ISBN 978-82-326-1424-0 (printed version)
ISBN 978-82-326-1425-7 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2016:42

Soumik Mondal

Continuous User Authentication and Identification

Combination of Security & Forensics

NTNU
Norwegian University of
Science and Technology
Faculty of Computer Science and Media Technology
NISlab - Norwegian Information Security laboratory

NTNU
Norwegian University of
Science and Technology

Doctoral theses at NTNU, 2016:42

NTNU

NTNU
Norwegian University of
Science and Technology

Soumik Mondal

Continuous User Authentication and Identification

Combination of Security & Forensics

Thesis for the degree of Philosophiae Doctor

Gjøvik, February 2016

Norwegian University of Science and Technology
Faculty of Computer Science and Media Technology
NISlab - Norwegian Information Security laboratory

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Computer Science and Media Technology
NISlab - Norwegian Information Security laboratory

ISBN 978-82-326-1424-0 (printed version)
ISBN 978-82-326-1425-7 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2016:42

Printed by Skipnes Kommunikasjon as

Continuous User Authentication and Identification

Combination of Security & Forensics

Soumik Mondal

Thesis submitted to
Norwegian University of Science and Technology

for the degree of
Philosphiae Doctor (PhD) in Information Security

NTNU
Norwegian University of
Science and Technology

February, 2016

Continuous User Authentication and Identification

Faculty of Computer Science and Media Technology
NTNU, Gjøvik, Norway

The saddest aspect of life right now is that science gathers knowledge faster than society gathers wisdom.

(Isaac Asimov)

Declaration of Authorship

I, Soumik Mondal, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed:

(Soumik Mondal)

Date:

Summary

In almost every aspect of human life have computing devices (such as PC, smartphone, tablet, or smart watches) become important gadgets. The communication services, aviation and financial services are very much controlled by computer systems. People entrust with vital information such as medical and criminal records, manage transactions, pay bills and private documents. However, this increasing dependency on computer systems, coupled with a growing emphasis on global accessibility in cyberspace, has unveiled new threats to computer system security. In addition, crimes and imposters in cyberspace are almost everywhere.

For most existing computer systems, once the user's identity is verified at login, the system resources are available to that user until he/she exits the system or locks the session. In fact, the system resources are available to any user during that period. This may be appropriate for low security environments, but can lead to session hijacking, in which an attacker targets an open session, *e.g.* when people leave the computer unattended for shorter or longer periods when it is unlocked, for example to get a cup of coffee, to go and talk to a colleague, or simply because they do not have the habit of locking a computer because of the inconvenience. In high risk environments or where the cost of unauthorized use of a computer is high, a continuous check of the user's identity is extremely important. Continuous authentication has built around the biometrics supplied by the user's physical or behavioural characteristics and continuously checks the identity of the user throughout a session. Continuous authentication is not an alternative security solution for initial login; it provides an added security measure alongside the initial login. In this work we describe a continuous authentication system where multiple behavioural biometric modalities are fused to increase the system performance and to avoid security holes that can be exploited by imposters to avoid detection.

This thesis does not only focus on the *Continuous Authentication (CA)*, but also on *Continuous Identification (CI)* which can be used for forensic evidence. During our research we address two issues. The first is related to CA (Is an imposter using the system?) while the second is related to CI (Can the imposter be identified once the continuous authentication system detects that an imposter uses the system?). To the best of our knowledge this is the first time that the CI issue is addressed in research. We present the achieved results for different biometric modalities and for different computing devices. We have used four different datasets for experiments of which three are publicly available; therefore the achieved results can be reproduced and verified.

We contributed a robust dynamic trust model algorithm that can be applied to any CA system irrespective of the biometric modality or computing device. Contrary to the state of the art CA approaches this algorithm is able to make decisions whether the user is genuine or imposter after each and every single action performed by the user. In most of the cases we found that genuine users are never wrongly locked-out from the system and very few actions were required to detect an imposter user. We applied a novel score boost algorithm that improves the results and the achieved results are superior when compared to state of the art results. We came up with a feature selection technique that could equally well be applied to other pattern classification problems.

