1-1-2009

Inverse biometrics for keystroke dynamics

Fatema Rashid
Ryerson University

INVERSE BIOMETRICS FOR KEYSTROKE DYNAMICS

By
Fatema Rashid

B.S. in Computer Science, National University of Computer and Emerging Sciences

Karachi, Pakistan, June 2004

A thesis
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of

Master of Science

in the Program of
Computer Science

Toronto, Ontario, Canada, 2009

Author's Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

_____

Fatema Rashid

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

Fatema Rashid

Borrow List

Ryerson University requires the signatures of all persons using or photocopying this thesis.

Please sign below, and give address and date.

Inverse Biometrics for Keystroke Dynamics

A thesis for the degree of

Master of Science in Computer Science, 2009

By
Fatema Rashid

Ryerson University

Abstract

Tremendous research has been done in the area of computer security using biometrics. But not much has been done in the field of inverse biometrics, which consists of synthesizing artificial biometric samples that can be used for testing existing biometric systems or protecting them against forgeries. Due to the complexity of the data collection process and privacy and legal issues that are involved, finding volunteers for data collection is a challenging task. In this thesis, we introduce for the first time an inverse biometrics model for keystroke dynamics that can be used to generate as much data as desired. We show that these synthetic data behave as close as possible like real human data, making our inverse biometrics model a model of choice for testing the existing and upcoming biometric systems. Keystrokes dynamics biometric is a behavioural biometric technology, which allows user recognition based on the actions received from the keyboard while interacting with a graphical user interface. The proposed inverse biometric model first learns from the real human data and based on this experience, it generates synthetic users. Each synthetic user generated by model has a unique behaviour, but follows the properties of real human users. A twofold cross-validation testing technique is employed to validate the synthetic data using a suitable analysis model. Comparable performance results are obtained when applying the model to real human data.

Acknowledgements

It is my honour to express my deep gratitude to people who made this challenge possible for me. I would like to thank my supervisor Dr. Isaac Woungang, for giving me the opportunity to work under him. He offered me such a challenging topic for my thesis, and with his strong support and constant guidance, I finally achieve my goals. He did not only guide me in the course of this thesis, but also provided me an opportunity to be benefited from his vast knowledge. He was always present to help me out and guide me whenever I needed his guidance.

I would also like to thank Dr. Issa Traoré for sharing his enormous knowledge about keystroke dynamics, and computer security in general, and for giving his time and guidance to make this thesis better.

I would also like to express my gratitude to Dr. Ahmed Awad E. A. as it is his work that provided me a basis to put together this research thesis.

Lastly, and most importantly, I would like to thank my mother Khakashan Khalid Zaki and my father Syed Kalid Zaki . It is their prayers and support that gave me courage to fulfil my goals.

I would like to express my profound gratitude to my husband who has been a constant source of encouragement for me throughout my studies. To him, I dedicate this thesis. And above all, I would like to thank my daughter, Nabiha, who was my strongest motivation throughout my studies.

Dedication

To my husband

Rashid

I would not be here without his support and his strong belief in me.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

FAR	False Acceptance Rate
FRR	False Rejection Rate
MSE	Mean Square Error
NN	Neural Network
SMAG	Synthetic Mouse Action Generator
QQ	Quantile Quantile
MLP	Multi-Layer Perceptron
MM	Mouse Movement
DD	Drag and Drop
PC	Point and Click
KS	Kolmogorov Smirnov
ROC	Receiver Operating Characteristic

Chapter 1
INTRODUCTION

1.1 Context

Not too long ago, only a few number of computer systems were around, which were huge and heavy to carry, at the extent that they could not be fitted in a small room. This is no longer true. Today, several networks of computers are involved in our daily life activities, carrying information across large numbers of autonomous and embedded devices. The entire scenario has dramatically changed, and now, computers are making our lives easier. But, there is a price to pay for this comfort as well. Indeed, with an increase in the number of hacking incidents, identity theft and cyber crimes, computer security has become a very significant concern to organizations and people. In addition, with the increasing trend of making more data available to the users on the Internet and any other networks, secure computing is a very important requirement for all systems. From now on, operators of computer systems cannot simply allow highly confidential data to be accessed using simply passwords and user ids since passwords have been recognized as an extremely poor form of protection[1]. For this reason, new methods of authentication have been developed, among which some have already been deployed and others are still under development. In this regard, biometric technologies provide an alternative method for user identity verification in specific environments. Biometric is the term used to indicate a set of physiological and behavioural human characteristics that may allow verification of personal

[1] CERT: stands for Computer Emergency Response Team, Canergie Mellon University's Software Engineering Institute, http://www.cert.org/

identity [5]. In [22], the authors defined biometric as "the application of statistical analysis to biological data". In the particular field of computer security, biometrics is defined as "the automated use of a collection of factors describing human behavioural or physiological characteristics to establish or verify a precise identity" [21]. Nowadays, biometric systems are widely deployed for authentication purpose, and are considered to be the best so far in the market. Biometrics includes facial recognition, voice recognition, iris pattern recognition, and fingerprint pattern recognition. These human features can be exploited for user identification purposes. To these effects, one needs to convert them in digital format first, before performing different detection analysis algorithms on them. Achieving this conversion mandates the use of some devices specifically designed to capture these human features in such a form that they can be further used for security purpose. Unfortunately, such special devices are often not affordable to common class organizations and people, and are often difficult to integrate in old systems, i.e., systems previously designed, which have been continuously used in most organizations for their current daily computer's duties. But, as the biometric technology is growing, some vendors have integrated such special devices in their systems. For instance, Dell computers have a device used for capturing thumb print, in order for users to login to their laptops or desktops. However, this dependency on special hardware devices is a major hurdle in the use of new biometric systems since there still quite an overwhelming number of machines that are not equipped with such special hardware devices, or for which the intrinsic manufacturing features do not facilitate their integration with existing biometric systems, or even for which this integration is simply impossible. For this reason, this thesis advocates the use of behavioural biometrics based on keystroke dynamics as a suitable alternative for circumventing the above difficulty [9]. Indeed, using keystroke dynamics does not involve the use of any special hardware device to be

integrated with the systems in order to use biometric technologies for security purpose [7, 11]. The only requirement is the use of a standard human-computer interaction device, in this case, the keyboard, which is readily available in any modern computer. Keystroke dynamics (referred to as typing dynamics), is the detailed timing information that describes exactly when each key was depressed and when it was released as a user is typing at a computer's keyboard [6].

One of the important benefits of using keystroke dynamics (as well as other behavioural biometrics) is that the False Rejection Rate (FRR) and the False Alarm Rate (FAR) – two parameters characterizing the performance of such systems - can be adjusted by changing the acceptance threshold at the individual level [1]. This adjustment mechanism allows the system to explicitly define individual risk mitigation – a feature that traditional biometric technologies can never achieve [4]. Another benefit of using keystroke dynamics is that they can be captured continuously, i.e., not just at the start-up time and they can accurately be used to trigger an alarm to another system or person. Additionally, keystroke dynamics can also be useful in many other situations. A typical example is when a single user has several passwords because in such a case, forgetting or loosing the passwords is a very common concern. Another well-known example is when dealing with purchasing a good or paying a bill through a Web site via the Internet. In this case, the user is asked to answer few personal questions, say for example, his login and password, for authentication purpose. If the user cannot remember this required information, he/she can choose to be authenticated through a method involving the use of a Web based client-side application, which gathers the keystroke timing, then sends it to the server who then use it for authenticating the user. The confirmation of the identity of a user is a key issue for secure systems. The above keystroke dynamics-based authentication method can also be used to confirm the identity of a user. Typically, if a typing model of each legal user of the system is

available, the person using an account that is raising a possible alarm may be asked to enter a new typing sample. This typing sample is then checked against the typing model of the legal owner of the account. If there is a match, the user's identity is confirmed. If not, the verification of the identity of the user has failed and the user is immediately disconnected from the system, and his/her account is blocked.

1.2 Research Problem

In order to achieve the above mentioned benefits of biometric recognition systems based on keystrokes dynamics, these systems should be comprehensively tested. As a key requirement, the testing phase must involve the use of a large number of human users. But, it is very hard to attract human users for this kind of experiments, which is a major challenge for this research area. Finding volunteers to carry out a wide range of tests is really a nightmare because some people are reluctant in submitting their keystrokes due to privacy reasons. Some others may raise legal issues since they might already have been involved in using their keystrokes for safety purpose on other systems and do not want to jeopardize the privacy requirements of those systems. The situation is even worse when the volunteers have to deal with data submission. Indeed, the volunteers who accept to be enrolled in the experiments may be required to spend numerous hours or days for data testing purpose, in addition to installing a data collection software module in their own laptops or desktop, in the form of a client application. These limitations and hassles often hinder people to volunteer as participant to the data collection process, making the testing process hard to realize due to the lack of sufficient amount of real human data. This type of limitations does not exists in biometric systems using fixed text such as the one proposed in [6], simply because few test samples per individual are required, thus can be

easily and quickly collected for testing purpose. However, in most available biometric systems, it is imperative to have enough real human data gathered, in order to be able to comprehensively test these systems. Thus, data collection is a major problem faced by the testers of biometric systems as these systems must be thoroughly tested and in a satisfactorily manner, i.e., in such a way that it is ultimately possible to estimate whether the biometric technology has been able to successfully identify the users or it needs to be further refined. This thesis proposes a novel inverse biometric model, capable of generating synthetic keystroke biometric data that could be used to test biometric existing and future keystroke analysis models. Using our inverse biometric model, one can generate as many data as desired, which mimic as close as possible the behaviour of real human data, thus can be used for the testing of any biometric system. The effectiveness of our proposal is demonstrated through statistical analysis and simulation.

1.3 The Approach

Similar to other biometric technologies, a possible solution to the above-mentioned challenge is to develop a simulator that can be used to generate synthetic keystroke data. To this effect, the idea is to develop an inverse biometric model for keystroke dynamics based on the raw data collected from real users, with the goal to produce as many amount of data as required or as many number of users as required to be generated. Inverse biometric consists of the synthesis of artificial biometric samples that can be used for testing existing biometric systems or protecting them against forgeries [15]. *Biometric synthesis* is the inverse problem of *biometric analysis*, which involves collecting and processing biometric samples from human users and then, simulating these samples to achieve the same purpose [4]. Our proposed novel inverse biometric model is able to generate as much data as required that mimic the behaviours of real human-like

data, while being as much accurate as possible. To achieve this challenging task, our model is designed in such a way that it first learns from the raw human data, then simulates and generates the synthetic data afterwards. To this effect, several steps are required as illustrated in Fig.1.1. Building an inverse biometric model (like the one proposed here) to simulate keystroke dynamics, requires studying various factors and then, recognizing a pattern to be able to successfully imitate the acquired biometric information. Once a reliable model for simulation has been obtained, one can thoroughly test our biometric system with as many users as desired until producing a satisfactory result. A detailed explanation of these steps is provided in Chapter 3.

This simulator is not only capable of generating as much data as desired, but it also is capable of injecting behavioural characteristics in the data so that these data really mimic the behaviours of real human-like data as accurately as possible. Moreover, the simulator is capable of injecting noise in the data from the range of 0% to 100 %, as well as injecting typing errors from the same range to make the data as close as possible to real human data. Through these features, the users of our simulator will be able to generate various different types and sizes of data sets that are suitable for testing their ongoing or already developed biometric recognition systems.

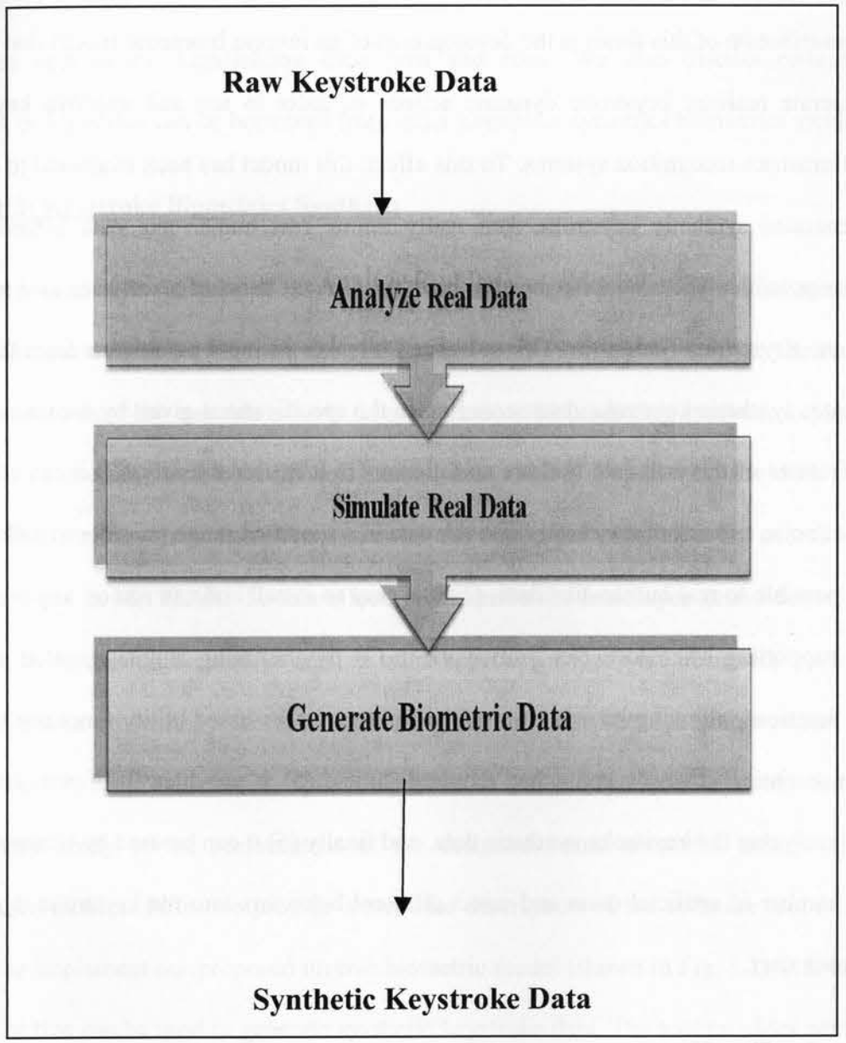


Figure 1.1: Simulation Process

1.4 Contributions:

The main contribution of this thesis is the development of an inverse biometric model that can be used to generate realistic keystroke dynamic actions in order to test and improve keystroke dynamics biometrics recognition systems. To this effect, this model has been evaluated to ensure that the generated synthetic keystroke data really mimic real human-like data behaviors as accurately as possible. The model has been implemented in the form of a software tool referred to as *Synthetic Keystrokes Generator*. This software tool takes its input parameters from the user, then, generates synthetic keystroke data according to the specifications given by the user. A few important features of this software tool are as follows: (1) It is user friendly, (2) it can allow its user to inject noise and errors very easily into the data in a specified range, in order to make them as close as possible to real human-like data, (3) it is easy to install and can run on any computer capable of supporting Microsoft .Net framework in C# programming language, (4) it embeds MATLAB functions, allowing its user to invoke neural networks-based utility functions that are suitable for statistical analysis and other required duties, (5) it provides the functionality of saving and analyzing the keystroke synthetic data, and finally (6) it can be used by researchers to create any number of artificial users and inject different behaviors into the keystroke dynamics actions of each user.

1.5 Thesis Outline

The rest of this document is composed of the following chapters:

Chapter 2: Related Work

In this chapter, we provide a state-of-the-art literature review of existing keystroke dynamics biometrics approaches, highlighting their pros and cons. We also discuss certain general principles or ideas that can be borrowed from other keystroke dynamics biometrics synthesis.

Chapter 3: Keystroke Biometrics Synthesis

The chapter constitute the core of this thesis. Our main contribution is presented therein. First, we describe the data collected from the real human users. Second, we analyze the real human data in order to establish suitable criteria (characteristics) for producing synthetic keystroke data which is as close as possible to the real human data. Third, we describe the data generation process. This step involves some discussion on the implementation of some neural networks-based functions in MATLAB. To this effect, we describe the multilayer neural networks-based architectures that were designed, as well as other approaches that were designed, but not finally adopted for data processing – here, reasons are given on why these other approaches were not retained. Fourth, we provide an in-depth comparison of real and synthetic keystroke data through an analytical study of the performance of the designed neural networks, using well known performance metrics. Fifth, we provide a stepwise description of the software tool (simulator), designed to implement our proposed inverse biometric model (shown in Fig. 1.1). It is essentially a simulator that can be used to generate synthetic keystroke data. The tool provides neat features for adding behavior to the keystroke dynamics actions, analyzing these actions and saving them to files. We also highlight the technical aspects of the implementation of our model, such as package diagrams and important use cases.

Chapter 4: Evaluation

Evaluating the findings is of course an essential part of the research work. In this chapter, we describe the different types of evaluation methods used to validate the keystroke synthetic data and discuss corresponding results. To this effect, we had to devise new validation techniques to ensure a thorough resemblance of the keystroke synthetic data to the real human-like data.

Chapter 5: Concluding Remarks

Finally, we conclude by analyzing and summarizing the results and highlighting future work that can be pursued for further investigation and enhancements. We also discuss few interesting questions that have arisen from the work carried in this thesis.

Chapter 2
Background Research

Biometric technology has been primarily used for user authentication and identification in a specific environment. The term "biometrics" is derived from the Greek words bio (life) and metric (to measure) [18]. Biometrics is the term used to indicate a set of human physiological and behavioural characteristics that may allow verification of personal identity [15]. Today, biometric systems are widely used for authentication purpose, and are considered to be the best so far in the market. Several different design approaches of biometric systems have been proposed in the literature. In this chapter, we review the current state of biometric technology and the approaches used so far in terms of keystroke dynamics.

2.1 Overview

With the exponential increase in the rate of computer usage and the increasing trend of presenting everything online, computer security has become a challenging task. User authentication is an important aspect when ensuring the integrity of computer networks. This justifies why new methods for authentications are being developed. Various methods of verification of the identity of an individual exist: (1) They can be based on a security system of personal identification which involves ID cards or keys – for instance, in the case of ID cards, each ID card has a unique number on it (2) They can be based on what the user knows and memorizes, such as passwords, (3) They can be done by using biological metrics such as retina prints, keystroke or finger prints, and (4) They can be done through encryption or cryptographic techniques; to name a few. The first two techniques have been widely used, but due to the

increasing risks to computer security and unauthorized access attempts, biometric technology is widely advocated as one of the methods of choice for effective authentication. Biometric features can be divided into two main categories namely physiological features and the behavioural features [5]. The physiological features include face, eye (precisely retinal or iris patterns), fingerprints, palm topology, hand geometry, wrist veins and thermal images. The behavioural features include voiceprints, handwritten signatures, keystroke, and mouse dynamics. Physiological features have been more successful than behavioural ones for the implementation of authentication systems because essentially, the former do not vary along time whereas the later may change drastically over a certain period of time or between two consecutive samplings. Typical well known examples of behavioural features are signature and keystroke dynamics.

2.2 History of Biometrics

Possibly, the first known example of biometrics in practice was a form of fingerprinting being used in China in the 14th century, as reported by the Explorer Joao de Barros [20]. He wrote that "the Chinese merchants were stamping children's palm prints and footprints on paper with ink to distinguish the young children from one another". This is one of the earliest known cases of biometrics in use, even nowadays.

