage of impostors who have successfully gained access to the system whereas FRR indicates the percentage of legitimate users who were denied access to the system [24]. Low error rates were produced by this study. The FAR and FRR derived from the decision tree classification process are listed in Table 1. Both error rates are presented utilizing datasets created by different numbers of participants. Results produced by 15, 25 and 30 users showed an increase in the error rates between 15 and 25 users. Yet, when increasing the number of users from 25 to 30 the error rates were very similar. When slightly enlarging the number of participants, we noticed the system reaching a stable performance level. However, more work is needed to prove this methods ability to work with datasets with large number of participants.

Using the nine features simultaneously had a good impact on the overall classification performance as the decision tree performs a form of feature selection in which only features that contribute to the overall-system decision are used in building the tree [8]. This is not the case when using only one or two features separately. This is due to the individual characteristics that each feature holds and that contribute collectively to the system's performance.

Table 1: System performance using multiclass classification.

	FAR			*FRR*		
Participants no.	15	25	30	15	25	30
Decision Tree	0.007	0.0104	0.0109	0.1	0.25	0.28
SVMs	0.0125	0.0181	0.0183	0.175	0.435	0.444

For comparison purposes, Support Vector Machines (SVMs) were also used in this experiment as it is one of the most successful classification techniques [25]. SVMs were chosen as a rival classifier because it follows a completely different mechanism to that of decision trees [26].

When using SVMs in classification, feature subset selection is in place. This is because a number of the non-conventional features are correlated with each other. Therefore, it is necessary to incorporate a feature subset selection mechanism when utilizing these features in order to reduce the dependency levels between the features [27]. Feature subset selection is also included in the building process of the decision tree where all redundant features are removed [8].

Feature subset selection is considered as an optimization problem, in which the space of all possible features is scrutinized to recognize the feature or set of features that produce optimal or near-optimal performance, i.e. those that minimize classification error [28]. Ant Colony Optimization (ACO) proved to be a good candidate for achieving that goal [29].

The selected features were passed to the multiclass SVMs machine learning mechanism in order to be used as the basic data for differentiating between classes. Leave-one-out cross-validation was also used to treat seven of the samples as the training sample set and the remaining one as the testing sample, in each cross-validation experiment. The classification process was implemented on MATLAB with the aid of the LIBSVM library [30].

This was done gradually by selecting one feature, using ACO, and then increasing the size of the feature set. Using only one or a small number of features yielded higher error rates. Similarly, using all or most of the nine features caused performance deterioration. The ideal size of feature set was 5 features which produced good FAR and FRR rates. A 0.0183 FAR and a 0.444 FRR were delivered using 5 features. Table 2 illustrates the influence of increasing the feature set size on the overall system's performance in a database containing 30 users.

Having the best features subset size to be only 5 features refers directly to the Curse of Dimensionality which corresponds to the problem that the amount of training needed grows exponentially with the number of features [31]. Since there were only 8 samples per person in this experiment, there has to be a reduction in the number of features used for classification to the least amount possible while conserving the maximum benefit provided to the classification process.

Using ACO, the features that contribute the most to the system performance in our experimentation were: negUD, Error Rate, RSB, LSA and LSB. Using these features solely in the classification process eliminated the redundancy caused by using all 9 features. That clearly contributes to improving the overall system performance. Furthermore, using only one or two of these features is not enough to find the fine differences between the typing behaviour of individuals in free-text keystroke dynamics.

Table 2: Error rates using different feature subset sizes.

No.	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
FAR	0.0315	0.0251	0.0248	0.0226	0.0183	0.0187	0.0191	0.0194	0.0203_
FRR	0.8194	0.6528	0.6435	0.5879	0.444	0.4861	0.4954	0.5046	0.52788

We understand that using a larger dataset and incorporating data from a greater number of participants will likely produce more reliable results. Therefore, similar to DTs, we incorporated data from datasets with different numbers of participants in the SVMs tests to understand how increasing the sample size will affect the system performance. In all these tests we decided to perform a subset selection of 5 features which proved to yield the best performance (as shown in Table 2).

Using datasets of samples size varying between 15, 25 and 30 users delivered a noticeable reduction in the system performance when increasing the number of participants from 15 to 25 (as shown in Table 1). Nonetheless, the increase from 25 users to 30 have produced very similar FAR and FRR. Similar to what was found in the DTs experiment; this shows the system reaching a stable performance level when slightly enlarging the number of participants. Nonetheless, experimenting with much larger number of participants is needed to provide sufficient evidence about the method's ability to work with datasets with large number of participants.