We came up with an identification technique called pairwise user coupling that can reduce a multi-class classification problem into several two-class classification problem. We applied this technique for CI and achieved a high identification accuracy even for weak biometric modalities. We believe however that there are some open issues which need to be addressed before this can be used as a deployable solution.

Acknowledgments

I wish to express my sincere appreciation and gratitude to those who have contributed to this thesis and supported me in one way or the other during this amazing journey.

First of all, I am extremely grateful to my main supervisor, *Prof. Patrick Bours*, for his guidance and all the useful discussions and brainstorming sessions, especially during the difficult conceptual development stage. His deep insights helped me at various stages of my research. I would also like to give him special thanks for accepting me as a PhD student under his supervision and allowing me to grow as a research scientist.

Very special thanks to the *Norwegian Information Security laboratory (NISlab)* for giving me the opportunity to carry out my doctoral research and for their financial support. I would like to thank the members of the PhD review committee for accepting their task, even in hardship. I also want to thank you for letting my dissertation defence be an enjoyable moment, and for your brilliant comments and suggestions.

A big "Thank you!" also goes out to everybody who participated in my experiments and provide their valuable biometric data.

PhD students often talk about loneliness during the course of their study, but this is something which I never experienced in Gjøvik. A heartfelt thanks to all my friends who made the Gjøvik experience something special.

Finally, a special thanks to my family. Words cannot express how grateful I am to my mother for all of the sacrifices that you have made on my behalf. Your prayer for me was what sustained me thus far. I would also like to thank all of my friends who supported me and incanted me to strive towards my goal.

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Inception

Chapter 1

Introduction

1.1 Motivation

People use access control mechanisms, like username-password, token, or biometrics, to protect against unauthorized access by another person. This means that a user needs to give proof of his/her identity when starting or unlocking a computer or mobile device. However, in many cases, people leave the computer physically unattended for shorter or longer periods when it is unlocked, *e.g.* to get a cup of coffee, to go and talk to a colleague, or simply because they do not have the habit of locking a computer because of the inconvenience.

Access control is generally implemented as a one-time proof of identity during the initial log on procedure. The legitimacy of the user is assumed to be the same during the full session. Unfortunately, if the device is left unlocked and unattended, any person can have access to the same information as the genuine user. This type of access control is referred to as *Static Authentication (SA)* or *Static Login (SL)*. On the other hand, we have *Continuous Authentication (CA)* (also called *Active Authentication* by DARPA[1]), where the genuineness of a user is continuously monitored based on the biometric signature left on the device. When doubt arises about the genuineness of the user, the system can lock, and the user has to revert to the SA access control mechanism to continue working. Continuous authentication is not an alternative security solution for static authentication; it provides an added security measure alongside static login.

In case of the CA, the system should lock to avoid any damage done by, or information revealed to the imposter. The obvious requirements are to detect an imposter as fast as possible to limit the amount of damage, while at the same time avoiding, to the largest possible extend, the incorrect locking out of the genuine user. Furthermore should a CA mechanism, much more than a SA method, perform its tasks unnoticed to the user. This immediately rules out the use of knowledge or possession based authentication system. Knowledge based systems will disturb the user when having to type a password, while possession based systems are not effective for users that do not remove their token when leaving the system unattended. Besides, a stolen token would give an attacker the same access rights as the genuine user and would not lead to detection by the computer system. This then leaves biometrics as a potential solution for continuous authentication. In the proposed system the behaviour of the current user is compared to the normal behaviour of the genuine user and deviation from this normal behaviour will lead to a lockout. The motivation behind the use of behavioural biometrics is the unobtrusive nature of the data collection for some behavioural biometrics *e.g.* keystroke dynamics, mouse dynamics or swipe gesture *etc.* (the details about these biometrics can be found in Chapter 2).

In our research, we are not only looking at *Continuous Authentication (CA)* where the system checks if the current user is the genuine user, but also at *Continuous Identification (CI)* where the system tries to identify the current user of a system. CI can be used as forensics evidence. During our research we address two questions:

- CA: Is the current user the genuine user of the system?
- CI: If an imposter is detected by the CA system, then who is this imposter?