Until the late 1800's, identification was largely dependent upon photographic memory. In the 1890s, a french anthropologist named Alphonse Bertillon sought to fix the problem of identifying convicted criminals and turned biometrics into a distinct field of study. He developed a method of multiple body measurements called Bertillonage (pseudonyme of his name) [20]. His system was used by police authorities throughout the world, until it started fading when it

was discovered that: (1) some people shared the same measurements, and (2) two people could get treated as one based on the measurements alone. After the failure of the Bertillonage technique, the police force started to use fingerprinting, a method developed by Richard Edward Henry of Scotland Yard, essentially reverting to the same methods used by Chinese people for years. In the past three decades, biometrics has moved from a single method (fingerprinting) to more than ten discrete methods [20]. There are over a hundred companies which are involved in designing/enhancing their biometric systems, using state-of-the-art technologies as they become available to them. Well-known practical examples of biometric technology deployments can be found in airports, naval air force locations throughout the world, hospitals, governmental institutions, to name a few. Usually, installing a biometric technology would involve a high cost of installation of its underlying surveillance system and other accessories, thus, cannot always be afforded by the general public.

2.3 Inverse Biometric

A major challenge in the area of biometrics system related research is the collection of real human data. Due to privacy and security issues, finding an adequate number of volunteers who will accept to carry out a wide test/experiment in order to validate the effectiveness of a biometric system is a tedious task, which can easily turn to be infeasible. More often, a common situation that arises is that the number of real human data samples available for testing tends to be smaller, which may seriously affect the trustworthiness of the experiment as well as that of the biometric system itself. The situation becomes even worse with the data collection process due to legal issues that may arise. In some cases, this process may involve having the participants

install a data collection software module on their personal computers, and this may require collecting several hours or days of test data, which is unlikely.

To circumvent this difficulty, inverse biometric data can be generated instead, and suitably be used in place of real human data. In [13], forward biometrics is defined as, "An analysis of Biometric Information that aims at classification, identification or recognition of this information". Whereas, the inverse is defined as, "Generation of Biometric Information to satisfy given characteristics, in particular, fluctuations, noises etc". Inverse biometric data are artificially created or synthesized biometric data. Since this type of data can be made available in huge quantity (providing that a suitable framework be designed for generating them), existing and future biometric systems can benefit from using them for testing purpose, and for protection against forgeries. Henceforth, the development of a framework in the form of a simulator that can be used to accomplish this goal is highly desirable. Such a simulator would be helpful for testing the biometric recognition system against any type of data, including extreme and normal data. Moreover, noise could be injected while testing the system, to determine how well the biometric system is able to cope with different scenarios and types of data. But, the generated synthetic data must be in accordance with real human data, so that they can be used to replace real human data.

There are various types of synthesized data [14], described as follows:

i. **Synthetic fingerprints:** The use of synthetic fingerprints is very beneficial in testing fingerprint identification systems. Today, automatic fingerprint synthesis involve dealing with problems such as testing fingerprint identification systems, training security personnel, biometric database security, and protecting intellectual property.

ii. **Synthetic signatures:** Signatures are related to handwriting, but more statistical data are available on handwriting than on signatures. Examples of methods for generating synthetic signature are those based on geometrical models.

iii. **Synthetic retina or iris images:** The synthesis of iris or retina images has not been developed yet. But some works are being done on the generation of iris layers pattern. Iris recognition systems scan the surface of the iris in order to compare patterns [19].

iv. **Synthetic speech and voice**: Considerable amount of work has been done on the synthesis of speech and voice. Now, a common goal for researchers is to improve the audio quality and the naturalness of speech by developing techniques for emotional coloring. New targets include identifying age, gender, emotion, personality, physical fitness, and social upbringing, to name a few.

v. **Synthetic modeling**: It is defined as the identification of a person through the pattern produced by walking [17]. Unlike other biometrics, gait offers potential for recognition at a distance or at low resolution. Gait signature is derived from the bulk motion and shape characteristics of the subject. Gait signatures are used in security system using screening machines.

vi. **Synthetic faces**: The problem with face recognition system is that changes in age, smile, and accessories make it harder for them to recognize the same face. This can be solved by training the system with these variations. Typically, these variations are generated artificially from real images.

2.4 Guidelines for Appropriate Use of Simulated Data

In [14], the main focus is on the formulation of guidelines for the proper use of simulated data for biometric authentication research purpose. The authors present a set of robust criteria for the use of synthetic biometric data, stressing that simulated data can only be used if they meet certain requirements. Three essential characteristics that simulated data must satisfy are as follows:

i. Simulated data must be flexible, meaning that in the random generation step, the distribution considered must have enough parameters, and these parameters must be robust enough to model the data under study.

ii. The data generation procedure should be as simple as the data can allow, but not simpler. The goal is to be able to represent the variation in the population using models capable of describing that variability, without extraneous and unnecessary parameters. The synthetic models and data should be complex enough to represent the real-time situation, but at the same time, should be simple enough to evaluate.

iii. The simulated data must satisfy the consistency and goodness of fit criterions. As an illustration, the authors in [16] compare the observed data to a proposed population model. If this population model is "statistically far" from the observed data, then that particular distribution is not a good fit for the targeted data, thus is rejected. However, if the population model is "statistically close" to the observed data, then it is judged as consistent with the observed data.

2.5 Databases for Synthetic Biometric Data

Collecting large databases with biometric information, such as fingerprints, is troublesome for many researchers due to the issue of protection of personal information. Imitation of the

biometric information can allow one to create databases with tailored biometric information, without expensive studies involving human subjects [13]. SFinGe is an example of tool used to create databases of fingerprints. This was developed at the University of Bologna. The reader can refer to *http://bias.csr.unibo.it/research/biolab/sfinge.html*, for more details about that tool. The generated databases have been included in the Fingerprint Verification Competition FVC2004 (see *http://bias.csr.unibo.it/fvc2004/databases.asp*) and perform just as well as real fingerprints.

2.6 Keystroke Analysis Research

Since we are dealing with inverse keystroke analysis and simulation, we will discuss, in this section, some of the works being done on keystroke biometric analysis. In these research works, real data from the keyboard are analyzed for user authentication. Initially, researchers were using fixed texts, which were entered at the time of initial authentication via user ID and passwords. Recently, we have started seeing some works on free text analysis, i.e., whatever text the user enters can be used for keystroke biometric recognition. In the sequel, we review some of the most well-known approaches used for the keystroke analysis of fixed and free texts.

2.6.1. Keystroke Analysis of Fixed Text

Several approaches have been proposed in the literature for keystroke analysis of fixed text. For instance Brown and Rogers used neural networks to solve the problem of identifying specific users through the typing characteristics exhibited when typing their own names [10]. In [7], Monrose and Rubin developed a technique to harden passwords based on keystrokes dynamics.

In [6], the authors investigated two problems related to keystroke dynamics, namely the aforementioned intrinsic variability of typing, and the possibility of typing errors. The approach tends to provide a reasonable level of accuracy even over remote connections. In their experiments, the authors defined a measure, precisely, the distance between two typing samples, which determines the elapsed time between the depression of the first key and the third key of a trigraph (i.e. three consecutive typed keys), and they called it the *duration* of the trigraph. Then, they take a typing sample and put its trigraph in an array and sort it on the basis of the trigraphs' duration in milliseconds. The output is then considered as a reference sample S1 and all other samples from the users are compared against it. Afterwards, the distance of any other sample S2 with respect to S1 is computed as the sum of the distances of each trigraph of **S2** with respect to the position of the same trigraph in **S1**. Based on this information, a specific user can then be authenticated. To test the efficiency of using the aforementioned distance measure for user classification and authentication purposes, the authors conducted an experiment involving 44 persons from their department, who were asked to type five times a fixed text of 683 characters, for a total of 220 samples. Their approach was able to yield a false acceptance rate (FAR) of 4% and an Impostor Pass Rate (IPR) of 0.01%. It was also pointed out that their approach faces some scalability issues, for instance, dealing with a larger number of legal users of the system quickly increases the chances that two legal users might have similar profiles. Furthermore, unlike other keystroke dynamics-based approaches for user authentication, their proposed approach allows typing errors, same typing samples for all users and smaller number of samples.

Instead of using trigraphs, in [8] another technique is used to verify the identity of the user using the keystroke dynamics his or her login string. The program measures the time duration between

the moment every key button is hit to the moment it is released. Users were asked to type the login string as the sample. So their text sample is also fixed and short in length.

2.6.2 Keystroke Analysis of Free Text

In contrast with fixed text analysis, few papers have been published on keystroke analysis of free text. In [11] the user authentication through keystrokes is done on the basis of the information gathered from the keystroke latencies. In [5], a comprehensive approach for keystroke analysis of free texts is presented. This research focuses on the issue of user identification through keystroke dynamics even after the login phase is over. The approach advocates that working with the typing rhythms of the free text - which is entered without any constraint – can make it possible to identify the user at any time. Few advantages of using keystroke dynamics have been described in this paper: (1) no additional task is performed for authentication as the user has to type anyways, (2) using keystroke dynamics systems is cheaper compared to other authentication systems because the only hardware required is the standard keyboard, which is readily available in almost all modern computers, (3) the typing rhythms are available even after the login phase is over, so that the user can be identified continuously. The authors in [5] have indicated some serious drawbacks when dealing with the analysis of typing rhythms. Firstly, from two consecutive keystrokes, it is possible to only extract the elapsed time between the release of the first key and the depression of the second (so-called digraph latency), as well as the amount of time each key is held down (so-called keystroke duration). Secondly, the information for the same user may change due to the changes in the environment, the type of keyboard used, and the text entered. To address these problems, most researches in the field of keystroke analysis have limited their experiments to samples produced from structured, predefined texts. The reason is

that entering longer texts may be tedious, and may not yield a satisfactory level of accuracy. One of the reasons justifying the use of free texts is that using shorter texts may not provide enough information, and, on the other hand, longer texts might require more time to be entered by the users.

Static keystroke analysis is defined as the analysis performed on typing samples produced by using the same predetermined text for all individuals under observation. Dynamic analysis implies a continuous or periodic monitoring of issued keystrokes. It is intended to be performed *during* a log-in session, after the authentication phase has passed [5]. In [5], the authors presented their typing samples in terms of the *n*-graphs, together with the duration of each *n*-graph. They observed that if the typed text is sufficiently long, the same *n*-graph may occur more than once. In that case, the *n*-graph is reported only once and the mean duration of its occurrences is determined. To handle short samples, their proposed solution is to merge these samples together to make them useful, but the results obtained are slightly worse than those obtained using long samples. Overall, their approach can achieve a FAR less than 5% and an IPR less than 0.005%.

In [3], an employee surveillance system based on free text detection of keystroke dynamics is introduced. The authors use real human keystrokes data for user authentication purpose. The detection approach for free text detection addresses various challenges, for instance, the ability to enrol a user using a non-predefined set of data, the ability to provide information about the user's identity based on a minimal amount of non-deterministic input, to name a few. The authors achieved their goals by utilizing a digraph approximation technique, which is based on a sorted time mapping technique. The approach utilizes neural networks to simulate and analyze the

user's behaviour based on encoded set of digraphs. The optimal performance achieved by their proposed detector was a FAR of 0.0152% and a false rejection rate (FRR) of 4.82%.

2.7 Inverse Biometrics for Mouse Dynamics

In the previous section we described the work done so far on keystroke dynamics. Since our research focuses on inverse biometrics for keystrokes dynamics, we need some background knowledge of both keystroke analysis of real data and inverse biometrics. Since the closest input device to keyboard is mouse, we explore the inverse mouse dynamics too. An interesting approach is presented in [4]. Here, the authors have discussed their experience on using synthetic data for research purpose in mouse biometrics. But, due to the lesser number of volunteers and the difficulty encountered in data submission, they were inclined to use synthetic data in addition to real human data. In [4], the mouse actions are classified under the following four different categories: (1) mouse-move (MM) - which corresponds to general movement, (2) drag-and-drop (DD) - where the action starts with the mouse button down, the movement, then the mouse button up, (3) point-and-click (PC) – where the mouse movement is followed by a click or double click, and (4) silence – i.e., no movement. The authors stressed that the monitoring silence intervals yield a lot of information about the user's behaviour, justifying why their analysis step is divided into movement analysis and silence analysis steps. For the movement analysis, some characteristics such as type of action, traveled distance (in pixels), elapsed time (in seconds) and movement direction, were considered. Few other factors were introduced, which are the same for all users of the system. For instance, the desktop resolution, the mouse cursor speed, and the mouse button configuration. Since the raw data itself cannot convey any information, they have been represented into various statistical graphs. For the analysis, the

authors considered the most representative factors which collectively represent mouse dynamics signatures. These are: movement speed, movement direction, action time, travelled distance and elapsed time. Typically, their mouse dynamics signature for a specific user corresponds to a sequence of 39 numbers corresponding to the values of the different factors involved. A mouse biometric synthesis model was then developed for simulation purpose. The main idea underlying the model is to take the raw data and the signatures, and based on them, create synthetic users such that every user is unique and everyone has his own mouse actions that represent his signature. The ideas from [4] proved to be very helpful in our proposed research, which is the development of a model for inverse keystrokes and their analysis. Until now no significant work is being done in the field of inverse biometrics for keystrokes. The use of synthetic keystrokes for user authentication is a pristine approach which is not yet explored much. We present the idea of first generating the synthetic keystrokes and then using those synthetic keystrokes for user authentication in the later chapters. The process of developing an inverse keystroke dynamics model is described in the next chapter.

Chapter 3

Keystroke Biometrics Synthesis

This chapter covers the main contributions of the thesis, which are twofold: (1) an inverse biometric model, in the form of a simulator, developed for the synthesis and simulation of keystroke dynamics – the workings of each of the modules that constitutes the model is described, (2) comparison of the generated keystroke synthetic data against real human data – the goal to demonstrate that the generated synthetic data can mimic as close as possible the behaviours of real human data. Our simulator can thus be used as a tool of choice for testing existing biometric systems or even those currently under development.

3.1 Overview

This chapter describes the process of how the data is being collected and how the generation of synthetic data is done. It highlights the characteristics of the volunteers and their typing trends. More precisely, raw data are collected from the user. After thorough examination of the raw data, a set of guidelines were established that would be used to ensure that the generated synthetic data depict the real users. Once these guidelines were defined, a keystroke biometric synthesis model was developed for simulation, producing synthetic data. After the data generation, the synthetic keystroke data is again examined to determine the extent to which the real human data and the synthetic keystroke data differ from each other. The chapter describes our designed inverse biometric model and its implementation, which comprises four modules. Each module is designed to perform a specific task, and all modules work together to achieve our ultimate goal of generating synthetic

keystroke data. Fig. 3.10 (in this Chapter) depicts a diagram describing the relationship between these modules

3.2 Keystroke Dynamics Data

The keystroke dynamics data are basically the actions generated by the keyboard when used by a specific user while interacting with the computer. The keyboard actions can be categorized as *key press* and *key release* [2].

Since the keyboard has only one way to interact with, which is the key, we need to identify different actions that are performed on the keys by the user while typing. The *key press* action is defined as the actual action when any key is pressed down, while the *key release* action represents the action when the key press is stopped and the key is free and being released. In order to quantity these actions, we need to define some way to calculate them. To this effect, we use the time as indicator to measure these actions, by considering the following time factors: *fly time* and *dwell time* (in milliseconds (ms)) [2, 3].

The *fly time* corresponds to the time taken by the user to move from the first key to the second one. More specifically, it is the elapsed time between the release of the first key and press of the second one. The *dwell time* corresponds to the elapsed time between the press of a key and the release of the same key. All the data that we collected is either the fly time or the dwell time.

3.3 Data Collection

The raw human data used in this thesis was collected in an experiment organized and presented in [2, 3]. In this section we briefly summarize the general settings for this experiment, which involves the description of the characteristics of the participants, the hardware, etc. Further details of the data collection process can be found in [2, 3]. We will briefly introduce the method of how the data was collected and what parameters were considered.

3.3.1 Detection and Analysis

There are different techniques which have been used to analyze and examine the real human data collected from users. The analysis may include recognizing the same user or differentiating between two different users. In [2], the detection and analysis is done with the help of neural networks. The enrolment process consists of training a neural network with collected digraphs on which the mapping order technique is applied. A digraph represents a typing action performed by the user from a specific key to another key on the keyboard. The time calculated in each digraph consists of the sum of the dwell times (the time needed to click on the key) and the flight time (the time required to move from one key to another). Mapped from/to key combinations are sent as inputs to the neural network in case of diagraphs, while the elapsed time is used as the training output. The neural network is a feed forward multilayer perceptrons. The number of input nodes is 2, which represents the "From" and "To" mapped keys. The output layer consists of one node, which represents the time needed to perform the digraph. The hidden layer consists of 20 nodes. The back propagation technique is used to train the neural network. A neural network is trained for each of the enrolled users. The weights of the trained neural networks for all users are saved in a repository for future use during the detection process. Further details of this detection algorithm can be found in [2].

3.3.2 Experimental Settings

An analytical model, borrowed from [2] was used in this study for data analysis purpose. This analytical model was validated through several experiments conducted in 2003, involving 23 participants who gave their informed consent. These participants include 16 males and 6 females, with varying computer skills and ages ranging from 13 to 48 years. The experiment configuration

involved the deployment of the client software on remote workstations connected to a central server via the Internet [2]. The tasks performed by the users varied from Web browsing to word processing and video game playing. The data collected was then sent directly to the central server and kept for future use.

In these experiments, the client software, which is responsible for monitoring keystroke actions, fed a detection server (software) with the monitored data. Then the user actions were monitored when the user login occurred and stops running when the user logout occurred. This client software was totally transparent since it does not affect any other applications. The detection server was installed on a local area network. It was configured to accept connections from local workstations and from outside the network over the Internet to allow remote users to participate in the experiment. A large number of participants were connected remotely to the network from their home computers. However, several users were also connected from national or international locations during the experiment. The server software stored the collected data in an internal database, along with other information, including the user ID. The hardware configurations for the participating computers varied from a Pentium 2 266-MHz processor to a Pentium 4 1.5- GHz processor. The server configuration was a Pentium 3 450-MHz processor with 256 Mbytes of RAM, running the Windows 2000 operating system. The client workstations could run different versions of the Microsoft Windows operating system (Windows 98SE, Windows ME, Windows 2000, and Windows XP).