Moreover, decision trees operate by automatically performing feature subset selection in which the non-important or redundant features are not involved in the tree building process [8]. Features: LSB, negUD, negUU and CapsLock usage contributed most in building the decision tree as they formed the first levels of the tree structure. Thus, they collectively have a high ability to split the targets [32], which allows for better differentiation between individuals. Therefore, these features correspond to the features with higher impact on the performance of the recognition system. This partly matches the features extracted using ACO; as both LSB and negUD were found to have a considerable effect on system performance in both decision tree and SVMs/ACO classification cases.

Conclusively, Decision trees have a slight performance advantage over SVMs. They produced a higher accuracy system as the ROC is plotted closer to the upper left corner of the diagram in Fig.3.

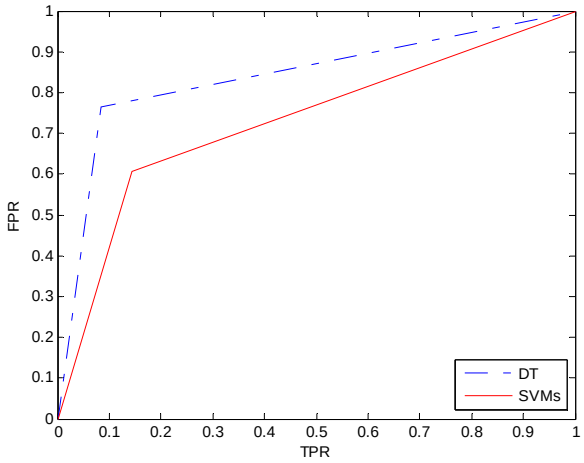

Fig. 3. Comparison between DT and SVMs by ROC curves.

The authentication process used until this point was done by training the system using data produced by the system's users to test if the system is able to recognize which of the system's users produced the test samples. Multiclass classification was utilized to achieve this aim. Multiclass classification works by deciding the test sample belongs to which of the available classes using the training data produced by all of the available classes [33].

In the second part of this study, we will focus on true intruder's recognition. In this section, typing samples from users who are completely un-known to the system are used to test the system's ability to recognize them as intruders and reject them. To achieve true intruder recognition, binary classification is used. For every test sample, binary classification, i.e. one-to-one classification, is performed against all available class to check if the system recognizes the intruder as any of the legitimate users.

Binary classification was performed using the training data of 25 genuine users and testing data from five intruders producing three typing samples each for testing the system. Table 3 shows the error rates produced by the 25 legitimate users without any intruders. The binary classification was performed for each user by representing the sample produced by that user as the positive class and all other samples are represented as the negative class [34]. This was carried-out using cross-validation similar to the multiclass classification experiment.

Lastly, the data from the five intruders was tested against each of the legitimate users' training data using binary classification. This produced similar FAR to that produced by the 25 legitimate users especially in case of SVMs. This provides some evidence that the system is able to recognize un-known intruders even when there is no prior knowledge about their typing patterns and the system was trained using samples from only legitimate users. Nonetheless, more experimenting is needed to prove that recognizing un-known intruders is in-place when there are much larger number of intruders.

Moreover, in true intruder recognition SVMs performed better than DTs. This is due to the nature of SVMs which leans towards the class with heavy samples [35] i.e. the class with negative samples in this study. The FRR in the intruders test was not computed due to not testing any legitimate users in this experiment.

Table 3: System performance using binary classification.

	Legitimate users		*Intruders*	
	FAR	**FRR**	**FAR**	**FRR**
Decision Tree	0.011	0.375	0.051	n/a
SVMs	0.0112	0.49	0.014	n/a

4.3. Discussion

This study was performed using the data collected in the research conducted by Alsultan et al. [29] in which the researchers considered user classification based on timing features only. These features included the hold time, Up-Up, Down-Down and Up-down of specific key-pairs. Although the performance of the system described in [29] was acceptable, there was a larger than desired FRR.

By using non-conventional features the FRR has been dramatically improved with a value of 0.28 in this study. While this figure is still not ultimate, it is quite good when considering the small amount of text used to recognize individuals. Nonetheless, a satisfactory FAR was also produced. The FAR, being as small as 0.011, is very comparable that produced by conventional features which leads to high expectations of further research in this area. The superiority of such non-conventional typing features over conventional timing ones, in user authentication, is proven by the low FAR and FRR produced by the non-conventional features.