To the best of our knowledge is this the first time that the CI issue is raised in research. Performing CA-CI by analysing the user's behaviour profile is challenging due to the limited amount of

[1] `http://www.darpa.mil/our_work/i2o/programs/active_authentication.aspx`

information that is available and the large intra-class variations. Previous research has mainly been done in a periodic manner, where the analysis was based on a block of a fixed number of actions (also sometimes with a fixed period of time). This creates a limitation of the system, that is, if the system could detect an imposter before that fixed number of actions is completed, then the imposter is still allowed to finalize that fixed number of actions. We have tried to mitigate this limitation in our research. We decide on the genuineness of the user after each and every action in our research

Most research on continuous authentication reports performance in terms of *False Match Rate (FMR)* and *False Non-Match Rate (FNMR)* or even *Equal Error Rate (EER)*. For a CA system it is important to know **if** an imposter is detected or not, but it is even more important to know **when** the imposter is detected, *i.e.* how much activity he/she has been able to perform before detection. We will use the *Average Number of Imposter Actions (ANIA)* and *Average Number of Genuine Actions (ANGA)* as our performance indicators. Hence, our performance indicator shows how much an imposter can do before he/she is locked out and how much a genuine user can do before he/she is, wrongfully, locked out of the system. These are the equivalents of FMR and FNMR for a CA system.

1.2 Goal of this Research

This work aims to explore new techniques for continuous user authentication to provide optimal security for the computing devices. Additionally, this work tries to explore the possibility to establish the identity of the intruder or adversary for forensic evidence.

With this background, we investigate the new possibilities for continuous authentication to overcome the drawbacks of the state of the art CA systems. Also, we provide an alternative performance measure metric for CA systems. We have used behavioural biometric modalities in our research, but the proposed techniques can be applied to any biometric modality based CA system. Our proposed techniques have been validated with extensive experiments by using two datasets from different computing devices (*i.e.* PCs and mobile devices), with different biometric modalities (*i.e.* keystroke dynamics, mouse dynamics and swipe gestures). We have also explored action based biometric features for various modalities in our research.

The concept of combining security and forensics in a continuous manner is investigated for the first time. We have proposed three different identification schemes with pairwise user coupling for CI. There are a variable number of actions available for identification, as well as high intra-class variations and low inter-class variations motivate us to formulate these schemes. We would like to mention that these schemes are not modality and device dependent and also do not dependent on the continuous data. Therefore, these schemes can be applied to any identification problem. These proposed schemes are also validated with extensive experiments with different modalities for different devices. These experiments were conducted for both an open-set and a closed-set identification setting.

1.3 Contribution of this Thesis

The primary contributions made in this thesis are as follows:

- The CA problem statement is not novel in the research domain. But, according to our observation, most of the state of the art CA research was performed in a periodic manner (*i.e.* periodic authentication). The experiment was conducted for these researches by having a fixed number of actions interval or fixed number of time intervals. In this thesis, we have explored the possibility to perform CA without having this constraint. We have developed a robust trust model algorithm that can be used for a CA system, irrespective of the biometric modality. This approach can be found in Chapter 3.

- As our CA system does not work in a periodic manner, we found that system performance reporting in terms of EER, or FMR and FNMR is no longer applicable for such a system. We have come up with a novel performance measure technique for CA systems. This technique can be found in Chapter 4.

- Extensive experiments were conducted to validate our approach with different biometric modalities for different computing devices (*i.e.* PCs and mobile devices). We have chosen behavioural biometrics (*i.e.* keystroke dynamics, mouse dynamics and swipe gesture) for our proof of concept. But, we believe that this approach can be applied to any biometric modality based CA system. The achieved results can be found from Chapters 6 to 9. During our research, we have also come up with some novel features for mouse dynamics and these can be found in Chapter 5.

- We have come up with a novel feature selection technique during our research. We found that in some datasets this algorithm works very well, but some other datasets, it does not work in an optimal way. This algorithm can be found in Chapters 8 and 9.