The software caches all keystroke dynamics and sends the data to the central server. The server processes and stores the data in the database for future analysis. The experiment lasted for 9 weeks and an average of 119.979 digraphs per user was collected. The entire process is presented in Fig. 3.1

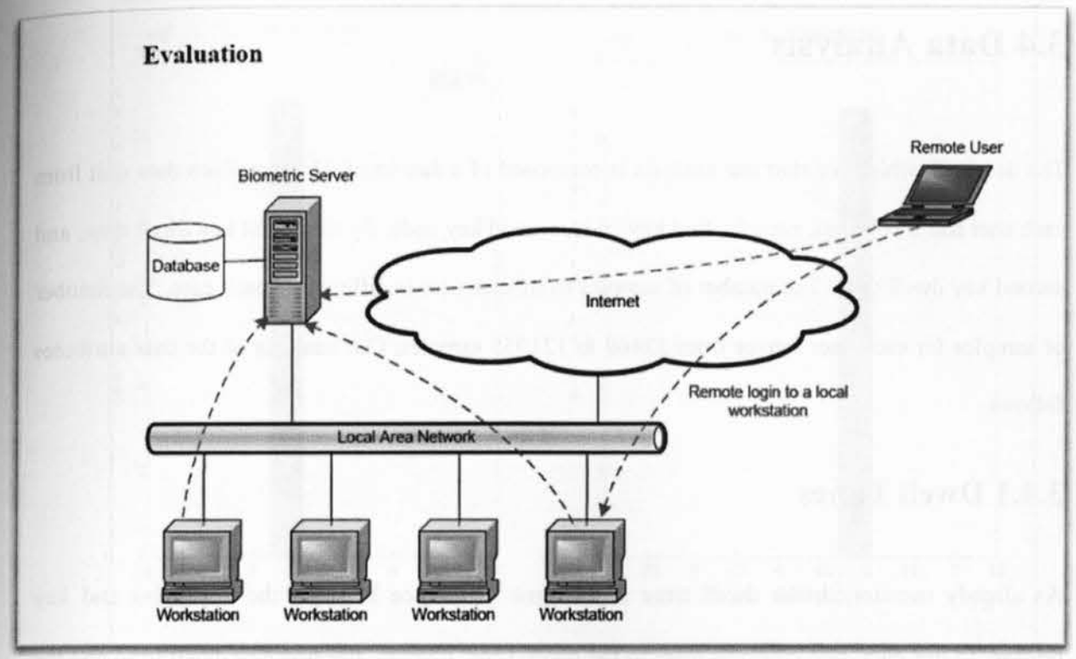


Figure 3.1: Experimental Environment

The optimal performance achieved by the detector was a false acceptance rate (FAR) of 0.0152% and a false rejection rate (FRR) of 4.82%. These rates are encouraging as a starting point, but more improvement are still needed, i.e., further tests are required in order to achieve more appropriate results by using larger data sets. However, as mentioned previously, it is very hard to have enough human real data for testing purpose due to the difficulty encountered in enrolling human users, added to privacy and security concerns which may arise in doing so. This thesis proposes an alternative solution which consists of synthesizing the above small sample (48 user's data) of raw keystroke data to generate as much synthetic keystroke data as desired, whose characteristics mimic as close as possible the characteristics of real human data.

3.4 Data Analysis

The data with which we start our analysis is composed of a dataset of 23 users. Each data unit from each user had 5 columns, namely, first key code, second key code, fly time, first key dwell time, and second key dwell time. The number of samples from each user is different in each case. The number of samples for each user ranges from 32468 to 121955 samples. Our analysis of the data attributes follows.

3.4.1 Dwell Times

As already mentioned, the dwell time is the time difference between the key press and key release. In our data, two columns refer to the dwell time, namely, the first key dwell time and the second key dwell time. Our analysis of the dwell time reveals that there is a large variation among the users. In Fig.3.2, it is observed that for one specific user the dwell times for both keys are ranging from positive to negative. But the dwell time cannot be a negative value. In this particular situation, the reason for having some negative values for the dwell time might be that a noise factor is involve in the process of submitting the data. Our simulation experiments aim at detecting and fixing this anomaly. It is also observed in Fig. 3.2 that the number of occurrences for dwell times less than zero are around 10 to 15 out of 108209, the total number of samples collected for this particular user. This represents only 0.00924% of the samples. Thus, one can conclude that these values can safely be omitted due to noise factors. This affirmation is also confirmed when running the comparison with other users. When other users are considered, we notice that in average, 0.02% to 0.9% of the samples from each user are negative values. Thus, we filtered all the negative values from all users.

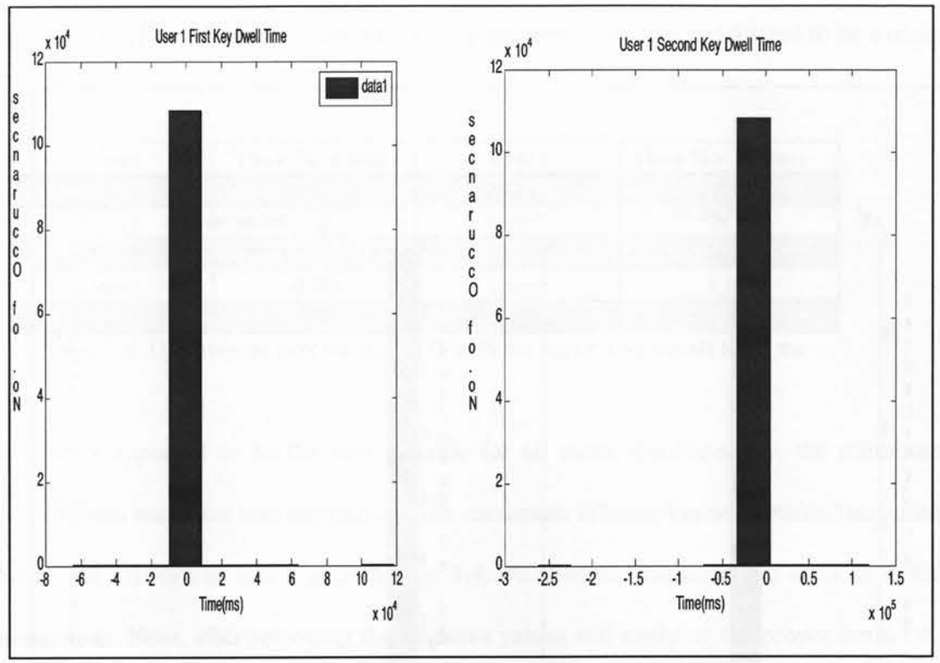


Figure 3.2: Histograms of dwell times for the keys of the same user.

In Fig.3.3, we compared two different users for their dwell times. It is observed that these two users are completely distinguished in their behaviour from each other. In case of user1, the dwell times are negative and positive for both keys, but in for user2, the dwell time values are well ranged, i.e., are positive values. We can thus infer that each user has its own characteristics, i.e., behavioural attributes, which can be used for authentication purpose. In this particular case, user1 has a higher noise level and user2 has a lower noise level.

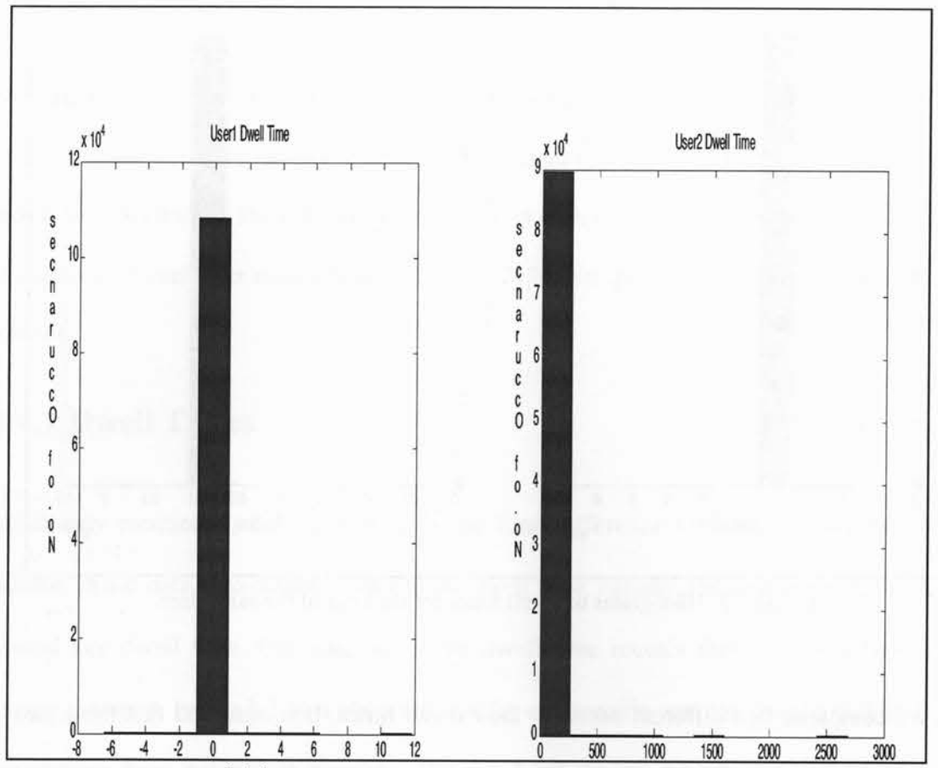


Figure 3.3: Histograms showing the dwell times for two different users.

Figure 3.3: Histograms showing the dwell times for two different users.

After filtering the negative values, we tried to find a suitable range of the dwell times which represents all the users. The values for dwell times were ranging from 1ms to 31104 ms. We analyze each user one by one to find the upper limit for the dwell time, and we retain different choices: 20 ms, 25 ms, 30 ms and 35 ms. We use 0 as the lower limit. The best results were obtained when the upper limit equals to 30 ms. Table 3.1 shows the percentage of dwell times which was found greater than 30 ms for a sample of 10 users. We then tested all other users and found similar results, but the table depicts only the results for 10 users. Since the table clearly

indicates that the dwell times greater than 30 ms are negligible, these are considered to be a noise factor.

Users	Time % >30ms	Users	Time%> 30(ms)
User1	0.3	User6	0.7
User2	0.9	User7	0.2
User3	0.45	User8	0.58
User4	0.96	User9	1.67
User5	1.9	User10	2.09

Table 3.1: Occurrences percentage of DT with the upper limit equals to 30 ms

These time limits appeared to be the best suitable for all users. Consequently, the minimum amount of data from each user was omitted and the maximum information was retained and used to train the neural network in later stages. In Fig. 3.4, we compare the same two users as in the above comparisons. Now, after removing the negative values and applying the proper limit, i.e., from 0 ms to 30ms, the observed data for the users is depicted in Fig. 3.4. For our training purpose, we took 3000 samples of dwell times from each of the 23 users. We can still notice that the two users maintain their own individual characteristic behaviour which is the soul of our research.

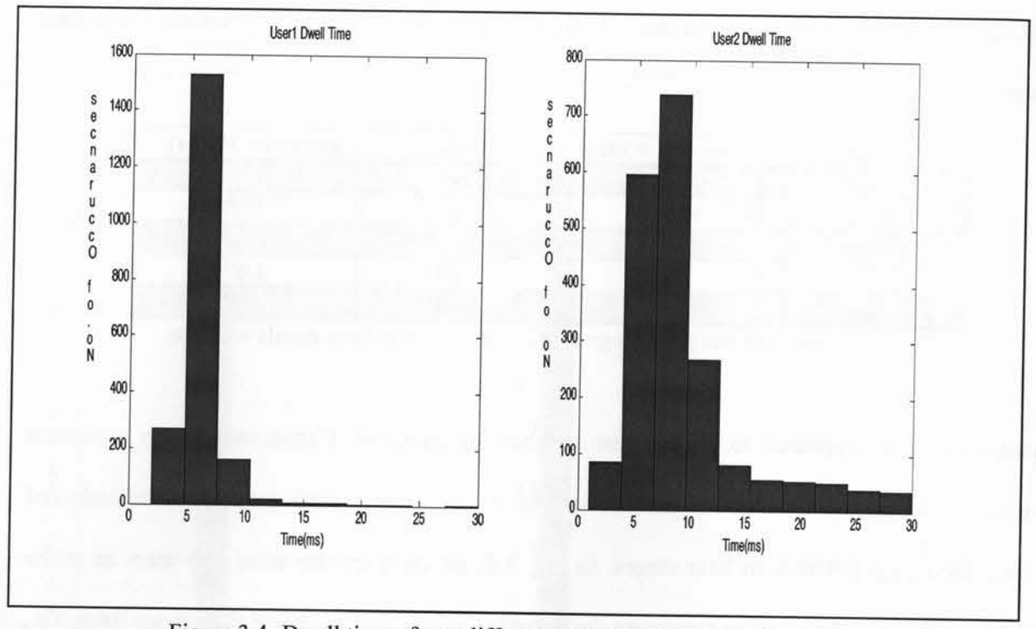


Figure 3.4: Dwell time of two different users after removing the noises

3.4.2 Fly Time

The fly time is defined as the time difference between the release of the first key and the press of the second key. When we analyze the fly times for different users, we found some important facts. Fig.3.5 shows that the range for fly time of one specific user is from 0 ms to 300 ms. Here, there is no issue of negative values as noise in any of the users' fly time. The range of fly times for this user is 0 ms to 300ms, which is consistent with that of all the 23 users we have analyzed.

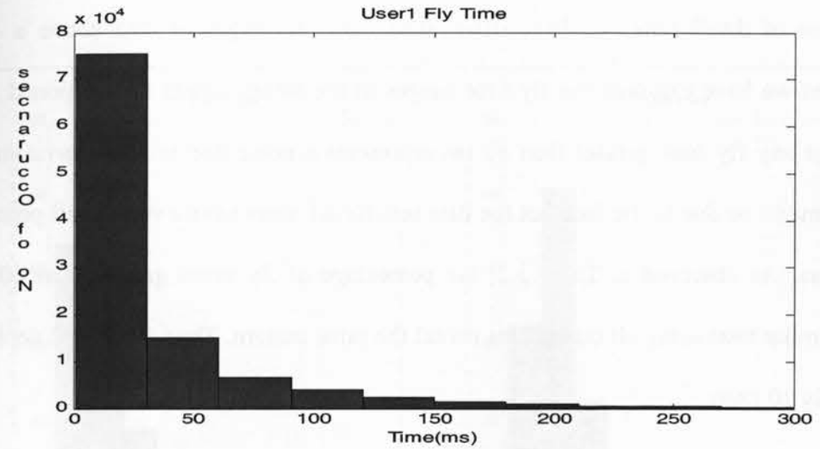


Figure 3.5: Histogram showing the fly time for one user.

In Fig.3.6, it is observed that the fly time ranges from 0 ms to 300 ms even for two different users. Moreover, the users also maintain their individual traits in fly time patterns. Each user has a different pattern in his/her fly time behaviour. But, we found that these ranges are not properly adjusted since the data above a certain threshold is only noise and do not have information pattern in them for every user.

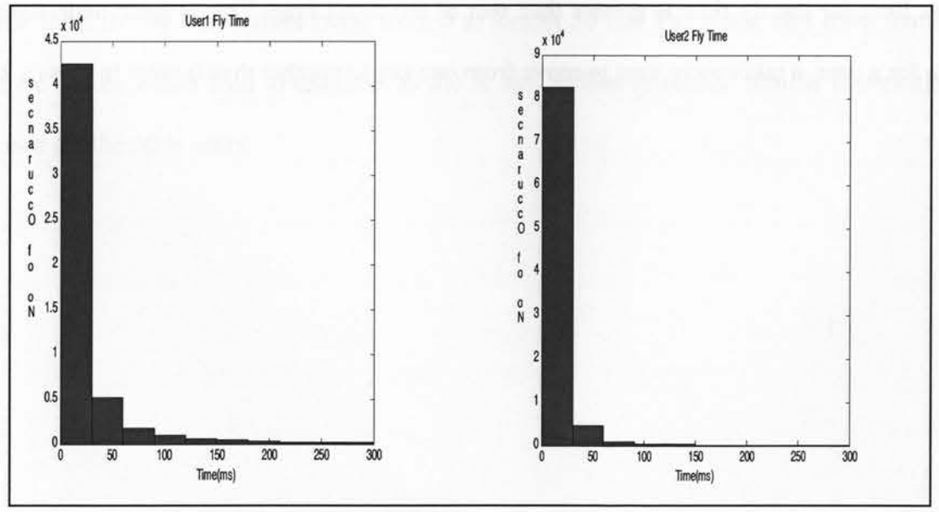


Figure 3.6: Fly times for two different users.

As in the case of dwell time, we have determined the percentages of data above a certain threshold, then we have explored the fly time ranges in the same manner as mentioned above. We found that any fly time greater than 60 ms represents a noise due to some environmental factors. This might be due to the fact that the data sets for all users have a very small percentage of such factors. As observed in Table 3.2, the percentage of fly times greater than 60 ms is negligible. Similar tests using all other users reveal the same pattern. Thus, Table 3.2 depicts the results for only 10 users.

Users	Time % >60ms	Users	Time%>60(ms)
User1	3.0	User6	2.01
User2	0.09	User7	1.2
User3	0.45	User8	0.49
User4	0.96	User9	3.47
User5	1.6	User10	2.53

Table 3.2: Percentage of fly time values above 59 ms.

After filtering the noises from the data so that it can be used to train the neural network (as described in the later sections), the histograms for the above mentioned two users now changed to the graphics shown in Fig. 3.7. In these histograms, it can be observed that more often, the upper limit for the fly time is greater than that of the dwell time. This is quite understandable since for a user, it takes more time to move from one key to another than it takes to press a key.

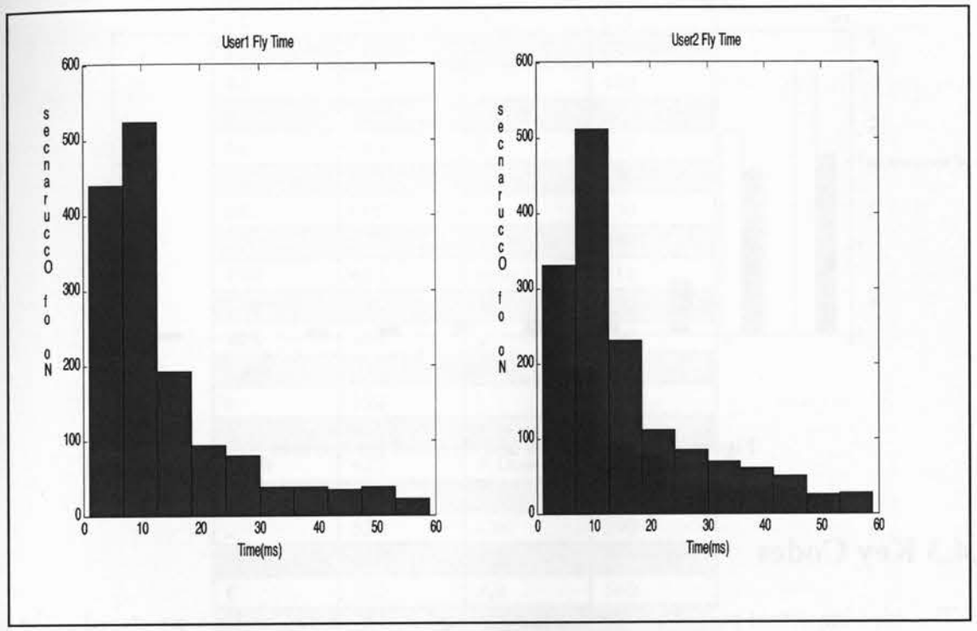


Figure 3.7: Fly times for two different users after removing noises.

For out training purpose, we took 3000 samples from each of the 23 users. It can be observed from Fig. 3.8 that the dwell time values are less than the fly time values in terms of ms – for this particular user, the fly time values range from 0 to almost 56 ms. The white bars show that the dwell time values which tend to diminish as the X axis values increases. Similar observations were made for the other users.

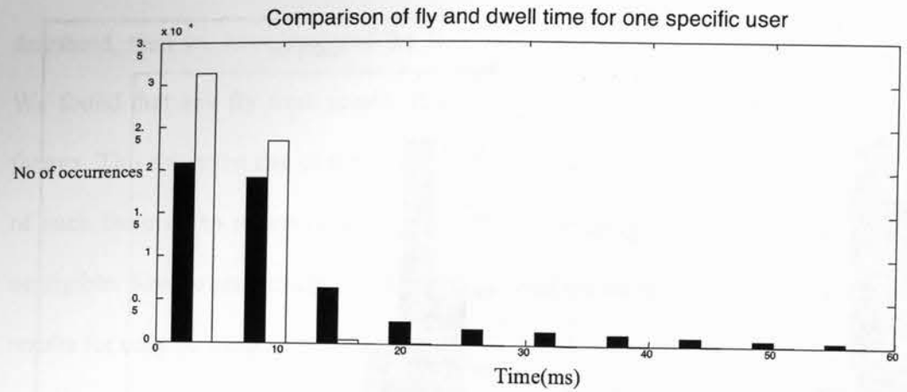


Figure 3.8: Dwell times and fly times for one user.