The use of non-conventional features proposed in this paper have succeeded in providing a reliable medium for user authentication because employing these features enlarges the amount of information that can be extracted from a user's input. This is due to the fact that non-conventional typing features are extracted collectively during the whole time a text is being input by the user, in which more information is available, such as: words-per-minute, error rate, percentage of negative UDs … etc. Therefore, using this wide range of information available about the user's typing patterns, the system is able to assemble better indications about the user's typing behaviour, thus distinctively distinguish between individuals. Moreover, as the none-conventional features are collected during the whole time of text typing i.e. relatively long period, any random incidence that might occur will be averaged. As appose to the conventional timing features where few noisy appearances can affect the overall understanding of the use's typing pattern significantly.

Moreover, non-conventional features were utilized in the research conducted by Hempstalk et al. [16]. In their experiment, 8 features were extracted, some of which were based around the typist's speed: average words-per-minute (WPM) rate, peak WPM, trough WPM, error rate: backspaces, paired backspaces, average backspace block length or slurring of key press and release events: press/release ordering, press/release rate. A dataset consisting of 15 emails for each of 10 participants was created. Using one-class SVMs an FAR of 0.113 and an FRR of 0.331 were achieved. These results show that our research proved to realize better FAR/FRR despite having more subjects involved in the study.

Similar research was conducted by Curtin et al. [36] in which 58 features were extracted. The features varied between conventional timing ones and non-conventional ones such as total time to enter the text, total number of key presses for Space, Backspace, Delete, Insert, Home, End, Enter, Ctrl, all four arrow keys, left and right shift keys and the number of left, right and double mouse clicks. Recognition accuracy of 98.5% resulted from data collected from 8 subjects typing ten 600-characters long training samples and ten 300-characters long testing samples. This would have been a very encouraging result if the number of subjects was larger. A comparison between the method proposed here and some of the state of the art similar studies is presented in Table 3.

Table 3: Comparison with state of the art studies.

Study	Participant no.	***Features***		***System performance***		
		Convent.	*Non-convent.*	*Accuracy*	*FAR*	*FRR*
Alsultan et al. [29]	25	√			0.001	0.504
Hempstalk et al. [16]	10		√		0.113	0.331
Curtin et al. [36]	8	√	√	0.985		
Proposed method	30		√	0.76	0.011	0.28

5. Conclusion

In this paper we examined the usefulness of incorporating non-conventional keystroke features in the user authentication process. Unlike conventional timing features, non-conventional features benefit from the extra information that can be extracted from long free-text input. Features that have semi-timing properties such as words-per-minute, percentage of negative Up-Down time and percentage of negative Up-Up time were used. Moreover, features that explain the user's editing behavior were also used. These included the error rate, percentage of CapsLock usage, and percentage of both right and left shift keys usage.

The experiment produced good results considering the fact that it used free-text for user authentication which gave a good balance between the system's security and the user's comfort. The FAR and FRR rates were both satisfactory with the FAR being the slightly better of the two.

Therefore, non-conventional features such as those used in this study appear to be highly significant in keystroke dynamics applications such as user authentication. Moreover, decision tree classifiers also demonstrated a high level of success in such cases.

There is much more that can be done to improve this approach. One example of which is to expand on the typing features to include other non-conventional features such as the users' inserting and moving habits. Experimenting with different classification methods might also contribute positively to the overall system performance.

The fusion of conventional timing features and the non-conventional features presented here might work in favor of a better understanding the user's typing patterns which can be utilized to improve the error rates produced by merely non-conventional features. This is clearly ongoing research in which results thus far are extremely encouraging.

Acknowledgements

The authors wish to extend their gratitude to the participants who were involved in this experiment for the time they took out of their busy schedules to contribute in this study.

References

[1] R. Biddle, M. Mannan, P.C.V. Oorschot, T. Whalen, User study, analysis, and usable security of passwords based on digital objects,