- The concept of continuous identification has been introduced for the first time in the research community. The combination of continuous authentication and identification will provide a robust system that not only protects the device from an adversary but also aims to establish the identity of the adversary. The related chapters about this concept can be found in Part III.

- We have developed three different identification schemes for CI by using a pairwise user coupling. These approaches were followed to mitigate the problem of behavioural biometric modalities (*i.e.* low inter-class variation and high intra-class variation). But, these approaches can be applied to any classical pattern recognition and identification problem. These approaches can be found in Chapter 10.

- The concept of CI was validated by experiments with different biometric modalities for different computing devices (*i.e.* PCs and mobile devices). These results can be found in Chapters 11 and 12.

1.4 Structure of this Dissertation

This dissertation consists of five parts: the introduction and the state-of-the-art are in Part I; the research contributions are in Part II (CA) and Part III (CI); the conclusion and the future work are in Part IV; and the additional research works that is loosely related to our main research can be found in the Part V. Figure 1.1 shows the overview of the chapters in Part II and Figure 1.2 shows the overview of the Part III chapters.

After some basic overview to the state-of-the-art in Chapter 2 we continue from Chapter 3 to Chapter 12 to clarify the details of our research approach and achieved results. The CA related contributions (see Part II) include the followed analysis approach in Chapter 3 and the performance reporting metrics in Chapter 4. The description of the datasets used in this research and the extracted features from the raw data can be found in Chapter 5. The CA research results in the PC application domain are shown in Chapters 6 to 8. Within that, Chapter 6 describes the achieved results by using only mouse dynamics, Chapter 7 describes the achieved results by using only keystroke dynamics, and Chapter 8 describes the achieved results by utilizing the combination of keystroke and mouse dynamics. Chapter 9, shows the achieved CA results in the mobile devices application domain.

The CI related research (see Part III) can be found from Chapter 10 to 12. The same datasets as described in Chapter 5 are used for CI. Chapter 10 clarifies the identification methodology followed in this research, whereas the achieved results for the PC and mobile application domain can be found in Chapter 11 and Chapter 12 respectively.

1.5 List of Publications

1.5.1 Journal

1. [95] MONDAL, S., AND BOURS, P. A computational approach to the continuous authentication biometric system. *Information Sciences 304* (2015), 28 – 53.