3.4.3 Key Codes

The third important part of the data is the key codes. These key codes were stored at the time the users submitted the sample texts, which contain all types of characters, including numerals and letters. The fly and dwell times were then calculated against these key codes. Numeral keys are keys that represent numbers 0 to 9 within the sample text. The keys which represent letters from A to Z are called letter keys. Other types of keys such as full stop (.), backspace, page up, page down, $, #, @,), [,* , to name a few, are called assisting keys. Command keys such as F1, F2, CTRL, Windows, to name a few, are called control keys. Table 3.3 shows a representation of all 84 symbols in terms of key codes, used in the sample texts.

Symbol	Code	Symbol	Code
F1	280	Pause	690
F2	600	Insert	821
F3	610	Del	831
F4	620	~	410
F5	630	1 to 9	20 to 100
F6	640	0	110
F7	650	-	120
F8	660	=	130
F9	670	B.Space	140
F10	680	Home	711
F11	870	Tab	150
F12	880	[	260
Caps	580	]	270
;	390	\	430
‘	400	P.Up	731
Shift	420	P.Down	811
,	510	End	791
.	520	Ctrl	290
/	530	Windows	911
€	720	Alt	560
Up	721	Space	570
Down	801	AltGr	561
Left	751	$	770
Right	771	Q to P	160 to 250
A to L	300 to 380	Z to M	440 to 500

Table 3.3: Symbols and their respective key codes

We use these key codes as input to our neural networks. These codes are collected at the time the users submit the sample texts. It should be noticed in the table that the letters on the third line of the keyboard from Q to P are coded from 160 to 250 with an increment of 10, while the letters on the forth line of the keyboard from A to L are coded from 300 to 380 with an increment of 10. Similarly, letters on the fifth line like from Z to M are coded from 440 to 500.

In Fig. 3.9, it is observed that the users maintain their individual behaviour in typing sample of texts. We took two users. Then we took out only the key codes representing the letters and numerals. We did not consider the assistant keys, or the control keys. We obtain the

histograms showing only the number of occurrences of letters and numerals. Fig. 3.9 shows that depending upon the number of samples inputted by the user, more or less equal number of letters and numerals are typed by any user. The same observation is made for all other users as well.

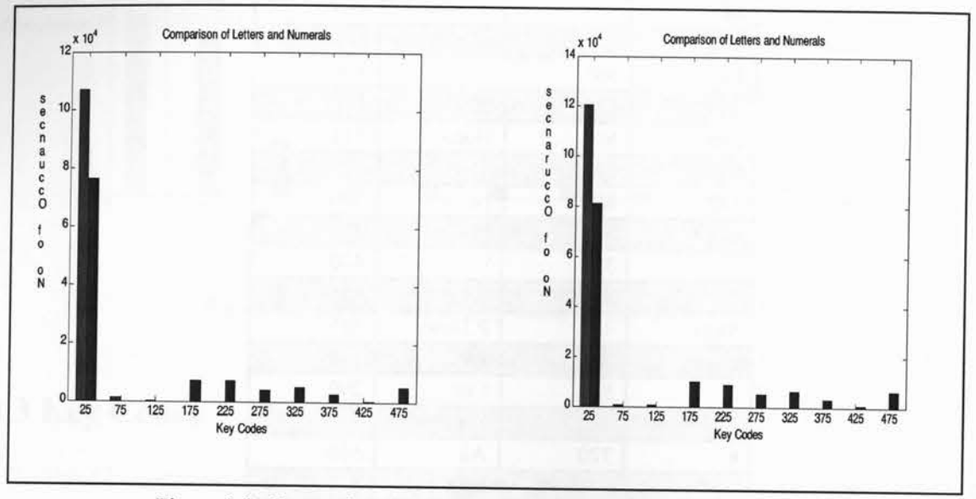


Figure 3.9: Comparison between letters and numerals for two users.

Table 3.4 shows a snapshot of the data taken from one specific user after removal of all noises and correction of all the ranges.

	1st Code	2nd Code	Fly Time	1st K D T	2nd KD T
	1	2	3	4	5
1	490	390	49	5	6
2	490	570	6	5	4
3	490	240	11	5	6
4	801	801	10	8	5
5	190	300	11	6	7
6	330	240	5	6	6
7	320	570	5	6	4
8	180	500	2	4	8
9	190	570	3	5	5
10	210	240	5	4	7
11	570	420	25	5	5
12	140	480	42	7	6
13	570	310	6	5	8
14	200	300	1	4	6
15	170	180	8	3	6
16	310	570	4	6	7
17	570	210	8	6	5
18	200	190	2	4	5
19	180	320	8	5	8
20	190	230	11	7	7
21	801	801	10	7	5
22	310	570	4	6	6
23	180	300	51	5	5
24	180	200	4	8	9
25	300	490	4	5	5

Table 3.4: Snapshot of the real human data after filtration.

3.5 Keystrokes Biometrics Synthesis

From the above analysis of raw data (from the users), we would like to generate synthetic keystroke data that depicts as close as possible the real human users. This section deals with that process, referred to as keystrokes biometrics synthesis. To this effect, some guidelines for the ranges of the fly and dwell times should be established, and some key codes must be considered. Based on these settings, a new keystroke biometric synthesis model is devised. This model is

capable of generating as much keystroke synthetic data as desired, which can be used to replace real human data when testing any keystroke biometric system. There is no limit on the amount users that can be produced as well. This model is based on four major modules as shown in Fig. 3.10, each of which is contributing its part to achieve the aforementioned goals.

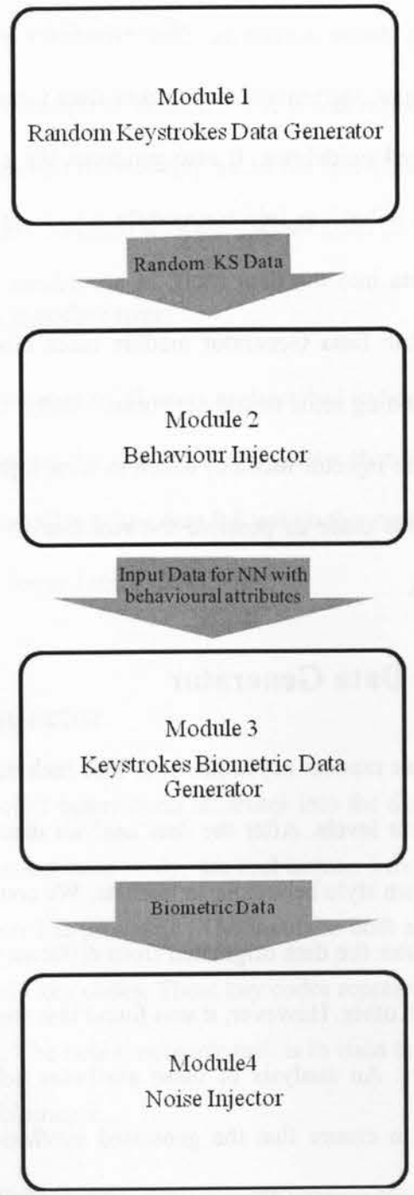


Figure 3.10: Architecture of the Synthetic Keystroke Data Generator.

The general functioning of the Synthetic Keystroke Data Generator is as follows: based on samples of raw data provided as input, the random Keystrokes Data Generator produces the key codes according to some predefined guidelines. It also produces the random keystroke data. These outputs are passed on to the Behaviour Injector module, which integrates the behavioural characteristics of the biometric data into the data itself, in accordance to the aforementioned guidelines. The Keystroke Biometric Data Generator module takes this input and creates the synthetic keystroke data through training some neural networks. Finally, this generated keystroke synthetic data is passed to the Noise Injector module, which in turn, injects noise elements into the data, with the goal to exhibit as close as possible the real human data characteristics. A description of each module follows.

3.5.1 Random Keystroke Data Generator

The role of this module is to generate random keystroke data. This includes numbers like number of samples per user and typing error levels. After the data analysis discussed in section 3.3, it was found that every user has his/own style/behaviour in his/data. We could not find any specific pattern which could be used to cluster the data originated from different users, the reason being that individuals are different to each other. However, it was found that some attributes are shared among the data from all the users. An analysis of these attributes helps us to devise some guidelines, which are to be used to ensure that the generated synthetic data are as close as possible from the real human data behaviours. These guidelines are as follows:

Dwell Time should be in a specific range

From our analysis of the data sets in Section 3.4.1, it was inferred that the upper limit and lower limit for the dwell time should respectively be 30 ms and 1 ms. This fact has been validated in Section 3.4.1 where we experimented with the data set from 23 users.

Fly Time should be in a specific range

From our analysis of the data set of 23 users as described in Section 3.4.2, it has been established that the fly time cannot be smaller than 1 ms and greater than 60 ms if it has to depict non noisy real human data and almost 97% of the data fall within this range. So the upper limit used for our analysis is 60 ms and the lower limit used is 1 ms.

3.5.2 Behaviour Injector

This module is used to inject behavioural attributes into the data. We are focussing on free texts rather than fixed texts. In our case study, the real human behaviour needs to be injected in the typing sample that the user has provided. The inputs to both networks, namely the fly time and the dwell time, are the only key codes. These key codes represent the type of the sample data that the user wants to submit. The neural network task is to train this sample data and generate some time values that make it biometric.

There are four types of key codes which can be used in the typing sample: letter key codes, numeral key codes, assisting key codes, and control key codes. The generated key codes are given to the network. They depend on the type of keys used. For instance, the user may want to create a typing sample consisting of only letters, setting the percentages of all other codes to

zero, or the user may want to create a sample consisting of all types of keys, setting a largest in number of numeral keys, followed by the letter keys, followed by control keys, and finally a smallest number of assisting keys. Figure 3.11 depicts the key codes used by two different users. It can be observed that user 1 has more key codes falling in the range [200, 300] whereas user 2 has more key codes falling in the range [100, 200]. It means that the sample submitted by user 1 has more letters and the sample submitted by user 2 has more numerals. Behavioural attributes are therefore user dependent.

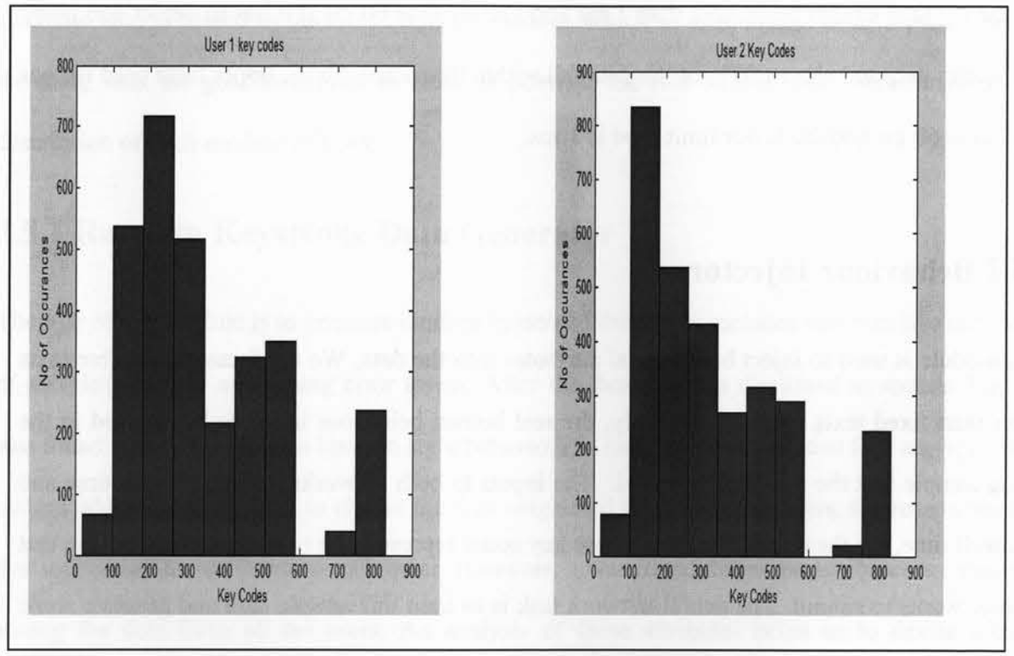


Figure 3.11: Comparison of key codes for two different users.

3.5.3 Keystroke Biometric Data Generator

This module is the heart of our system since it is the one responsible for creating the synthetic data. It takes the raw data as input, along with the behavioural attributes, then, generates the

biometric data. It involves using two different neural networks for training the raw data: the dwell time neural network and the fly time neural network. The design methodology and learning phase of both neural networks follows.

3.5.3.1 Dwell Time Neural Network Design

We used the generalized feed forward network with two layers: the hidden layer and the output layer. The hidden layer has 20 neurons, while the output layer has only one neuron as shown in Fig. 3.12.

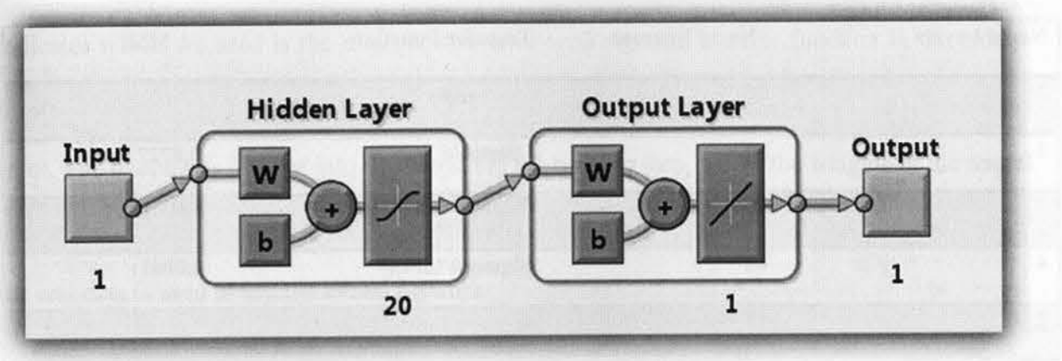


Figure 3.12: Architecture of the Dwell Time Neural Network.

The generalized feed forward neural networks (Fig. 3.12) can fit multidimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layers. In these networks, the connections between the units do not form a directed cycle. The networks are able to process larger data sets in a short period of time. Since the data we had is from the real human users, it has been considered as consistent. But, there is a trade off between the number of neurons and the processing time. When more neurons are involved, the time it takes to process/train the data is much longer, which may result into a better learning. In other words, a

fewer number of neurons would yield a shorter data processing time, which may result into a decreasing learning capability. In the hidden layer, we make use of the *log sigmoid* as transfer function, which calculates the layers output from its net input. Another alternative would have been to use linear functions, but through experiments, we have observed that the log function is the bset choice since it provides the best possible results. After experimenting with different number of layers and learning functions, we have determined the lowest value for the mean square error (MSE), and we have used it as our performance indicator. Table 3.5 shows the variations for the different combinations that we have experimented.

No. of Layers	No. of Neurons	Transfer Function	MSE
3	40	All Linear	0.0800
2	20	Sigmoid	0.0727
2	20	Linear	0.0722
4	40	Sigmoid/Linear	0.0801
2	20	Sigmoid/Linear Mixed	0.0713

Table 3.5: Combination of variables used to yield the best MSE.

As observed in Table 3.5, the least mean square error is achieved when we used a two layers neural network with the sigmoid hidden neurons and the linear output neurons, keeping the number of neuron to 20. The input to the neural network is the key code taken from the real user's data. The output of the neural network is the period of time the user pressed that key or dwelled on that key. The neural network should then learn how to calculate the dwell time given any key code from the same coding scheme as that of the input. This makes the data biometric. In [4], the authors indicated the reason of not using a classifier in lieu of the neural network for this type of data. In fact, using a classifier involves using both real and synthetic data. We have

followed the same advice in this thesis. We define the time as a function of the key codes and we get the neural network to learn this function from the data provided.

3.5.3.2 Learning Phase

The neural network learning phase is supervised. Learning is defined as a procedure for modifying the weights and biases of a network [2]. There are two types of learning: supervised and unsupervised. The difference between supervised and unsupervised learning is that the former is provided with a set of proper network behaviour examples, whereas in the latter, the weights and biases are modified in response to the network inputs only. The performance indicator which we used is the mean square error with sigmoid transfer function in the hidden layer. This indicator measures the network's performance according to the mean of the squared error. The learning is divided into two steps: (1) the training step, where the weights of the neural network are modified under a supervised learning, and (2) the testing step, where some part of the real data is used to test the neural network.

3.5.3.2.1 Training Phase

We train the network by using the *Levenberg-Marquardt back propagation* algorithm. To this effect, we select 18 of the 23 users, then, consider 3000 samples from each user. We keep the remaining 5 users out the testing exercise. The purpose of the training is to get the neural network to produce data that are as close as possible to real human data. The aforementioned 3000 samples of data from each user are placed in a matrix format and are randomly shuffled. The number of samples taken from each user is considered the same so that the training process is unbiased towards any user, i.e. does not depend on a particular user. This mechanism has

appeared to be useful in ensuring that the input data given to the neural network follows (at a certain degree) the guidelines that were derived from raw data (as described in Section 3.4).

3.5.3.2.2 Data Processing Phase

In this phase, the data needs to be transformed in a form that the neural network would find it easy to learn based on the information enclosed in the data. Initially, we considered a binary coding scheme where each key code is represented by a binary number, resulting into 84 inputs for the dwell time network and 84x2 =168 inputs for the fly time network. We then assign numbers to each key sequentially. For instance, for the key number 7, using any specific code number, 1 will appear as input in the 7^{th} position (out of the 84 positions) and the rest of positions will be filled out by 0. But, the major issue with this coding scheme turns is that not every user employs every key in his/her typing sample (for instance, for one particular user, some of the bits in the input might always be equal to 0). This coding scheme was not suitable for the learning of the neural network since we have observed that using such a scheme increases in the number of inputs, which consumes enough memory in the MATLAB environment, and the network could not reach an optimal performance.

To address the above problem, we pre-process the data in some other way, taking into account that some transfer functions require that the inputs and targets be scaled to fit within a specified range. To this effect, we use the *premnmx* function. This function pre-processes the data so that its minimum is -1 and its maximum is 1. Its syntax is

[pn,minp,maxp] = premnmx(p)

where p is the normal input and pn in the transformed data scaled in the range [-1, 1], and *maxp* and *minp* are respectively the maximum and minimum ranges of the input data. This function normalizes the inputs and targets of the network training set in such a way that they fall in the interval [-1, 1]. When the data is pre-processed before the experiments, and after the simulations, it needs to be converted back to the original scale. The corresponding post-processing subroutine for *premnmx* is the *postmnmx* function.he weights of the neural network are initially set to 0, but after each epoch, the weights are updated according to the mean square error. More precisely, at each epoch, the time produced by the network is compared with the real time shown in the data, and accordingly, the weights of the network are adjusted. This process takes place until the real target time output and the produced output time are as close as possible to bring the mean square error to its lowest possible value. The mean square error finally achieved is 0.0713 in 102 epochs. The training performance of the dwell time neural network is depicted in Fig. 3.13. It can be observed that the MSE equals 0.0713. This means that after the training phase, when the neural networks calculates the dwell time, the mean square of the difference between the calculated time and the real time is at least 0.0713 seconds, which is very small when time is measured in milliseconds.