IEEE Transactions on Information Forensics and Security. 6 (2011) 970–979.
[2] F. Monrose, A. Rubin, Authentication via keystroke dynamics, in: 4th ACM Conference on Computer and Communications Security, New York, 1997: pp. 48–56.
[3] M. Villani, C. Tappert, G. Ngo, J. Simone, H. St. Fort, S. Cha, Keystroke Biometric Recognition Studies on Long-Text Input under Ideal and Application-Oriented Conditions, in: The IEEE Computer Society Workshop on Biometrics, 2006.
[4] P. Bours, H. Barghouthi, Continuous authentication using biometric keystroke dynamics, in: The Norwegian Information Security Conference, 2009.
[5] F. Monrose, M.K. Reiter, S. Wetzel, Password Hardening Based on Keystroke Dynamics, in: The 6th ACM Conference on Computer and Communications Security, New York, 1999: pp. 73 – 82.
[6] D. Stefan, D. Yao, Keystroke-dynamics authentication against synthetic forgeries, in: The 6th International Conference on Collaborative Computing (CollaborateCom), Chicago, 2010: pp. 1–8.
[7] E. Lau, X. Liu, C. Xiao, X. Yu, Enhanced User Authentication Through Keystroke Biometrics, in: Computer and Network Security, Massachusetts Institute of Technology, 2004.
[8] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed., Prentice Hall, Englewood Cliffs, 2010.
[9] A. Alsultan, K. Warwick, Keystroke dynamics authentication: a survey of free-text methods, International Journal of Computer Science Issues. 10 (2013) 1–10.
[10] S.P. Banerjee, D.L. Woodard, Biometric Authentication and Identification using Keystroke Dynamics : A Survey, Journal of Pattern Recognition Research. 7 (2012) 116–139.
[11] J. V. Monaco, J.C. Stewart, S. Cha, C.C. Tappert, Behavioral biometric verification of student identity in online course assessment and authentication of authors in literary works, in: IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS), Arlington, 2013: pp. 1–8.
[12] E. Lau, X. Liu, C. Xiao, X. Yu, Enhanced user authentication through keystroke biometrics, in: Computer and Network Security, 2004.
[13] H.R. Lv, Z.L. Lin, W.J. Yin, J. Dong, Emotion recognition based on pressure sensor keyboards, in: IEEE International Conference on Multimedia and Expo, Hannover, 2008: pp. 1089 –1092.
[14] H. Saevanee, P. Bhattarakosol, Authenticating user using keystroke dynamics and finger pressure, in: The 6th IEEE Consumer Communications and Networking Conference, 2009: pp. 1–2.
[15] L.C.F. Araujo, L.H.R. Sucupira, M.G. Lizarraga, User authentication through typing biometrics features, IEEE Transactions on Signal Processing. 53 (2005) 851 – 855.
[16] K. Hempstalk, E. Frank, I.H. Witten, One-class classification by combining density and class probability estimation, in: The European Conference on Machine and Learning and Principles and Practice of Knowledge Discovery in Database, 2005: pp. 505–519.
[17] N. Bartlow, B. Cukic, Evaluating the reliability of credential hardening through keystroke dynamics, in: The 17th International Symposium on Software Reliability Engineering (ISSRE'06), 2006.
[18] J. Ilonen, Keystroke dynamics, Lappeenranta University of Technology. (2006).
[19] D.Y. Liliana, D. Satrinia, Adaptive behaviometrics using dynamic keystroke for authentication system, in: International Conference on Future Information Technology, Singapore, 2011.
[20] D. Gunetti, C. Picardi, Keystroke analysis of free text, ACM Transactions on Information and System Security. 8 (2005) 312–347.
[21] M. Friedl, C. Brodley, Decision tree classification of land cover from remotely sensed data, Remote Sens. Environ. 61 (1997) 399–409.
[22] B. Deshpande, Decision tree digest - understand, build and use decision trees for common business problems with RapidMiner, SimaFore. (2014).
[23] P. Refaeilzadeh, L. Tang, H. Liu, Cross-Validation, Encyclopedia of Database Systems. (2009) 532–538.
[24] M. Karnan, M. Akila, N. Krishnaraj, Biometric personal authentication using keystroke dynamics: a review, Applied Soft Computing. 11 (2011) 1565–1573.
[25] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery. 2 (1998) 121–167.
[26] K. Warwick, R. Craddock, An introduction to radial basis functions for system identification. A comparison with other neural network methods, in: 35th IEEE International Conference on Decision and Control, Kobe, 1996: pp. 464–469.
[27] D. Shanmugapriya, G. Padmavathi, An Efficient Feature Selection Technique for User Authentication using Keystroke Dynamics, International Journal of Computer Science and Network Security (IJCSNS). 11 (2011) 191–195.
[28] M. Dorigo, A. Colorni, V. Maniezzo, The ant system: an autocatalytic optimizing process, Tech. Rep. 91-016, Université Libre de Bruxelles, Milano. (1991).
[29] A. Alsultan, K. Warwick, H. Wei, Feature Subset Selection for Free-text Keystroke Dynamics, 2016.
[30] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, (2001). http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[31] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Transactions on Pattern Analysis And Machine Intelligence. 22 (2000) 4 – 37.
[32] L. Rokach, O. Maimon, Data mining and knowledge discovery handbook, 2010.
[33] G. Saggio, G. Costantini, M. Todisco, Cumulative and ratio time evaluations in keystroke dynamics to improve the password security mechanism, Journal of Computer and Information Technology. 1 (2011) 2–11.
[34] R. Giot, M. El-Abed, B. Hemery, C. Rosenberger, Unconstrained keystroke dynamics authentication with shared secret, Computers & Security. 30 (2011) 427–445. doi:10.1016/j.cose.2011.03.004.
[35] H. He, A. Ghodsi, Rare class classification by support vector machine, in: Proceedings - International Conference on Pattern Recognition, 2010: pp. 548–551. doi:10.1109/ICPR.2010.139.
[36] M. Curtin, C. Tappert, M. Villani, G. Ngo, J. Simone, H.S. Fort, et al., Keystroke biometric recognition on long-text input: a feasibility study, in: IMECS, 2006.