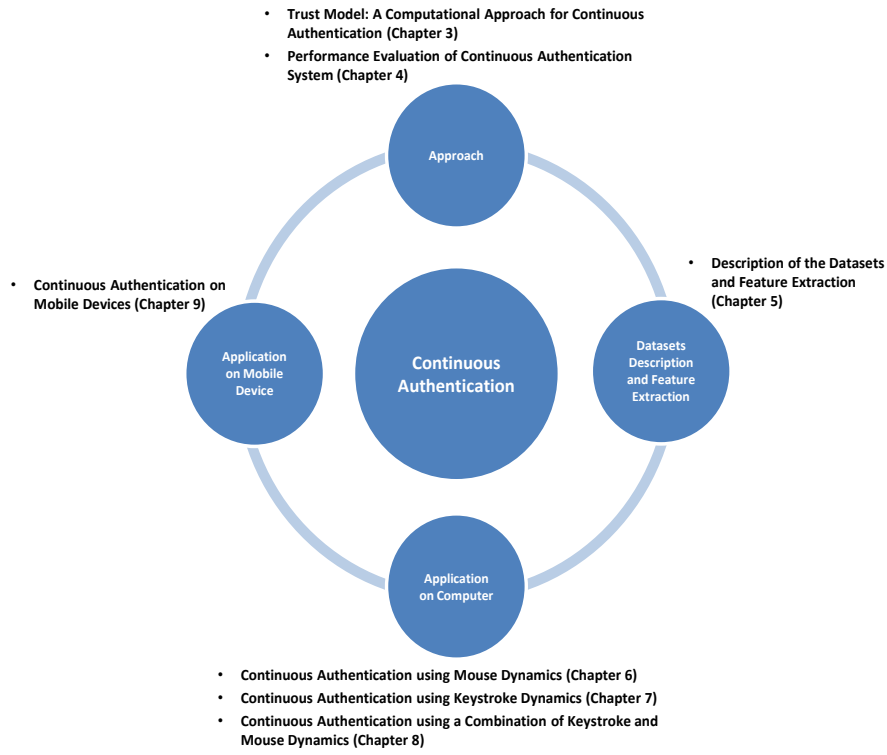


Figure 1.1: Overview of the Part II chapters: Continuous Authentication

2. [22] BOURS, P., AND MONDAL, S. Performance evaluation of continuous authentication systems. *IET Biometrics* (2015), 1–7.

3. [104] MONDAL, S., AND BOURS, P. A study on continuous authentication using a combination of keystroke and mouse biometrics. Under Review in Neurocomputing, 2016.

4. [103] MONDAL, S., AND BOURS, P. Person identification by keystroke dynamics using pairwise user coupling. Under Review in IEEE Transactions on Dependable and Secure Computing, 2016.

5. [102] MONDAL, S., AND BOURS, P. Continuous user authentication and adversary identification: Combining security & forensics. Under Review in IEEE Transactions on Information Forensics & Security, 2016.

1.5.2 Book Chapter

1. [21] BOURS, P., AND MONDAL, S. *Continuous Authentication with Keystroke Dynamics*. Science Gate Publishing, 2015, ch. Recent Advances in User Authentication Using Keystroke Dynamics Biometrics, pp. 41–58.

2. [106] MONDAL, S., BOURS, P., JOHANSEN, L., STENVI, R., AND ØVERBØ, M. *Importance of a Versatile Logging Tool for Behavioural Biometrics and Continuous Authentication*

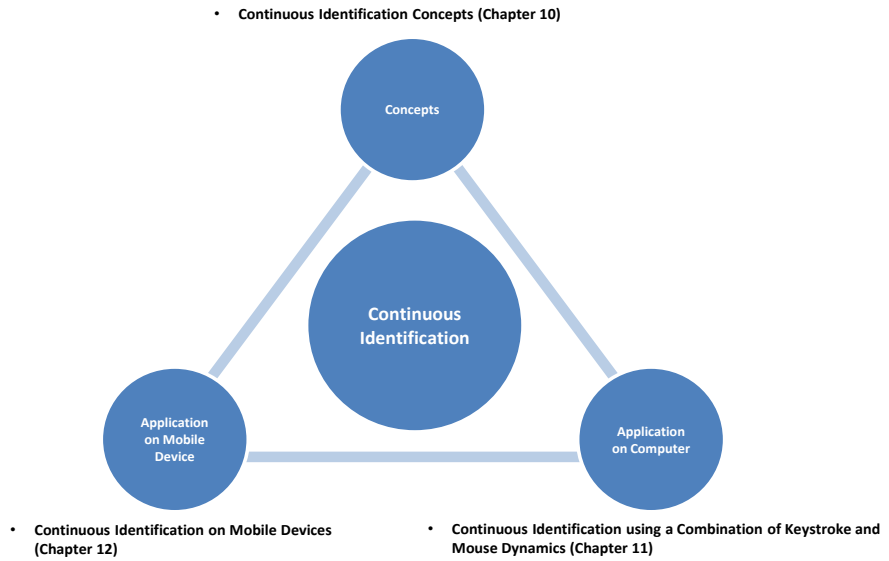


Figure 1.2: Overview of the Part III chapters: Continuous Identification

Research. IGI Global, 2015, ch. Handbook of Research on Homeland Security Threats and Countermeasures.

1.5.3 Conference

1. [92] MONDAL, S., AND BOURS, P. Continuous authentication using behavioural biometrics. In *Collaborative European Research Conference (CERC'13)* (2013), pp. 130–140.

2. [93] MONDAL, S., AND BOURS, P. Continuous authentication using mouse dynamics. In *Int. Conf. of the Biometrics Special Interest Group (BIOSIG'13)* (2013), IEEE, pp. 1–12.