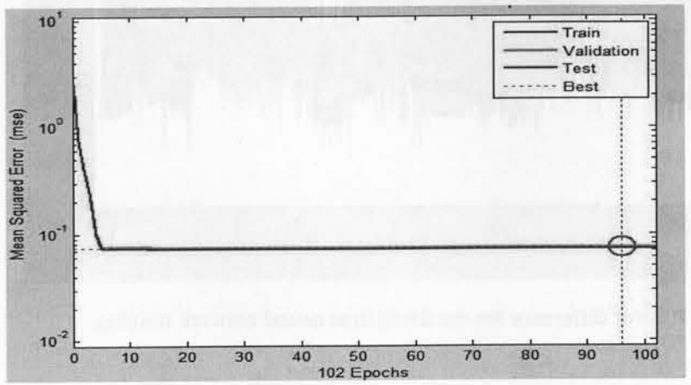


Figure 3.13: Training performance of the dwell time neural network.

The error difference, i.e. the difference between the targeted output and the generated output of the dwell neural network training, for the same input, is depicted in Fig. 3.14. It is observed that for most of the times, the calculated error is in the range [-0.5, 1], which is negligible. This means that for most key codes, the neural network has been able to determine a dwell time value which is very close to that of the real human data. Typically, the neural network has been shown to accurately guess (at a certain degree) the time value after learning from its input, i.e. from the human behaviour provided to it in the form of training data. After this phase, the neural network, along with its current configurations of weights and biases is saved and kept for future generations of as many data as desired, and also for testing the remaining 5 users that were not previously involved in the testing phase.

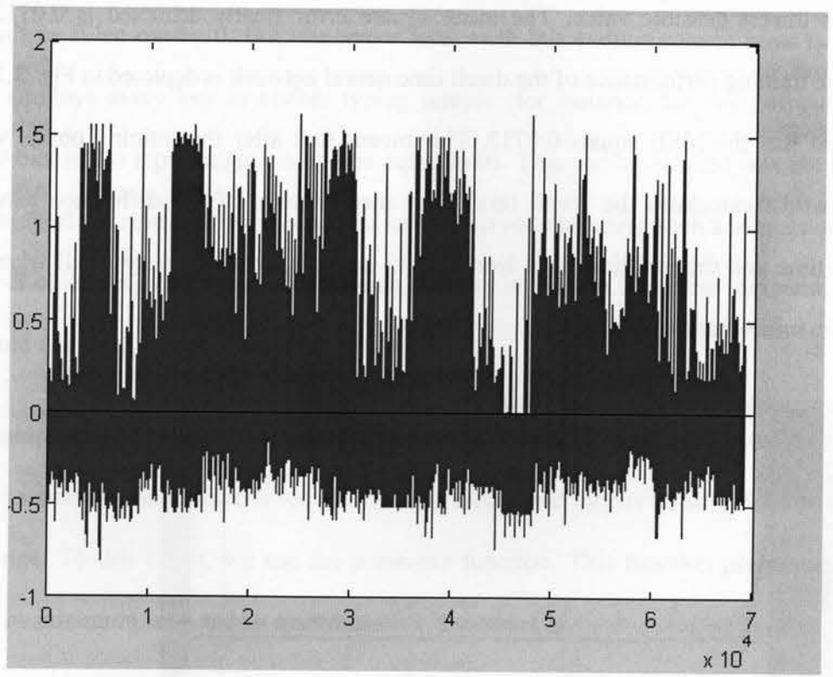


Figure 3.14: Error difference for the dwell time neural network training.

Testing Phase

In the testing phase, we used our remaining 5 users that were excluded from the above step. We extracted the same number of 3000 samples from each of the 5 users. The samples are made by key codes only, then, are put in a matrix format, and then, are randomly mixed to make the results unbiased. It should be noted that the dwell time for these 5 users is not shown to the network. Rather, the neural network determines the dwell time value for the 5 users based on what it has learned from the previous learning experience of the 18 users. From this, one can determine the error difference, i.e. the difference between the resulting output time values and the actual time values given in the data set for the 5 users. Typically, the goal has been to train the network for establishing an association between the key code and its dwell time, and for generating the biometric data based on this association. The error difference, i.e. the difference between the time values generated by the network and the time values from the real human users is depicted in Fig. 3.15. It can be observed that this difference is mostly in the range [-0.4, 0.9], which indicates that the generated values are quite close to the real ones when time is measured in milliseconds.

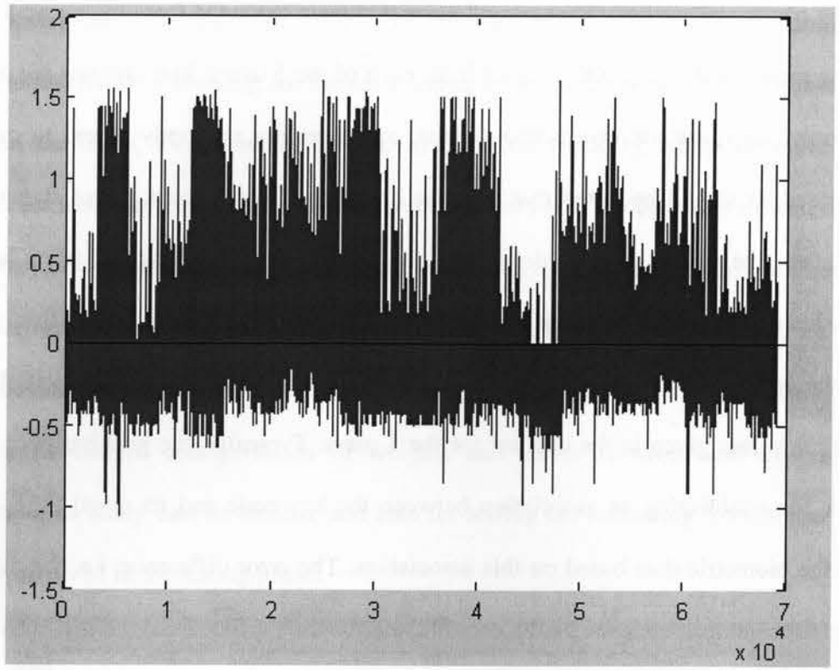


Figure 3.15: Error difference between the real and generated times values for the remaining 5 users.

3.5.3.3 Fly Time Network

The fly time network is of the same type as the dwell time network. It is also a generalized feed forward network with two layers: the hidden layer and the output layer. The hidden layer has 40 neurons, while the output layer has one neuron as shown in Fig. 3.16.

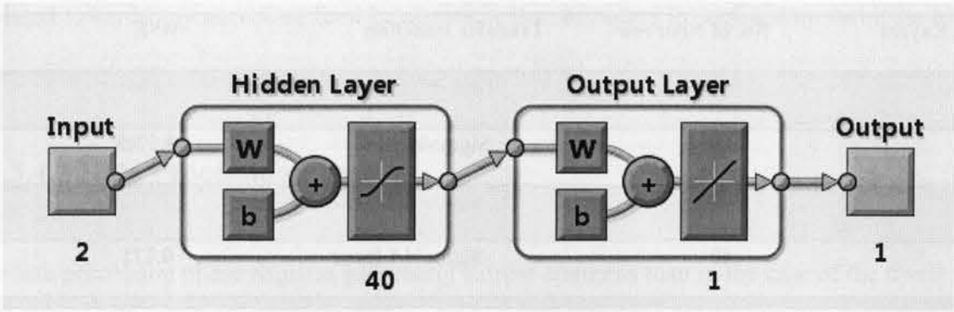


Figure 3.16: Architecture of the fly time neural network.

In order to achieve the least mean square error (MSE), we have considered various combinations of factors such as number of neurons in the hidden layer, the number of layers, the choices of transfer functions, to name a few. The least MSE of 0.1678 is obtained after running 56 iterations of the neural network. We have used 2 as the number of layers, 40 as the number of neurons, along with some sigmoid hidden neurons and linear output neurons transfer functions. The combinations of different variables leading to the minimum MSE are shown in Table 3.6. The first and second key codes are the inputs to the neural network whereas the fly time between the key with the first code and the key with second code is the output of the network. The input and output are made noise free according to the guidelines described in Section 3.2. The network is trained to learn how to generate the fly time in order to make the data biometric. The rest of the settings are similar to what was done in the context of the dwell time network.

No. of Layers	No. of Neurons	Transfer Function	MSE
2	35	Linear	0.1701
2	20	Sigmoid	0.1705
2	40	Sigmoid/Linear	0.1678
4	40	Sigmoid/Linear	0.171
3	20	Sigmoid/Linear	0.1702

Table 3.6: Combination of different variables used to yield the best MSE value.

3.5.3.4 Learning Phase

The supervised learning is used for the neural network. The performance function employed is mean square error as in the case of the dwell time network, along with the sigmoid transfer function in the hidden layer. This function measures the network's performance according to the mean of the squared error. The learning is divided into two steps: (1) the training step, where the weights of the neural network are modified under supervised learning, and (2) the testing step, where some part of the real data are used to test the neural network.

3.5.3.4.1 Training Phase

As we did previously, the neural network is trained using the *Levenberg-Marquardt back propagation* algorithm, by selecting 18 out of the 23 users, keeping the remaining 5 users for the testing phase. As previously done, we have considered 3000 samples from each user, and have put them in a matrix and have randomly shuffle them. The purpose of the neural network has been to train it so that it produces the fly times that are as close as possible to that of human data. The number of samples taken from each user is the same, i.e., 3000, so that the training is

unbiased towards any user. Our final input matrix is constructed in such a way that it no longer depicts any specific user, but only the human behaviour.

3.5.3.4.2 Data Processing Phase

The data processing phase requires addressing similar concerns than in the case of the dwell time network: (1) the issue of transforming the data in a form suitable for the neural network to easily learn from the information – here, the *premnmx* functions are used for pre-processing the data instead of binary coding schemes (as previously described in Section 3.5.3.3), (2) the neural network initially sets the weights to 0.0, and after each epoch, the weights are adjusted according to the MSE value. This process is continued until the least MSE is achieved – here, the obtained MSE is equal to 0.1678 after 151 epochs (as shown in Fig. 3.17). This means that when the neural network tries to generate the fly time values based on its own learning, the difference between the generated fly time value and the real fly time value is at least 0.1678, which is negligible when the time measured is in milliseconds.

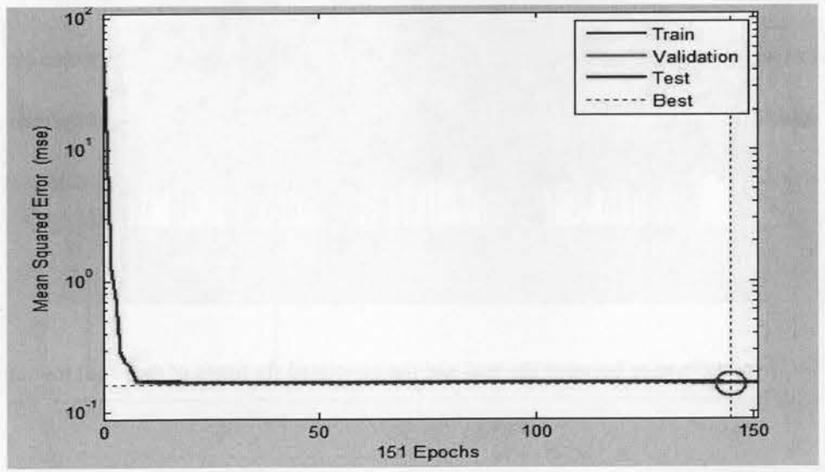


Figure 3.17: Performance of the fly time network.

The error difference, i.e. the difference between the values of the synthetically generated fly time and the real fly time of the input training data, are compared in Fig.3.18. This represents the difference of error for the fly time values for the 18 users' data that we used for training purposes. It is observed that the error difference is in the range [-0.4, 0.9], which is negligible when the time is measured in milliseconds. Therefore, we can infer that the values predicted by the neural network are very close to the real values which represent the human behaviour. This network and its weights and biases are then saved and kept for future generations of data, and for the testing of the real of the remaining 5 users.

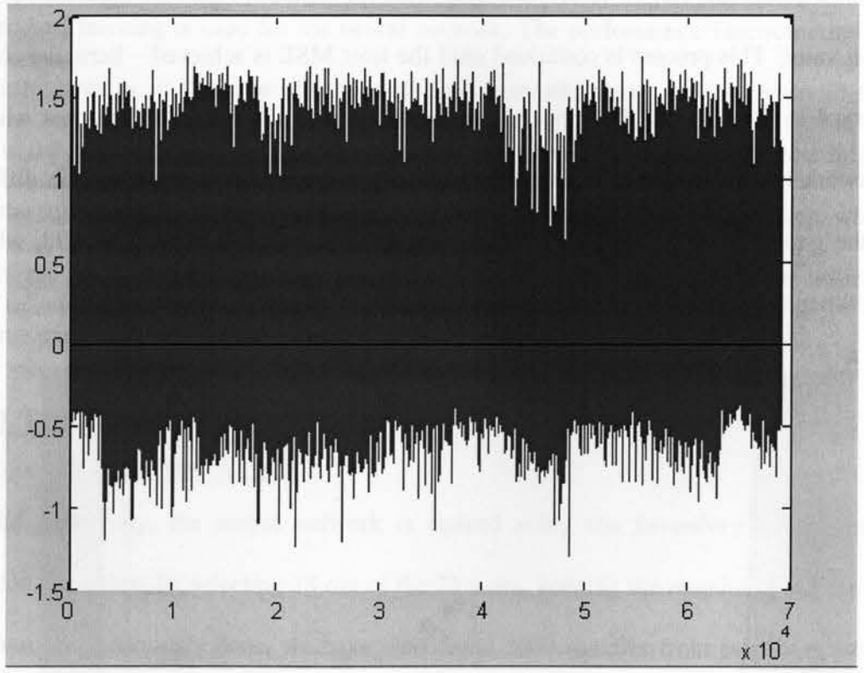


Figure 3.18: Error difference between the real and the generated fly times of the input training data.

3.5.3.4.3 Testing Phase

The testing phase is conducted in a manner similar to what was done in the case of the dwell time neural network, i.e.: (1) we use the remaining 5 users, and extract 3000 samples from each user, which are composed of the first key code and the second key code only, then we put them in a matrix and randomly mixed them to make the results unbiased, (2) we avoid showing the fly time for these 5 users to the network, (3) we train the neural network to calculate the fly time value for the 5 users on the basis of what it has learned from the above experience with the 18 users. Fig. 3.19 depicts the error difference, i.e. the difference between the output time values generated by the network and the actual time values from the data set of the 5 real human users. It is observed that the difference between the two sets of data is in the range [-0.3, 0.9], which shows that the network has been able to establish an association between the inputs and the outputs.

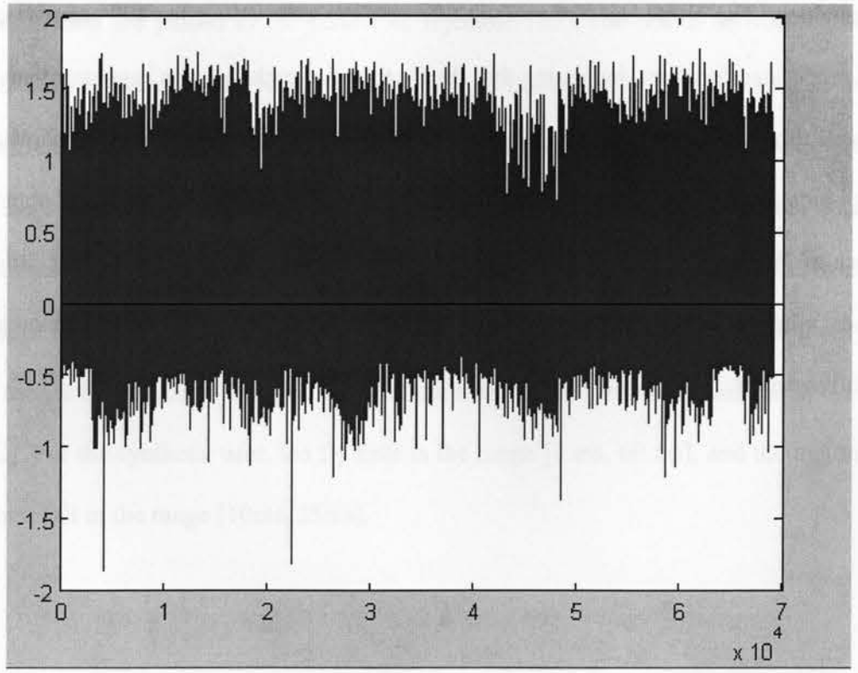


Figure 3.19: Error difference between the real and generated fly times.

3.5.4 Noise Injector

This module performs an important task, which is inserting the elements of noise in our generated data. It should be remembered that this generated data has been filtered out based on some guidelines as discussed in Section 3.2. A noise is typically defined as a value that falls outside specified ranges. From our data analysis phase, the data that were outside some specific ranges and were not meaningful in fulfilling the prescribed guidelines have been identified. These data are also considered when generating the noises. The Noise Injector works as follows:

The time values generated by the neural network are sent to the Noise Injector. Meantime, the user specifies and selects the percentage of noise he/she would like to insert into the data, for instance, 20% noise out of a total of 2000 samples – which corresponds to 400 samples having noisy time values. The Noise Injector module then sets the time values for 400 samples outside the ranges predefined for them, indicating that the fly time for such samples is greater than 60 ms and the dwell time values are greater than 30 ms. For one specific user, Fig. 3.20 shows some discrepancy between the fly times obtained in the absence of noise and the fly times obtained in the presence of 30 % noise. Indeed, in the fly time without noise, the observed upper limit is 60 milliseconds, whereas in the presence of noise, the upper limit for the fly time has increased to reach 98 milliseconds.

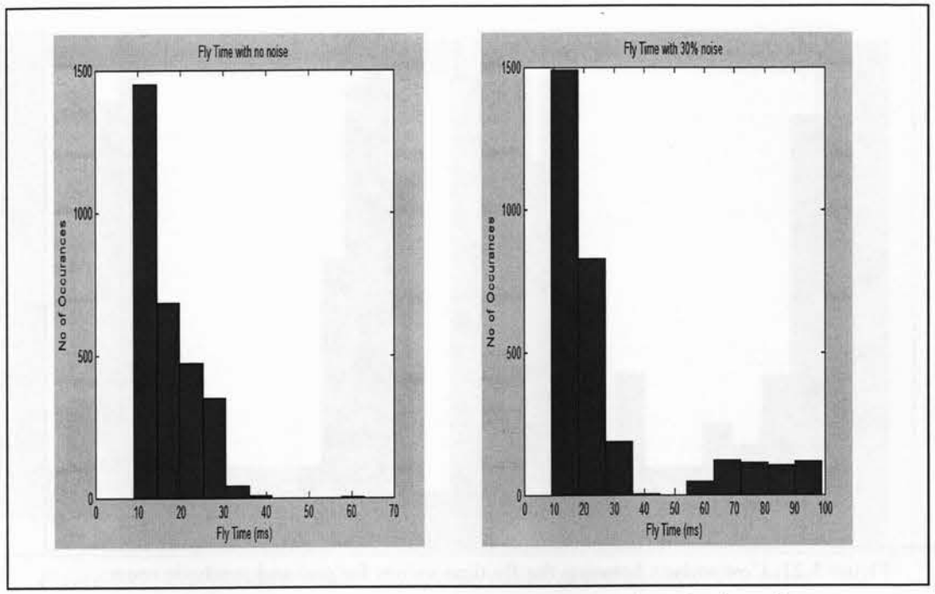


Figure 3.20: Fly times for one user before and after the noise insertion.