School of Systems Engineering

The Use of Arabic Language in Free-text Keystroke Dynamics Authentication

Arwa Alsultan | Kevin Warwick | Hong Wei

Introduction

The approach followed in this study involves the use of the keyboard's key-layout. The method extracts timing features from specific key-pairs. These key-pairs are pressed consecutively and have a relationship on the keyboard's layout. This relationship depends mainly on the key position of each character on the keyboard with relation to the other character. Moreover, these relationships can vary depending on the location of the two keys with respect to the overall keyboard layout.

Related Work

Most studies in keystroke dynamics involved only English input from the user [1]. Whilst such experimentation is very important, there is a clearly a lack of language variation used in such systems.

Key-pair Formation

There are five categories for key-pair relationships:

1. Adjacent: keys located next to each other on the keyboard.
2. Second adjacent: keys that are one key apart from each other.
3. Third adjacent: keys that are two keys apart.
4. Fourth adjacent: keys that are three keys apart.
5. None adjacent: keys that are more than three keys apart.

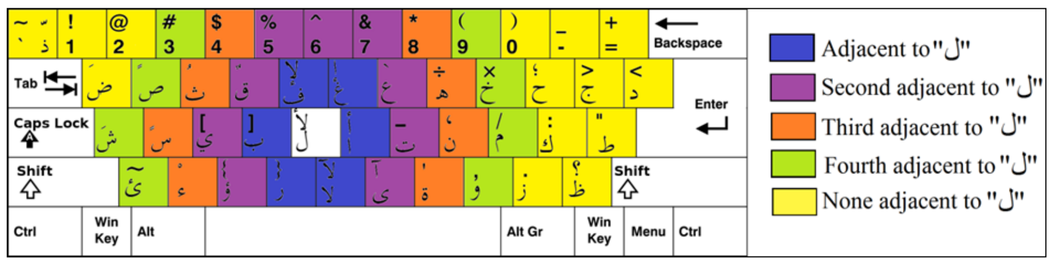

Figure (1) Key-pair formation example.

Each of these relationship categories can fall into one of the following overall locations:

1. Both keys are on the right hand side of the keyboard.
2. Both keys are on the left hand side of the keyboard
3. The two keys are located on different sides of the keyboard.

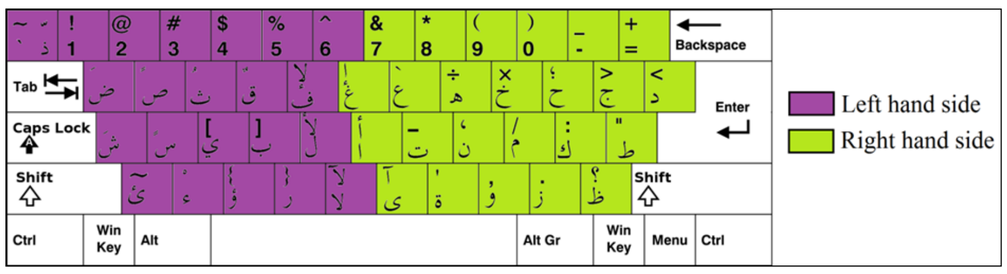

Figure (2) Overall key location.