3. [105] MONDAL, S., BOURS, P., AND IDRUS, S. Z. S. Complexity measurement of a password for keystroke dynamics: Preliminary study. In *6th Int. Conf. on Security of Information and Networks (SIN'13)* (2013), ACM, pp. 301–305.

4. [94] MONDAL, S., AND BOURS, P. Continuous authentication using fuzzy logic. In *7th Int. Conf. on Security of Information and Networks (SIN'14)* (2014), ACM, pp. 231–238.

5. [98] MONDAL, S., AND BOURS, P. Continuous authentication in a real world settings. In *8th Int. Conf. on Advances in Pattern Recognition (ICAPR'15)* (2015), IEEE, pp. 1–6.

6. [96] MONDAL, S., AND BOURS, P. Context independent continuous authentication using behavioural biometrics. In *IEEE Int. Conf. on Identity, Security and Behavior Analysis (ISBA'15)* (2015), IEEE, pp. 1–8.

7. [100] MONDAL, S., AND BOURS, P. Swipe gesture based continuous authentication for mobile devices. In *Int. Conf. on Biometrics (ICB'15)* (2015), IEEE, pp. 458–465.

8. [99] MONDAL, S., AND BOURS, P. Does context matter for the performance of continuous authentication biometric systems? an empirical study on mobile devices. In *Int. Conf. of the Biometrics Special Interest Group (BIOSIG'15)* (2015), IEEE, pp. 1–5.

9. [97] MONDAL, S., AND BOURS, P. Continuous authentication and identification for mobile devices: Combining security and forensics. In *7th IEEE Int. Workshop on Information Forensics and Security (WIFS'15)* (2015), IEEE, pp. 1–6.

10. [101] MONDAL, S., AND BOURS, P. Combining keystroke and mouse dynamics for continuous user authentication and identification. In *IEEE Int. Conf. on Identity, Security and Behavior Analysis (ISBA'16)* (2016), IEEE, pp. 1–8.

Chapter 2

State of the Art

In this chapter, we describe how continuous authentication has been achieved in the past using unimodal behavioural biometrics or with biological biometrics. We also explore the methods followed to fuse multiple behavioural biometric modalities to increase the performance of the continuous authentication system and to avoid security holes that can be exploited by imposters to avoid detection. We will present results achieved with these methods.

This chapter is based on the paper published in: [92] MONDAL, S., AND BOURS, P. Continuous authentication using behavioural biometrics. In *Collaborative European Research Conference (CERC'13)* (2013), pp. 130–140.

2.1 Introduction

In almost every aspect of the human life, computer systems and networks have become an important gadget. Communication services, aviation, and financial services are already controlled by computer systems. People entrust vital information to these systems, such as medical and criminal records, financial transactions, and personal emails. However, this increasing dependency on computer systems coupled with a growing emphasis on global accessibility in cyberspace, has unveiled new threats to computer system security. In addition, crimes and imposters in the cyberspace appear almost everywhere. Crimes on the computer networks may cause serious damages, including communication blocking, perusal of classified files, and commerce information destruction.

Attacks on a computer system can happen on the network level, system level or user level, or any combination of these three levels. Network-level attacks include network denial of service and probing. System-level attacks include privilege escalation, such as buffer overflow, program modification, perhaps caused by a Trojan horse or virus, and denial of service. User-level attacks include masquerade and imposter attacks. In our research, we are mainly concentrating on user level attack *i.e.* imposter attacks.

For most existing computer systems, once the user identity has been verified at login, the system resources are available to the user until the user exits the system or the session will be locked. This may be appropriate for low-security environments, but can lead to session hijacking in which an attacker targets a post-authenticated session. In high risk environments or where the cost of unauthorized use of a computer is high, continuous verification or authentication of the user is extremely important. A *Continuous Biometric Authentication System (CBAS)* was built with the biometric data supplied by a user's physical or behavioural characteristics, and it continuously checks the identity of the user throughout the session [145]. However, a single biometric modality may be inadequate for user verification either because of noise in the data sample, unavailability of a sample at a given time or universality/uniqueness issues of that particular biometric modality. To overcome this limitation, researchers have proposed the use of multiple biometric modalities and have demonstrated increased accuracy of verification [126].