3.6 Generated Biometric Keystroke Data

The random data are passed to the behaviour injector. Then, the neural networks generate the biometric data and finally, the Noise Injector module injects the noise into the data to make it behave as real human data as close as possible. After all these steps, the biometric keystroke data is generated. In order to determine if the synthetic data behave the same as real data or how different these two might be, we run the following comparisons. We compare the synthetic fly time against the fly time for one specific real user. In Fig. 3.21, it is observed that the fly time of the real user is in the range [2 ms, 60 ms] and the majority of fly time values fall in the range [2 ms,15ms]. For the synthetic user, the fly time is the range [1 ms, 60 ms], and the majority of fly time values fall in the range [10ms, 25ms].

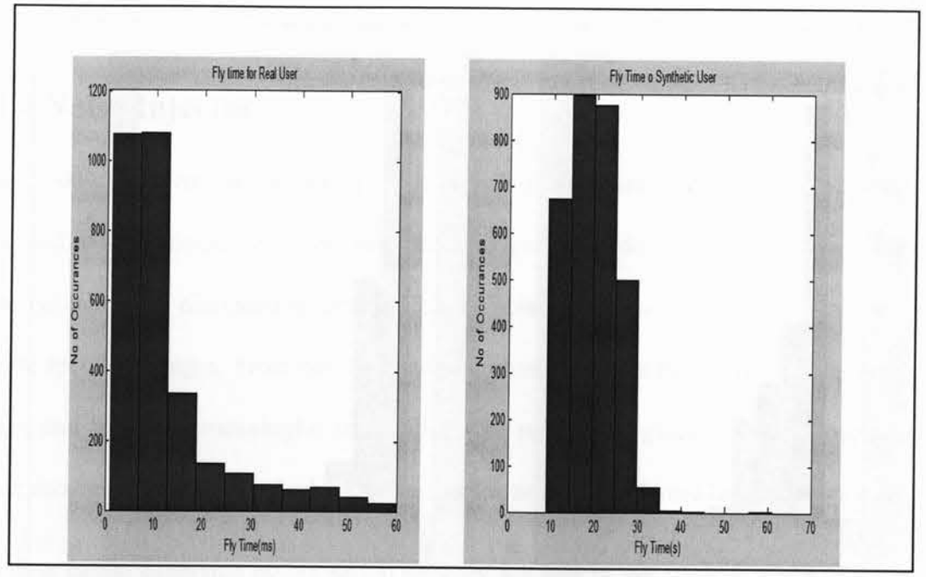
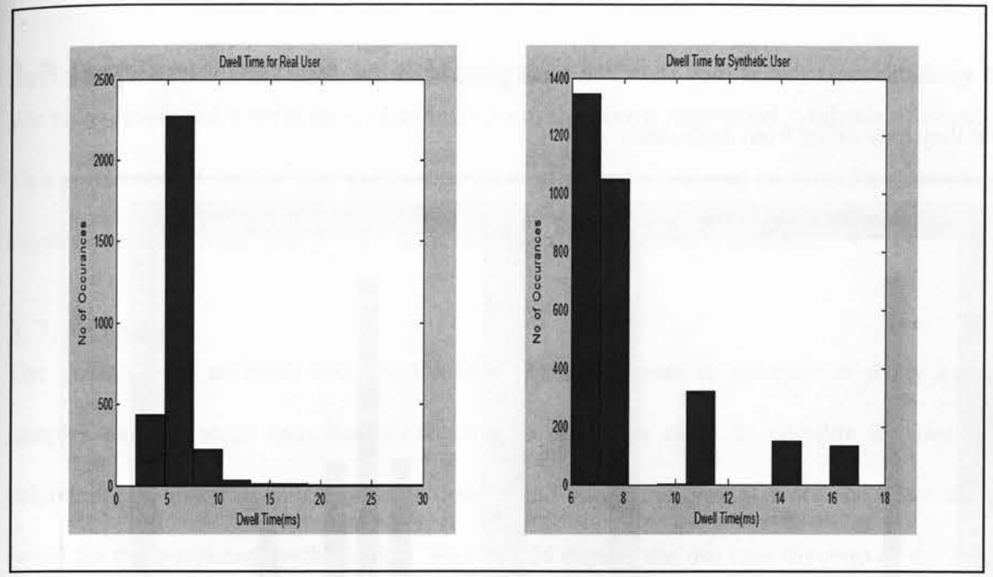


Figure 3.21: Comparison between the fly time values for real and synthetic users.

We also compare the dwell time values of real and synthetic users. In Fig. 3.22, it is observed that in case of real user, the majority of the dwell time values fall in the range [5ms, 10ms] with the highest value being 30ms, while for the synthetic user, the dwell values fall in the range [1ms, 8ms] in most of the cases with the highest value being 18ms. This means that these two users are not even closer to each other in this respect. But the both the real and synthetic users follow the same guidelines and have their own unique behaviour.



]

Figure 3.22: Comparison between dwell time values of real and synthetic users.

The goal is to judge how much a single user deviates from her own track, i.e. how much a single user is different from herself while pressing two keys one at a time. To this effect, it is observed from Fig. 3.23 that in terms of dwell time, a specific user does not show much difference in behaviour when pressing the first and second key. The first bar shows the values for the first key and the second bar shows the values for the second key. In this case, an almost equal number of occurences are observed for the dwell time values for both keys. Similar observations happen with the synthetic user, even though most of the values concentrate in the range [6ms, 7ms] for the case of the real user and in the range [7.8 ms, 8 ms] for the case of the synthetic user. Which indicates the fact that although the real and synthetic users show different behaviours but follow the same trends and guidelines as indicated in the beginnig of the chapter. Each real user has his own pattern for the dwell time vales of first and second keys. this pattern is the same for both the keys of one specific user but might be diffrent from other users. And the same trend is visible in

case of synthetic users that is they show the same pattern for the dwell time values of both the keys but they may differ from each other.

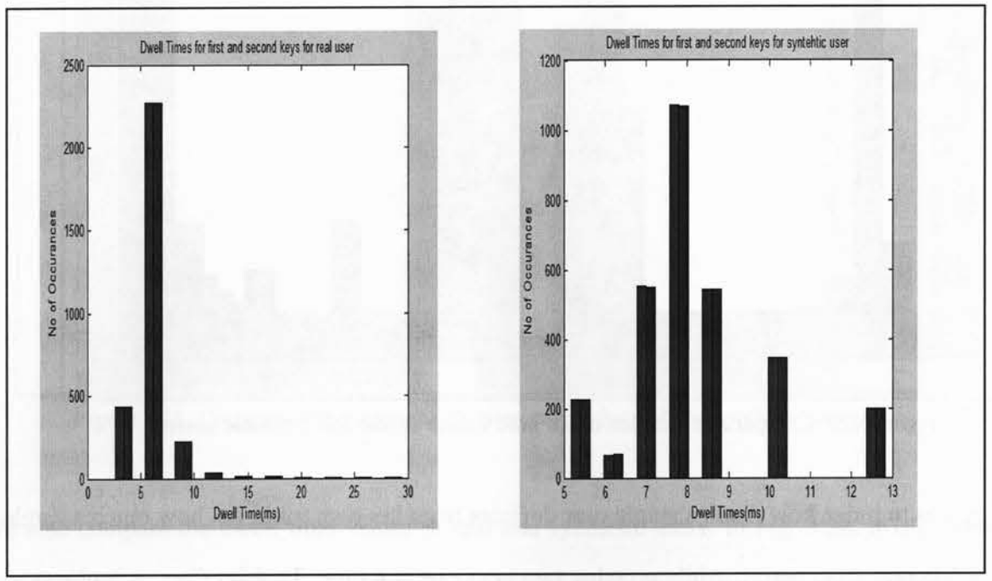


Figure 3.23: Comparing the real and synthetic user' first and second key codes' dwell times.

It should be noticed that we cannot compare the key codes for real and synthetic users since these are not generated by the neural network, but are set through the Behaviour Injector module by the user.

From the above comparisons, it can be concluded that the generated synthetic users are different from the real ones and do not have the same trends or patterns in their data, but they do fulfill the guidelines established in Section 3.2. The neural network learns from the real users' data and then is able to generate synthetic users, each having a unique random behaviour. Thus, we have been able to generate biometric keystrokes data for any specific user with unique behavioural properties.

3.7 Implementation and Design

We have developed a software tool to implement the above mentioned synthesis of keystrokes. This software tool implements the inverse biometric model and can be used for generating the keystroke actions without any a-priori knowledge of the program's modules.

3.7.1 Design

The goals of our software tool are twofold: (1) allow users to generate as many keystroke samples as they want such that these samples depict as close as possible the real human behaviour, (2) insert as much noise as desired and adjust the typing errors and other attributes easily for the generated synthetic data. Figure 3.24 depicts the use case diagram of the software tool and highlights the functionalities of the program. Some key features of the software tool are as follows. It allows the generation of synthetic data, the injection of noise in the generated data to make it behave as close as possible like real human data, the analysis and extraction of data for other purposes. The software tool is also designed in such a way that it can be evolved and extended to incorporate more functionality. The tool is designed in Microsoft .Net Visual Studio by using C# and MATLAB. We used COM objects to incorporate MATLAB with .NET framework.

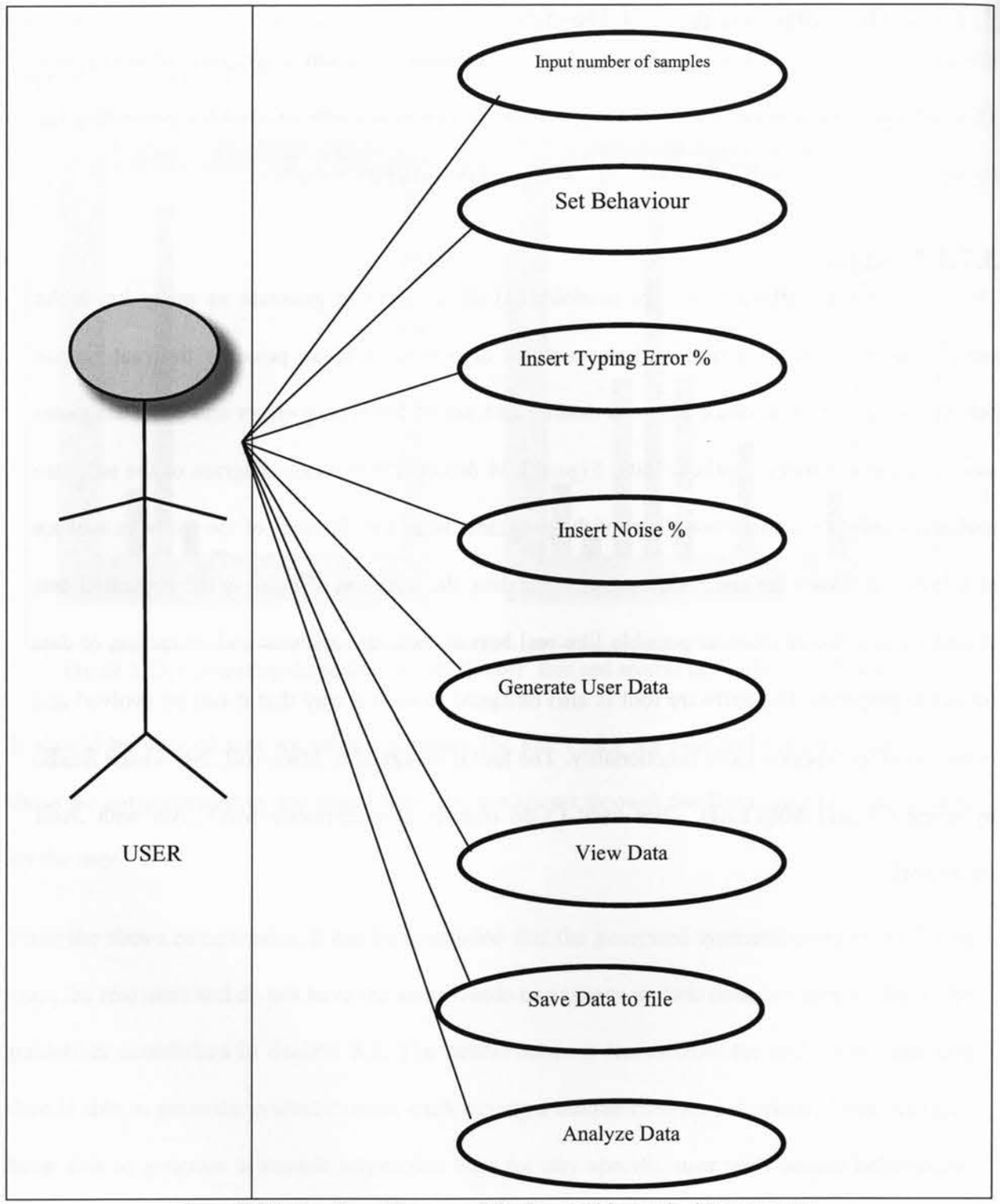


Figure 3.24: Use Case Diagram of the inverse biometric data generator.

Input Values

This interface helps the user to provide the inputs to the program with respect to the number of keys that the user would like to include in the typing sample.

Set Behaviour

This module allows the user to specify the behavioural characteristics of the sample, for instance, the percentage of letters, numerals, control keys and assisting keys.

Insert Typing Error

This module allows the user to specify any percentage of typing errors he/she would like to include in the typing sample.

Insert Noise

This module allows the user to insert the percentage of samples he/she wants to be infected with noise in order to make the synthetic data exhibit the true human behaviour.

Generate User Data

This module allows the user to actually generate the data with all the above mentioned specifications by using the neural networks implemented in MATLAB.

View the User

This module allows the user to view the data properly fixed in a grid with the first key code, the second key code, the fly time, the dwell time of the first key and the dwell time of the second key.

Save Data in a File

This module allows the user to save one user in one file with all the above mentioned five columns. The name of the file would be the word "user" followed by the user's number.

Analyze User

This module allows the user to analyze the properties of the generated data graphically by using the Zed bar graphs. To this effect, the user must select two synthetic users in order to compare their data.

We use .Net as programming platform and C # as programming language. Since C# is highly object oriented, we have designed classes and objects in our program that implement this feature. Fig. 3.25 depicts those classes and objects, as well as their relationship to each other. .

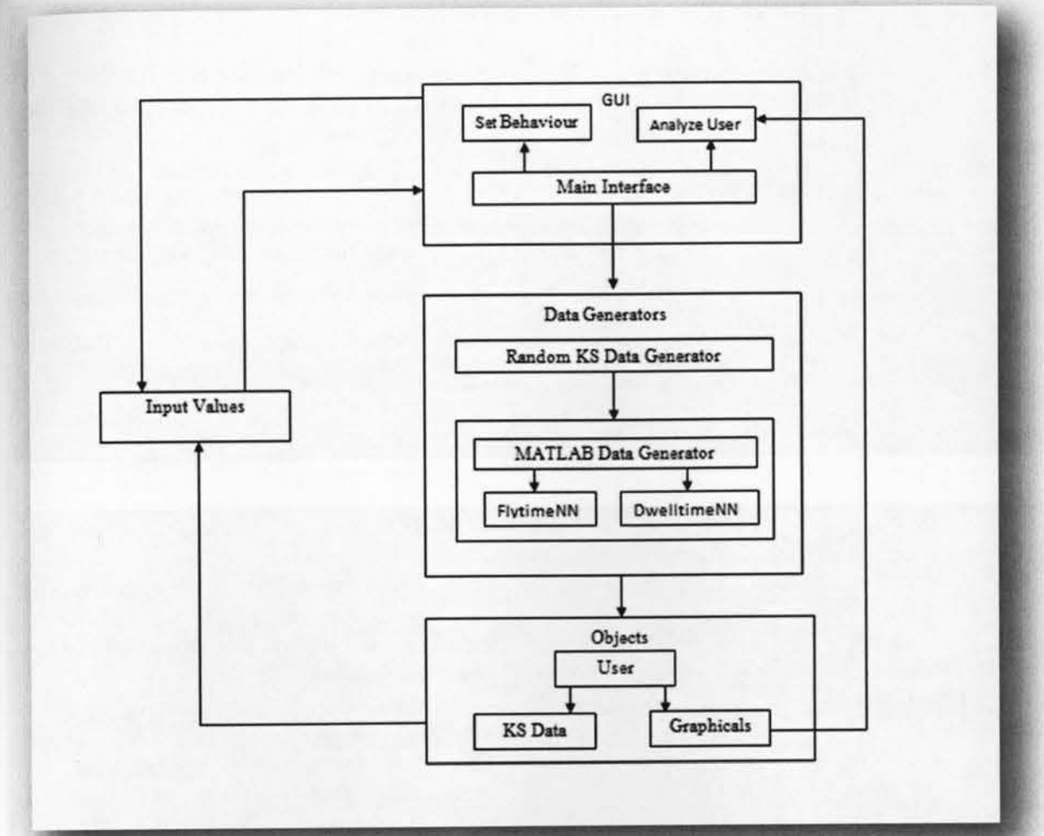


Figure 3.25: Package diagram showing the classes and their relationships in the software tool.

Each package and its classes are described in Figure 3.26.

	CLASSES	DESCRIPTION
Package : GUI	Main Interface	This class implements the main interface for the user, which will be used to start the program and take the inputs.
	Set Behaviour	This class implements the interface that allows the user to set the behavioural attributes of the typing sample.
	Analyze User	This class implements the graphical interface that will allow the user to analyze different users graphically.

	CLASSES	DESCRIPTION
Package:Data Generators	Random Keystroke Data generator	This class generates the data that consists of the key codes according to the user's specification and gives this as an input to the next generator.
	MATLAB Data Generator	This class has all the implementation to generate the synthetic data. It calls the neural networks implemented in MATLAB and retrieves back the results.

	CLASSES	DESCRIPTION
Package: Objects	Users	This class implements all the functions and properties of one specific user.
	KS Data	This class implements all the functions and properties of the keystroke data for any user.
	Graphicals	This class implements all the functions and properties of the Zed graphs used to analyze the user.

Figure 3.26: Packages and their classes

3.7.2 Interface and Features

In this section we will show the snapshots of the interfaces used in our program.