Feature Definition

Features computed for every key-pair in the text [1]:

1. Hold time: is the time a key is pressed until it is released. Consequently, each key-pair has two hold times:
   1. Hold time for the first key (H1).
   2. Hold time for the second key (H2).
2. Keystroke latencies: there are three types of latencies:
   1. Down-Down (DD) time: is the interval time between two successive key presses.
   2. Up-UP (UU) time: is the interval time between two successive key releases.
   3. Up-Down (UD) time: is the interval time between a key release and the next key press.

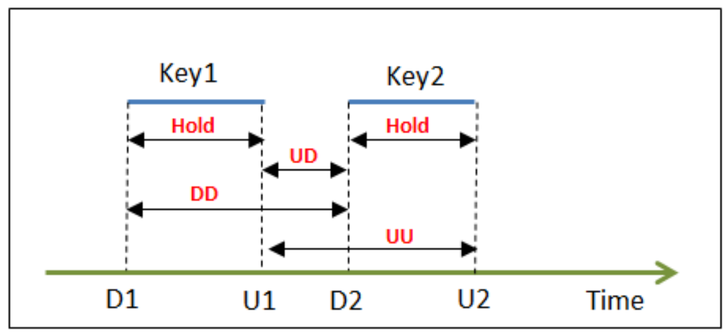

Figure (3) Timing features for a key-pair.

Experiment and Results

The overall number of timing features is 75 (15 key-pairs * 5 timing features). A total of fifteen users participated in this study for data collection. During data collection, the participants were asked to perform two 180 characters-long typing tasks. Ant Colony Optimization (ACO) [2] was utilized for feature subset selection. In addition, Radial Basis Kernel multiclass SVMs [3] was used for classification.

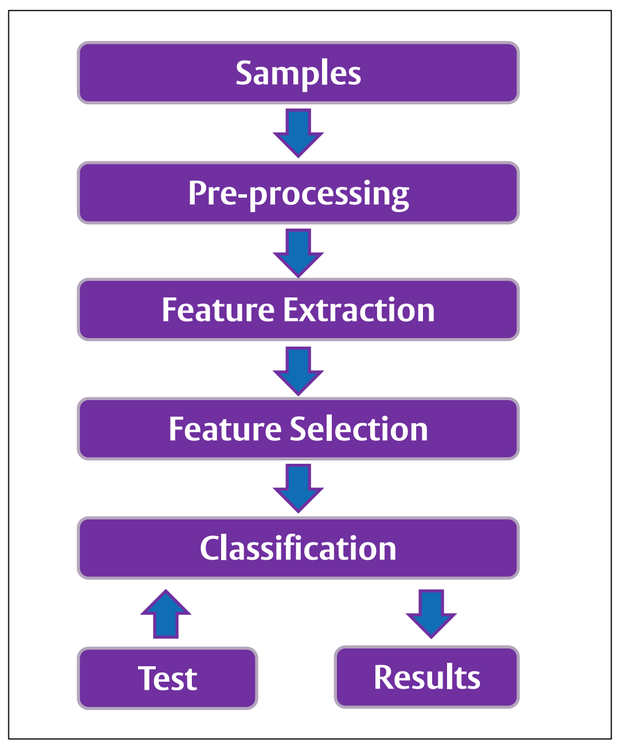

Figure (4) Flow of the system.

Table (1) System performance.

False Accept Rate (FAR)	False Reject Rate (FRR)
0.224	0.392

Conclusion

Results obtained from this study are encouraging as low FAR and FRR [4] were achieved in the experimentation phase. This signifies that satisfactory overall system performance was achieved by using the typing attributes in the proposed approach, while typing text in Arabic.

Acknowledgements

The authors wish to extend their gratitude to the participants who were involved in this experiment for the time they took out of their busy schedules to contribute in this study.

References

1. M. Karnan, M. Akila, and N. Krishnaraj, "Biometric personal authentication using keystroke dynamics: a review," Applied Soft Computing, vol. 11, no. 2, pp. 1565–1573, 2011.
2. M. Dorigo and T. Stutzle, Ant Colony Optimization. The MIT Press, 2004.
3. A. Vlachos, "Active Learning with Support Vector Machines," Masters Dissertation, University of Edinburgh, 2004.
4. S. P. Banerjee and D. L. Woodard, "Biometric Authentication and Identification using Keystroke Dynamics : A Survey," Journal of Pattern Recognition Research, vol. 7, no. 1, pp. 116–139, 2012.