2.1.1 Application areas of Continuous Authentication

Continuous Authentication can be applied in any environment where the cost of unauthorized access is very high. Some of the examples are,

- On-line banking and shopping;

- E-learning and on-line exams;
- Defence computer controls;
- Computers for airline cockpit and marine controls;
- Health care;
- Cyber-criminal profiling;
- Mobile devices (*i.e.* smart phone and tablet PC).

2.2 Background Knowledge

In the information security or the computer security domain there are two types of systems that enable the link to a person and his/her identity,

1. ***Identity verification or Authentication:*** When the user claims who he is and the system accept (or declines) his/her claim. Authentication can be divided into three according to the way of their implementation,
   a) *Static Authentication (SA):* The system will authenticate the user only one time, that is at login time;
   b) *Periodic Authentication (PA):* The system will re-verify the identity of the user after a fixed number of actions or a fixed time intervals;
   c) *Continuous Authentication (CA):* The system will re-verify the identity of the user continuously.
2. ***Identity Identification:*** When the system established the identity of a person (or fails to do it) without any prior claim.

In any secure system the Authentication or Identification can be done in three ways, *i.e.* by

1. ***Something the user knows:*** *e.g.* password, pass-phrase, PIN and the answer of any security questions *etc.*;
2. ***Something the user owns:*** *e.g.* smart card, SIM card, phone, security token, software token and navigator cookies *etc.*;
3. ***Something the user is or does:*** This is actually biometric systems (*i.e.* "Automated recognition of individuals based on their behavioural or biological characteristics[1]"). There are two types of biometric modalities,
   - *Biological Biometrics (also called Physical Biometrics): e.g.* face, DNA, fingerprint, palm-print, retina, iris, hand-vein, ear, facial thermography, oder, hand geometry and voice *etc.* [65].
     - `Pros:`
       a) Permanent;
       b) Universally unique.
     - `Cons:`
       a) Special hardware required for biometric data capturing;
       b) Generally obtrusive;
       c) Computational complexity is high.

[1] `http://www.iso.org/iso/iso_technical_committee?commid=313770`

- *Behavioural Biometrics: e.g.* gait, mouse dynamics, keystroke dynamics, signature, software interaction *etc.* [151].
  - `Pros:`
    a) Some of the modalities don't require any special hardware *e.g.* keystroke dynamics, mouse dynamics and swipe gesture;
    b) Can be unobtrusive;
    c) Computational complexity is low for many modalities *e.g.* keystroke dynamics, mouse dynamics and swipe gesture.
  - `Cons:`
    a) Not permanent;
    b) Unique within a small group of users;
    c) System performance is lower when compared to biological biometrics.

Figure 2.1 shows the block diagram of a biometric system with four basic components of enrollment, authentication and identification. These four components are:

- ***Biometric Data Capture:*** This component involves the capture of the biometric data from an individual. This component includes both hardware and software. Some biometric systems also include a *Quality Checker* component after the data capture module to check the captured data quality for further processing but, this is an optional component.

- ***Feature Extraction:*** This component extracts the distinguishable features of the captured biometric data to generate a user's profile or template. These profiles are stored in the profile database (*i.e. Profile DB*) in the enrollment phase and retrieve from the database during the authentication or identification phase.

- ***Profile DB:*** This component is used to store and manage the biometric templates or profiles generated from the individuals.

- ***Matcher:*** During recognition (*i.e.* authentication or identification) this component used to compare the presented biometric sample (*i.e.* extracted features of the presented biometric sample) with the stored profile and generate the decision. In case of authentication, the matcher tries to match the claimed individual's profile with the presented biometrics and give the decision in true/false. In case of identification, the matcher tries to compare all the profiles in the database with the presented biometrics and gives a decision as an identified user based on the closest match.

Biometric authentication systems can make two types of errors during matching, which define the performance of that system. There are some industry defined measures for describing the performance of any biometric authentication system. The terms are described as [65, 67],

- ***False Match Rate (FMR):*** The probability that the system accepts an imposter user. For example, an FMR of 3% means that 3 out of every 100 imposter users are falsely matched by the *Matcher* component as genuine users.