3.7.2.1 Main Interface

At the beginning of the program, the first windows form that appears is the main interface of the program. This interface is shown in Fig. 3.26. It has the following features:

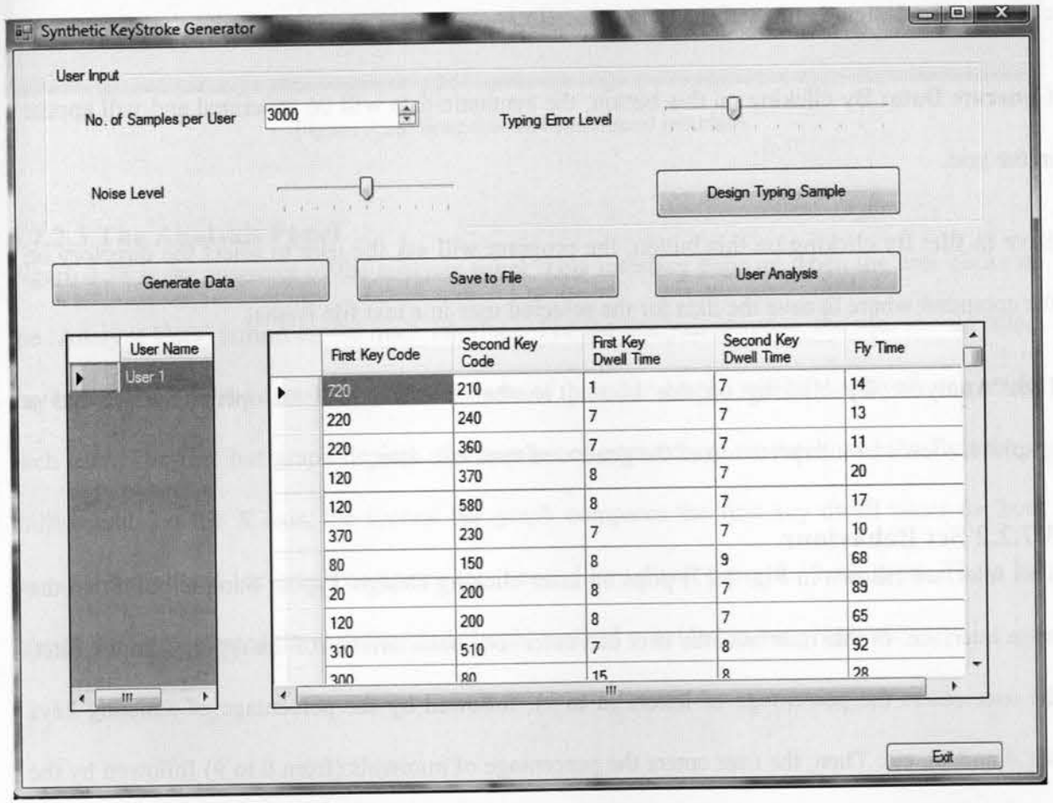


Figure 3.27: Snapshot of the main interface.

Number of Samples per User: here, the user enters the number of samples to be generated

Typing Error Level: here, the user can adjust through this track bar the percentage of typing error to insert. This percentage ranges from 0 % to 100 %.

Noise Level: here, the user can adjust through this tract bar the percentage of noise to be inserted in the synthetic data. This percentage ranges from 0% to 100%.

Design Typing Samples: by clicking on this button, a new interface will pop up where the user can setup the design features of the typing sample.

Generate Data: By clicking on this button, the synthetic data will be generated and will appear in the grid.

Save to file: By clicking on this button, the program will ask the user to select the directory on the computer where to save the data for the selected user in a text file format.

User Analysis: By clicking on this button, another interface will be opened, which has a graphical view and interpretation of the generated synthetic data.

3.7.2.2 Set Behaviour

This interface (shown in Fig. 3.27) pops up after clicking Design Typing Sample button on the main interface. In this interface, the user can enter the characteristics of the typing sample. First, the user enters the percentage of letters (a to z), followed by the percentage of assisting keys &,(,# and @, etc. Then, the user enters the percentage of numerals (from 0 to 9) followed by the percentage of assisting control keys (F1, F7, etc). In order to make the program user friendly, the user can also click on the "Assign Random Values" button and the program would assign random percentages to the above mentioned parameters. Before proceeding further, the program also ensures that the user has entered all the 100% values.

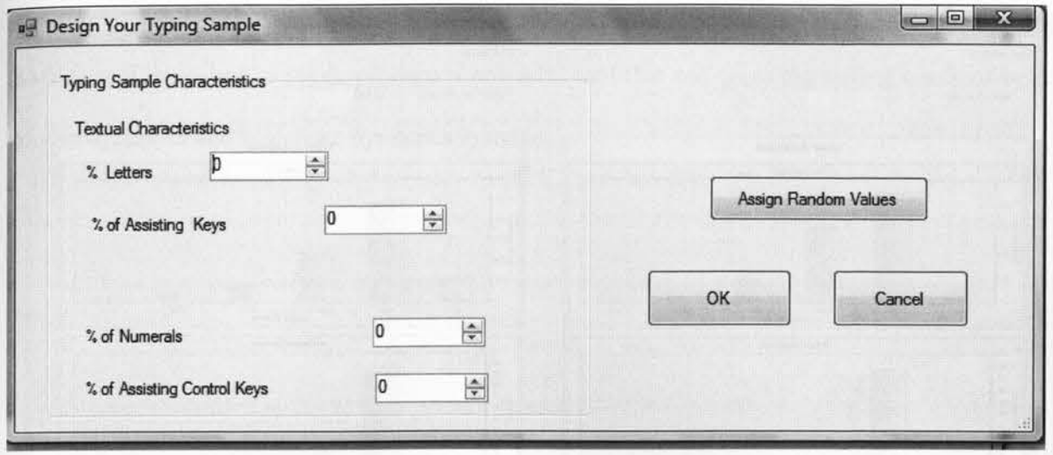


Figure 3.28: Snapshot of behavioural interface.

3.7.2.3 The Analysis Panel

Figure 3.28 is the snapshot of the analysis panel. This interface pops up when the user clicks on the 'Analyze User' button on the main interface. For the analysis purpose, the user should select any two users from the drop down list. Two sets of three bar graphs will be displayed, one set for each user. The first bar graph depicts the number of occurrences on the Y and the fly time in milliseconds on the X axis, the second bar graph compares the first key dwell times for both users, and the third bar graph compares the second key dwell times for both users. These bar graphs are implemented through the 'Zed Graph' method provided in our program.

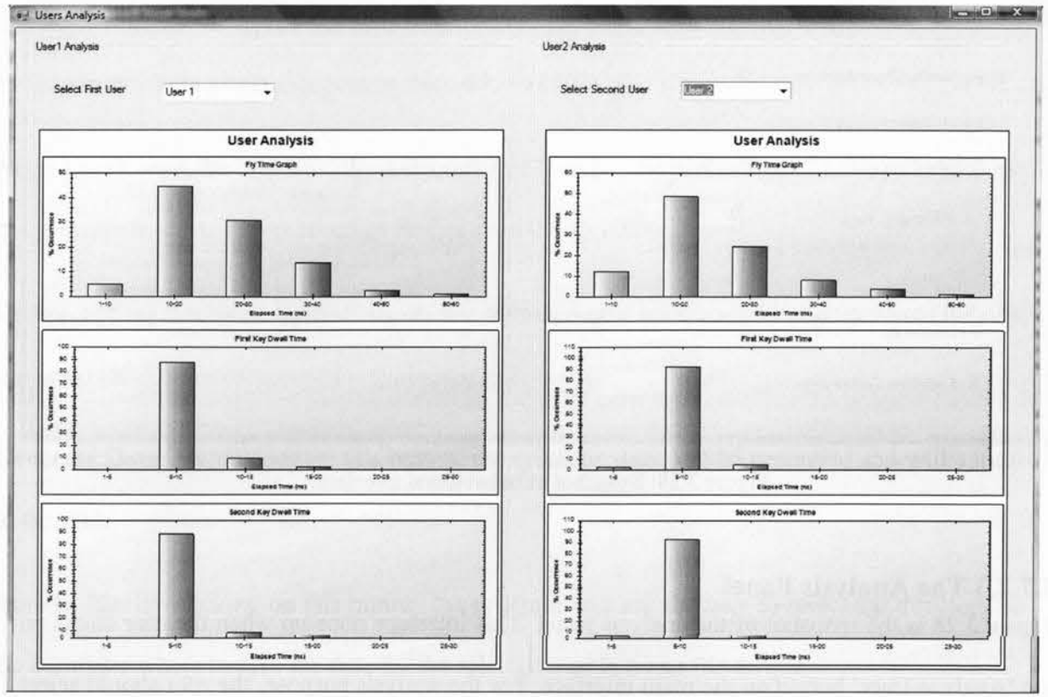


Figure 3.29: Snapshot of the analysis panel.

3.8 Summary

We have developed a new inverse biometric model, in the form of a simulator, for generating the synthetic keystroke data. These goals are achieved by taking some raw data from the real users and analyzing it, and then train some neural networks on that data which leads to synthetic keystroke data that can mimic as close as possible the behaviours of real human data. Comparisons between the generated and real data are given to validate the system. This chapter has also discussed the design and features of the software tool that we have developed to implement the inverse biometric model, describing how the tool performs the tasks of synthetic data generation, and analysis of the inverse biometric data. This tool used as a framework of choice for testing

existing biometric systems or those currently under development. It can also be merged with the SMAG tool described in [4] to produce a powerful tool that can serve the testing needs of both mouse dynamics and keystroke dynamics systems.

Chapter 4
Evaluation

This chapter discusses different evaluation approaches that have been proposed in the literature, and then presents our evaluation procedures and results.

4.1 Context

In order to judge the quality of any model or the data generated by it, the generated data must undergo some validations, by comparing them against real data through various validation techniques. To date, the focus of the research on biometric synthesis has primarily been on the design of data generators rather than being on the validation of the proposed models themselves [8, 15]. As a result, there is a great deal of consensus about how to empirically or even analytically validate synthetic biometric generators. One way to achieve validation is by comparing the biometric data with the real data visually [19]. But this cannot be applied in the case of synthetic keystrokes as they do not have a visual signature like iris patterns or fingerprints do. Therefore, it is necessary to find other types of validation schemes. Another approach which can be used for the same purpose is to compare the recognition performance achieved for some existing analysis models using synthetic biometric samples or alternatively real biometric samples [9, 30]. This strategy is probably best suited for us since we can compare the generated synthetic keystroke data against the real data on the basis of the performance results they have generated when tested on the same analysis model. It is then expected that acceptable synthetic keystroke data would yield performance results closer to that of real data when applied to a given analysis model.

Apart from using analysis models for validating synthetic keystroke data, statistical testing techniques can also be useful for this type of data. Such testing approaches consist of comparing the statistical distributions of real data against the statistical distributions of synthetic keystroke data. Example methods that can be used to achieve this kind of testing approach are: Chi-squared tests, Kolmogorov-Smirnovtests, and Anderson-Darling tests. Graphical statistical methods such as the Quantile-Quantile (Q-Q) plot can also be used for the same purpose to achieve similar type of results. The Q-Q plot has been used by Daugman to establish that matching scores for a collection of iris-scans fitted well the beta-binomial distribution [23]. However, all the aforementioned approaches rely on hypothesis testing. Hypothesis testing is a common method of drawing inferences about a population based on statistical evidence from a sample [2]. These testing approaches associate certain assumptions with the data being used. Behavioural biometrics such as keystroke dynamics (that we deal with in this thesis) are characterized by their strong variability, which may conflict or contradict many of the assumptions required by existing statistical testing techniques [4]. We argue that effective statistical testing for synthetic keystrokes data generation would require the design of new testing paradigms which address the specific nature of the considered synthetic keystrokes data. Based on the above considerations, the rest of the Chapter describes two types of evaluations: (1) we compare the generated synthetic keystrokes data against real human data using the Kolmogorov-Smirnov statistical testing method, implemented in MATLAB (see Section 5.2), the goal being to show that synthetic keystrokes data have behavioural attributes closer to that of real human data, but are different from real human data, and (2) we apply and compare the recognition performance results obtained by applying synthetic keystrokes data and raw data to an existing analysis

model. Our results confirm that the performances of synthetic keystrokes data and real human data are very similar.

4.2 Kolmogorov-Smirnov Testing

The results obtained in Section 3.6 are from comparing one real user against one synthetic user. We can clearly infer that the two users are quite distinguished from each other. But these results hold for two users only, one being real and the other being synthetic. We are not sure about the rest of the real users or in other words we don't know about an overall impression of the all the real users that how close they are to the generated synthetic users. For this purpose we conducted a Kolmogorov-Smirnov (KS) test, implemented in MATLAB under the name 'kstest2' function. The experiment was carried out as follows: (1) we collected 3000 random keystroke samples from each of the 23 real users and merged them into one data set called the Real Data Sample, (2) we then generated 150 synthetic users, and took 3000 random keystroke samples from each of them and merged them into another data set, called the Synthetic Data Sample. For this type of test, the Real Data Sample and the Synthetic Data Sample may not contain the same number of elements. Sample data from these two sets is shown in Table 5.1

The purpose of the two-sample KS test is to check whether the two samples come from the same continuous distribution or not. The definition for this test given in the Matlab is a function H = kstest2(X1, X2) and it performs a two-sample Kolmogorov-Smirnov test to compare the distributions of values in the two data vectors which it named as X1 and X2 and each of length n1 and n2, respectively. They represent random samples from some underlying distributions. The null hypothesis for this test is that X1 and X2 are drawn from the same continuous distribution and the result H is 1 if X1 and X2 do not belong to the same distribution or 0 if that do belong to the same continuous distribution. For each potential value X, the KS test compares the

proportion of X1 values less than X with the proportion of X2 values less than X. The kstest2 function uses the maximum difference over all X values as its test statistic [24]. Mathematically, this can be written as:

$$MAX (|F1(X) - F2(X)|)$$

where F1(X) is the proportion of X1 values less than or equal to X and F2(X) is the proportion of X2 values less than or equal to X [24].

Real Data			Synthetic Data		
First Key	Second Key Code	Fly Time	First Key	Second Key Code	Fly Time
490	240	11	**770**	720	18
190	300	11	**720**	751	24
330	240	5	**310**	430	13
320	570	5	**80**	711	31
180	500	2	**430**	630	10
190	570	3	**20**	660	30
210	240	5	**770**	520	26
570	420	25	**40**	10	16
140	480	42	**620**	811	7
570	310	6	**771**	771	16
200	300	1	**660**	290	13
170	180	8	**620**	90	42
310	570	4	**100**	620	27
570	210	8	**811**	310	23
200	190	2	**650**	630	17
180	320	8	**630**	560	18
190	230	11	**711**	650	21

Table 4.1:Sample Data for KS Test

We performed the testing on both the fly time and dwell time values. We first compared the fly time values of real and synthetic users, and then we compare the dwell time values of real and synthetic users. In all tests performed, the results came out to be 1, which means that the

hypothesis that the samples are taken from the same distribution can be rejected. Just to strengthen the results obtained from the first test, we also performed another experiment, similar to the first one, but this time, using real data only. We collected 3000 samples from each real user, ensuring that these samples are different from the one taken earlier. Then we merged them in a similar manner as we did above, into one real data set and then compared this real data set against the earlier Real Data Sample. The result came out to be 0, meaning that the two data sets being compared are from the same distribution. This is obviously expected, but, it verifies that the kstest2 function is able to detect similarity as well as difference. The 'kstest2' function gives the 0 value for the null hypothesis.

These tests confirm that the synthetic users generated by our inverse biometric model have completely different behaviour from the real users that were used to train the neural networks. In other words, the synthetic users that are generated are not perturbed instances of the data that was collected from real users, but in fact, they exhibit their own behavioural properties. The actual behaviour and behavioural characteristics are to be distinguished. The synthetic data that is generated has behavioral characteristics similar to real data, but the actual behavior of synthetic users is different from that of real users. For instance, one synthetic user may take longer dwell time for all the samples than a real user would do, or the fly times may be shorter for real users. It is completely up to us as to what behavior we want to insert into a user. Synthetic users are different in their behaviours because of the fact that every real user is also different in its own behaviour. Since synthetic users are created on the basis of the guidelines derived from a comprehensive analysis of real users, they have the same behavioural properties and attributes than real users.

Fig. 4.1, compares the reference signature against the real data for user 1 for fly time, while Fig. 4.2 compares the reference signature for user 1 against the real data for user 2 for fly time. It can be observed that when a user is compared with itself (Fig. 4.1), the difference between the two data sets is not significant, but, when a user is compared against another user (Fig. 4.2), the difference in the patterns is quite significant. This observation validates our research motive that each user has his/her own characteristics, different from that of other users, and these characteristics can be used to identify that user which is further proved in the next section.

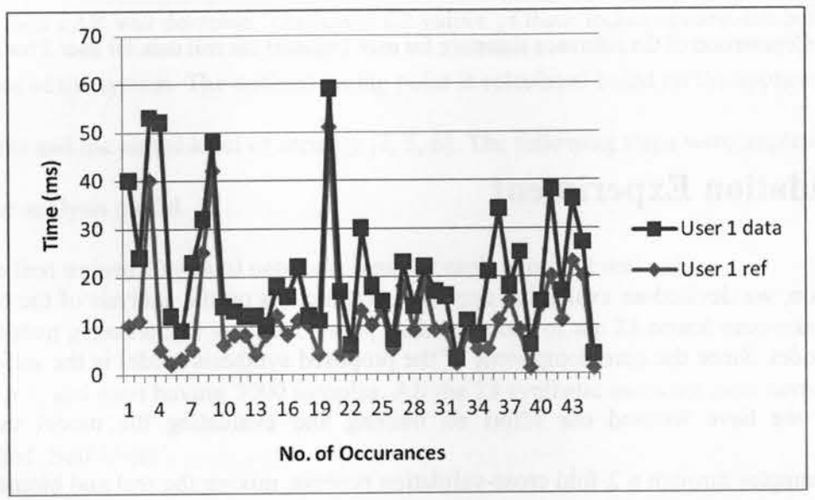


Figure 4.1: Comparison of the reference signature against the real data for user 1 for fly time.

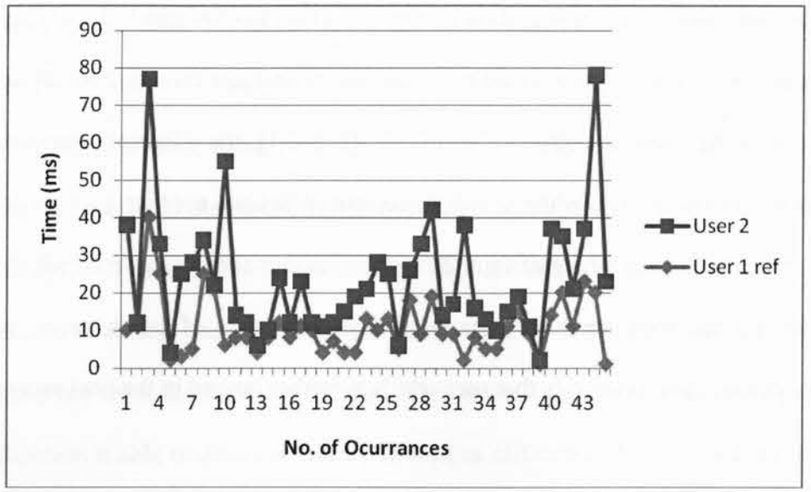


Figure 4.2: Comparison of the reference signature for user 1 against the real data for user 2 for fly time

4.3 Validation Experiment

In this section, we devised an evaluation approach that focuses on the analysis of the biometric synthesis model. Since the core component of the proposed synthesis model is the self-learning component, we have focused our effort on training and evaluating the model using real biometrics samples through a 2-fold cross-validation process, mixing the real and biometric data for comprehensive testing as done in [4]. We use the following definitions introduced in [3]:

False Acceptance Rate (FAR): This metric is defined as the ratio between the number of occurrences of accepting a non-authorized user compared to the number of access trials.

False Rejection Rate (FRR): This metric is defined as the ratio between the number of false alarms caused by rejecting an authorized user compared to the number of access trials.

Receiver Operating Characteristic Curve (ROC): The ROC curves are obtained by plotting the FRR values versus the FAR values based on threshold limits. The purpose of the ROC curves is to show the relationship of the FRR and FAR rates at variable threshold limits.

The optimal operating point depends on the relative cost of a false acceptance compared to that of a false rejection [1]. If the system is designed to minimize FAR to make the system more secure (for instance, for access control systems and intrusion detection systems purposes), FRR will increase [2, 5, 6]. On the other hand, if the system is designed to decrease FRR (for instance, in the case of digital forensic systems) by increasing the tolerance to input variations and noise, then FAR will decrease. The lower the values of these indicators are, the better is the performance of the system. The optimal tuning point is calculated based on the application requirements and the aimed level of security [2, 5, 6]. The following steps were implemented to validate our analysis model.

1. We first trained 23 neural networks, one for each of our 23 real users.
2. We then generated 23 synthetic users, each from one of the 23 neural networks trained in Step 1, and each having 3000 samples. All the 23 synthetic users are now saved in a pool called 'Self Users'.
3. We made a pool of 30 synthetic users, each having 3000 samples from the neural networks trained in Step 1. But, we assured that none of the synthetic users is produced from the network trained for that particular user. For instance, for the pool of user 1, we have produced 30 synthetic users trained using all neural networks, except the user 1 's neural network.

4. Similarly, we made 23 pools, each of 30 users for all the 23 users, keeping in consideration the fact that the pool for any user should not have a synthetic user generated from him/her.
5. Now, to calculate the FAR, we compared each user reference with the pool created for that user. Each user is attacked by 30 other users having 3000 samples. For instance, to calculate the FAR for user 1, we simulated the user 1's network with his pool of 30 synthetic users whose target data were already generated through their neural networks. Then, we calculated the deviation between the targeted time and the simulated time, then we decided whether this synthetic user is user 1 or not. The way that we designed the pool has revealed that this synthetic user is not user 1. By calculating the percentage of deviation and changing this threshold level, we determined the FAR for user 1.
6. We repeated the process in Steps 1 to 5 for all the 23 users, comparing them with the pools designed for them. We then performed the whole scheme for fly time and dwell time. Afterwards, we calculated the average of the FAR values obtained from these experiments to determine the FAR value of the system.
7. To calculate the FRR, we compared each user with himself. For instance, for user 1, we simulated user 1's network with the synthetic user 1, which was generated from the same network earlier and was saved in the so-called 'Self Users' pool. Then, from the calculation of the deviation between the targeted time and the generated time, we determined if this user can be identified as user 1 or not (as a matter of fact, this user is user 1). Afterwards, we changed the percentage of deviation and based on this threshold, we determined the FRR for this user. Then, we repeated the whole exercise for all 23 real

users by comparing them with themselves in terms of dwell times and fly times, and then, we determined the FRR of the system as the average of the obtained FRR values.

It should be noticed that while doing these experiments, we have injected 10 % noise in our synthetic users' data. To calculate the deviation, we used the formula in [2], i.e.

$$D = abs\ (\ di\ (n,3) - ttx(n)\ ) * 100 / ttx\ (n)$$

where *abs* denotes the absolute value, *di* is the targeted time and ttx is the simulated time for of dwell or fly time values. The result is stored in the form of an array, which has 3 columns, referred to as the reference user id, the current user id, and the result of comparison, where the reference user is the user who is being tested and the current user is the user with whom the reference user is being tested. Fig. 4.3 and Fig. 4.4 depict our evaluation approach for calculating the FRR and FAR respectively.

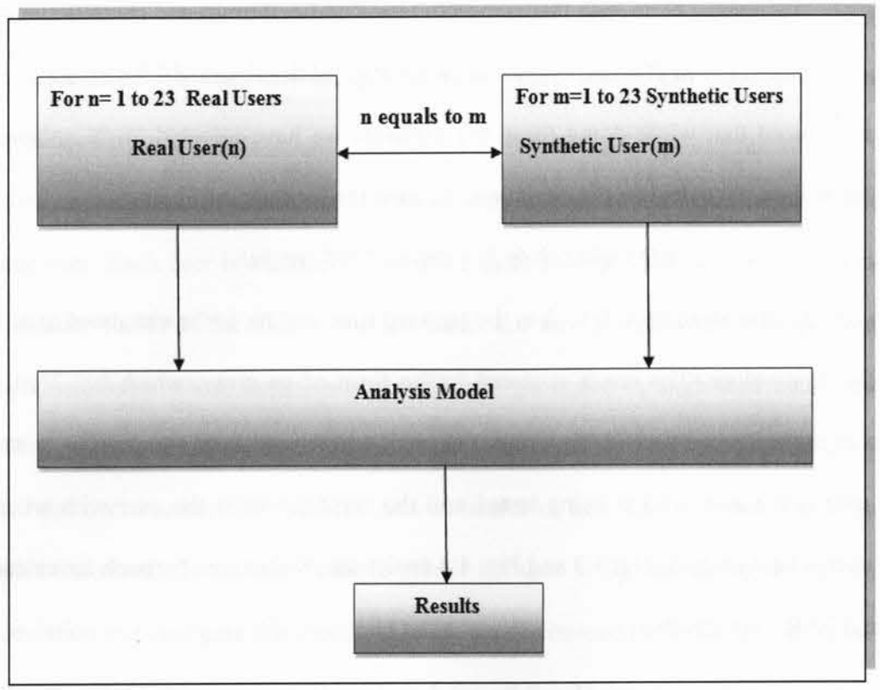


Figure 4.3: Validation Experiment – Case of FRR.

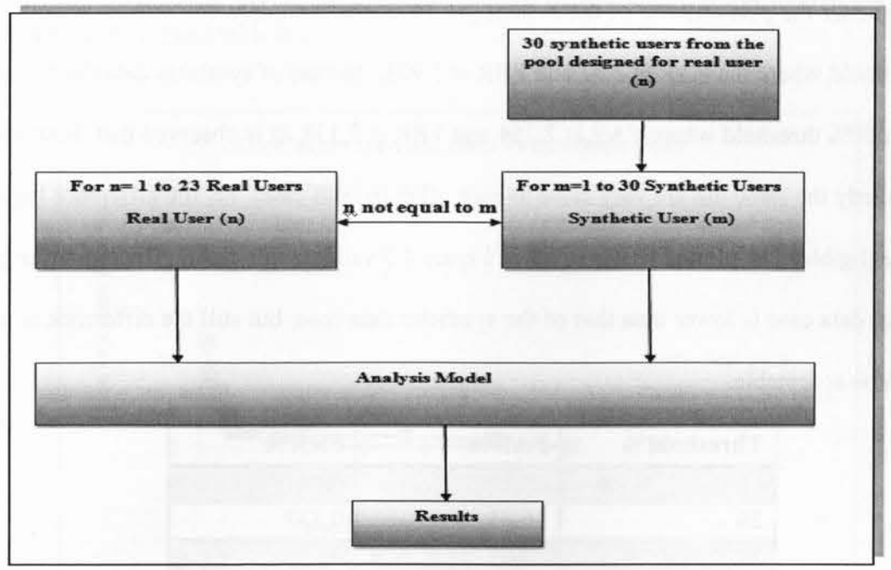


Figure 4.4: Validation Experiment – Case of FAR.

After conducting the above mentioned testing experiments, we have achieved an optimal performance of FAR = 3.806 % and FRR = 1.962% using a 50 % threshold limit.

We repeated the above experiment with a new set of real users. We obtained 20 real users' data from a data set originally collected in [1] and used that data to evaluate FAR and FRR for real users using the same analysis model. The purpose is to compare the real data against the synthetic data that we have previously generated. We used new set of real users for ensuring that the analysis model does not generate any bias results based on the previously used real data. Table 4.2 and Table 4.3 shows the FAR and FRR based on thresholds. The threshold values are taken from 10% to 100% with an interval of 10, thus, there are total 10 points for each FRR and FAR. We then used the data presented in these tables to plot the ROC curves for both real and synthetic data in Figure 4.5. In a ROC curve, there is usually a point at which the FRR and FAR are likely equal. At the point where the FRR and FAR coincide, the lowest the value of this

point, the better the performance of the system [4]. In case of real data, the crossover point is at 50 % threshold where the FAR is 2.92 and FRR is 1.903. In case of synthetic data the crossover point is at 60% threshold where FAR is 2.754 and FRR is 2.138. It is observed that these values are not exactly the same but are very close to each other in both cases, i.e. the difference between them is negligible. The plotted ROC curves in Figure 4.5 validate this claim. The crossover point for the real data case is lower than that of the synthetic data case, but still the difference is small enough to be acceptable.

Threshold%	FAR%	FRR%
10	25.17	0.091
20	16.45	0.137
30	12.01	0.158
40	5.51	0.813
50	2.092	1.903
60	2.305	2.095
70	1.842	3.071
80	0.854	5.610
90	0.509	9.135
100	0.430	11.804

Table 4.2: Analysis Results of Real Data

Threshold%	FAR%	FRR%
10	27.15	0.109
20	21.63	0.837
30	15.08	1.390
40	07.51	1.72
50	3.806	1.962
60	2.754	2.138
70	2.201	4.3409
80	1.609	6.601
90	0.719	8.013
100	0.531	12.35

Table 4.3: Analysis Results of Synthetic Data

The following are the ROC curves for real and synthetic data plotted with the help of the data presented in table 4.2 and table 4.3.

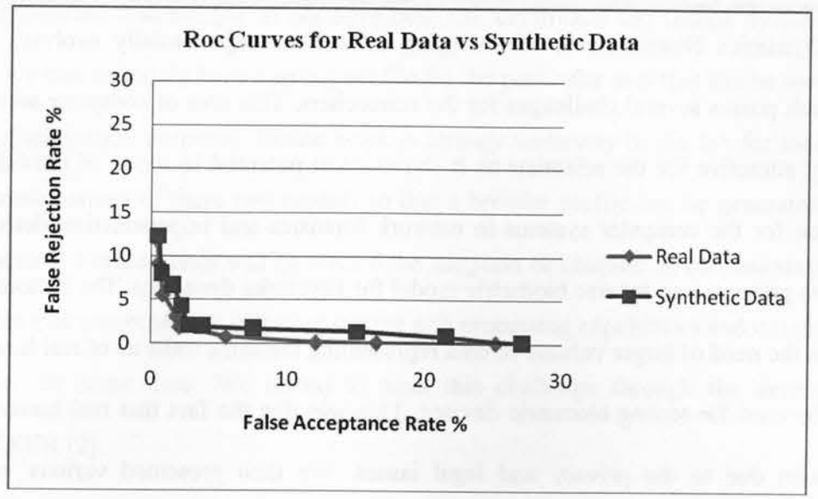


Figure 4.5: ROC Curves of Real Data vs. Synthetic Data

4.4 Summary

In this chapter, we have evaluated our inverse biometric model for keystroke dynamics based on two different validation schemes. First, we have tested both the synthetic and real data using some statistical testing approaches. These tests showed that the two data sets are not the same and do belong to the same distribution. These tests also confirm that each synthetic user possess its own characteristic behaviours which is not the same as that of the real human behaviour, but is based on the same foundation as that of the real user. We also compared the real human data against the synthetic data using the same analysis model, and we obtained some close resemblance between the FAR and the FRR values.

Chapter 5
Concluding Remarks

Keystroke Dynamics Biometrics is an emerging as well as exponentially evolving area of research which posses several challenges for the researchers. This area of computer security has become very attractive for the scientists as it shows great potential in terms of providing user authentication for the computer systems in network forensics and impersonation detection. In this thesis we presented an inverse biometric model for keystroke dynamics. The bottom line for this model is the need of larger volume of data representing the same traits as of real human data which can be used for testing biometric devises. This was due the fact that real human data is hard to obtain due to the privacy and legal issues. We then presented various validation techniques from different paradigms to ensure that the synthetic data does have behavioural properties similar to the real data. We showed many comprehensive comparisons at different stages of our research to show the properties of synthetic data in comparison of the real data. We also explained the design and implementation details of the model in the later parts of the thesis. This model can help the researchers and industrialists interested in keystroke dynamics technology to improve and test their recognition systems comprehensively.

5.1 Future Work

In the future, we intend to test and improve our own keystroke biometric analysis model. We intend to use this model in the areas of network forensics and continuous user authentication. We have inverse mouse dynamics implemented and evaluated in [4] by our fellow researchers in our lab. Since both the technologies are similar in many aspects and can be used effectively to achieve the goals of computer security, we are planning to combine the two of them. By combining mouse dynamics with keystroke dynamics, we can have a Multimodal Biometric

Identification system. A big advantage of having a Multimodal Biometric System as compared to a Unimodal Biometric System is that a multimodal system can significantly narrow down the number of possible matches [4]. If we have both the keystrokes and mouse dynamics for the same user we can definitely have a strong profile for the particular user that can be used in future for user authentication purposes. Hence, work is already underway in our lab for looking at the possible combinations of these two models so that a broader profile can be generated about the user's behaviour Future trends will be toward the adoption of ubiquitous, continuous monitoring technologies that combine cost-effective storage and processing capabilities and mitigate privacy concerns at the same time. We intend to meet this challenge through the development of BIOTRACKER [2].

5.2 Improvements

Although we tried to develop our inverse biometric model for keystroke dynamic very cautiously and scientifically but there is always room for improvement. Particularly when the area of research is evolving like ours one always has to be careful about new developments and requirements in such fields. Since we also have to deal with human behaviour we have to have a wide variety of human behaviour to comprehensively test our system which makes it a challenging task. The issues which currently can be further investigated are as follows:

1. What is the effect of being a right handed or a left handed user in case of keystroke dynamics? We would like to further investigate the point that how much being a right or left handed person contributes towards the validity of synthetic keystroke data.
2. How does the typing speed of a user affect the synthetic data produced by our model? Since we know that with the passage of time the users become more proficient and fast in

their typing speed, we would like to know that if it also effects their typing behaviour or not?

3. What will be the affect of increasing age on the typing skills and typing speed? This point is important for us as we can discover how much the user's typing attributes and behaviours changes with the age so that we can also incorporate those changes in our model.
4. As we used FRR and FAR for the performance measurement, we would like to further improve these indicators by doing more and more research and bring any improvement that is possible.
5. Can we improve the performance of the analysis model by having a better system for detection and noise filtration?

All these issues need to be addressed in future. We have research going on in this regard in our labs. We presented only a few aspects of keystroke dynamics and its usage in computer systems' security especially for user authentication. Since it is an emerging field much more is yet to come. We would try to further analyze and study even the minor details of the human behaviour in order to enable our inverse biometric model to generate more accurate and realistic keystroke actions.

References

1. Ahmed, A.A.E., Traore, I. "*A New Biometric Technology Based On Mouse Dynamics*", IEEE Transactions on Dependable and Secure Computing, pp 165-179, Vol. 4, No. 3, July-September 2007,.

2. Ahmed, A.A.E., Traore, I. "*Detecting Computer Intrusions Using Behavioral Biometrics*", Proceedings of the Third Annual Conference on Privacy, Security and Trust, St. Andrews, New Brunswick, Canada, pp. 91-98., October, 2005.

3. Ahmed, A.A.E., Traore, I. "*System And Method For Determining A Computer User Profile From A Motion-Based Input Device*", PCT patent application with the World Intellectual Property Organization, filed by the University of Victoria (PCT/CA2004/000669) filing date: 3 May 2004 priority date: 2 May 2003.

4. Nazar, A.,Traore, I., Ahmed, A.A.E." *Inverse Biometrics For Mouse Dynamics*", International Journal of Pattern Recognition and Artificial Intelligence, 16:37, September 16, 2007.

5. Guneti, D., & Picardi, C. "*Keystroke Analysis Of Free Text*". ACM Transactions on Information and System Security, Vol. 8, No. 3, pp. 312-347, (2005).

6. Bergadano, F., Guneti, D., Picardi, C. "*User Authentication Through Keystroke Dynamics*". ACM Transactions on Information and System Security, Vol. 5, No. 4, pp. 367-39(2002).

7. Monrose, F., & Rubin, A. D. "*Keystroke Dynamics As A Biometric For Authentication*". Future Generation Computer Systems, 16(4), 351-359, (2000).

8. Obaidat, M. S., & Sadoun, B. "*Verification Of Computer Users Using Keystroke Dynamics*". *IEEE* Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, *27*(2), 261-269 , (1997).

9. Bleha, S., Slivinsky,C., & Hussein. B. "*Computer-Acess Security Systems Using Keystroke Dynamics*". IEEE Trans. Patt. Anal. Mach. Int. PAMI-12, 12, 1217-1222, 1990.

10. Brown, M. E. & Rogers, S.J. "*Method And Apparatus For Verification Of A Computer Userr's Identification, Based On Keystroke Characteristics*". Patent Number 5,557,686, U.S. Patent and Trademark Office, Washington, D.C., Sept, . 1996.

11. Joyce, R., and Gupta , "*Identity Authentication Based On Keystroke Latencies*" Commun. ACM , Vol .33, pp. 168 -176, Feb. 1990.

12. Polemi, D. Biometric techniques: "*Review And Evaluation Of Biometric Techniques For Identification And Authentication*" . Report prepared for the European Comission DG XIII-C.4 on the Information Society Technologies, 2000 .

13. Yanushkevich, S., Stoica, A., Srihari, S., Shmerko, V., Gavrilova, M. Simulation of Biometric Information: "*The New Generation Of Biometric Systems*" Proc. BT2004 Int'l Workshop on Biometric Technologies, pp.87-98, (2004).

14. Ma, Y., Schuckers, M., & Cukic, B. "*Guidelines For Appropriate Use Of Simulated Data For Bio Authentication - Research*". Automatic Identification Advanced Technologies, Fourth IEEE Workshop on, 251-256, 2005.

15. Yanushkevich, S. N., Stoica, A., Shmerko, V.P. and Popel, D. V. "*Biometric Inverse Problems*", Taylor & Francis / CRC Press, 2005.

16. Nicholas M. Orlans and Douglas J. Buettner and Joe Marques: "*A Survey Of Synthetic Biometrics: Capabilities And Benefits*" , Proceedings of the International Conference on Artificial Intelligence (IC-AI'04), CSREA Press, Vol. I, 499- 505, 2004

17. Yanushkevich, S.N., Stoica, A., Wang, P. S., Srihari, S.N. "*Introduction To Synthesis In Biometrics*", Chapter 1 in Image Pattern Recognition: Synthesis and Analysis in Biometrics, Vol. 67, WSPC (World Scientific Publishing Co.), pp. 5-29, 2007.

18. Questbiometrics.com, 'Biometric Definition', http://www.questbiometrics.com/biometric-definition.html, (Last Updated 2005).

19. Zuo, J. and Schmid,N.A. "*A Model Based, Anatomy Based Method For Synthesizing Iris Images*", Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506, USA, 2002.

20. GlobalSecurity.org, 'Biometrics Overview', http://www.globalsecurity.org/security/systems/biometrics-history.htm, (Last Updated 2009).

21. Matyas, V. Jr., Riha,Z. "*Toward Reliable User Authentication Through Biometrics*", IEEE Security & Privacy Magazine, Vol. 1 No. 3, pp. 45-49, 2003.

22. Pearsall,J., Trumble,B. (Eds), The Oxford English Reference Dictionary, Oxford University Press, Great Claredon Street, Oxford OX2 6DP, 2001.

23. Daugman,J., "*The Importance Of Being Random: Statistical Principles Of Iris Recognition*", Pattern Recognition, 36(2), pp. 279-291, 2003.

24. Mathworks.com, 'MATLAB Central', http://www.mathworks.com/matlabcentral/newsreader/view_thread/154150, (Last Updated 2009.)