- ***False Non-Match Rate (FNMR):*** The rate that the system rejects the genuine users. For example, FNMR of 3% means that 3 out of every 100 genuine users are rejected by the *Matcher* component as imposter users.

- ***Equal Error Rate (EER):*** The rate at which FMR equals FNMR. From the above examples the EER is 3%.

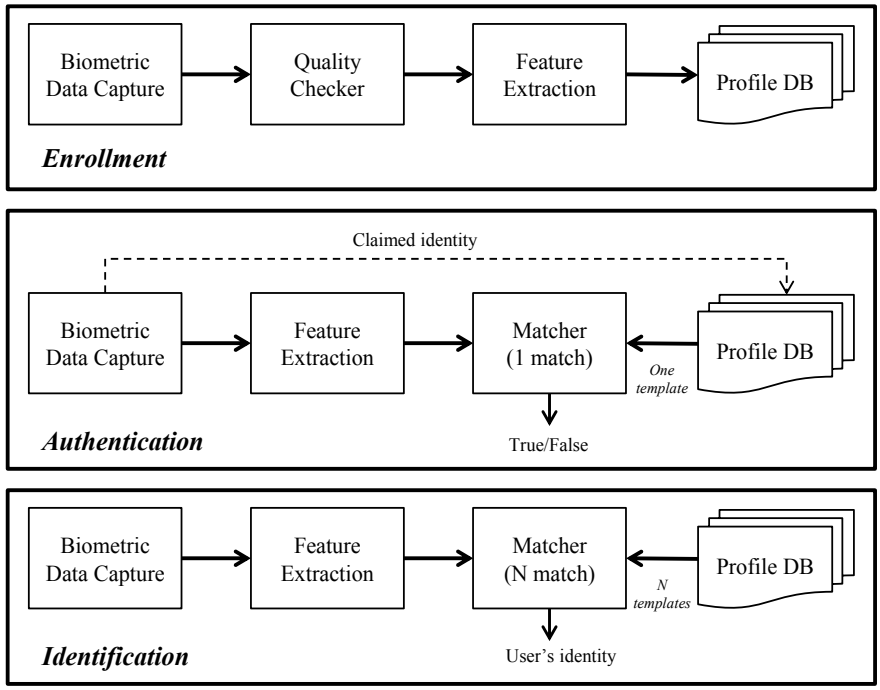


Figure 2.1: Block diagram of a biometric system [65].

2.2.1 Keystroke Dynamics

In *Keystroke Dynamics (KD)*, users are identified or authenticated based on the way they type on a keyboard. When a password is typed not only the correctness of the password itself is checked, but also if the typing rhythm when entering the password is correct. This process is sometimes called password hardening. The use of KD as a method of identification is not new. During the early days, the telegraph operators were able to identify each other by their *Morse* code typing pattern. This identification method, known as *"The Fist of the Sender"*, was used as a verification or identification method during World War II [49]. Nowadays software is available for password hardening for example the software from BioPassword[2].

A KD based authentication or identification system is low cost and easy to implement, because most of systems are software based. In such a system, the keystroke timing information has to be captured and features for authentication or identification are extracted. Sometimes some special keyboard (*i.e.* pressure sensor based keyboard [143]) or key press sound information [113] was used to capture the key pressure information or key sound information for authentication or identification.

Any KD data capture tool can record the key press time and key release time as a raw data. The following features for pattern recognition are calculated from the raw timing information [11]:

1. *Key Code:* Key code is the ASCII code that represents each key on a keyboard.

2. *Down-Up (DU) Time:* The DU time is defined as the time interval a key remains pressed. In literature this is also referred to as dwell-time or hold-time or duration.

[2] http://www.biopassword.com/

3. *Up-Down (UD) Time:* The UD time is the time between releasing one key and pressing the next key. This feature is also referred to as the keystroke latency between two keys. In literature this is also referred to as seek-time. The latency value is generally positive, but it can be negative. In case the next key was pressed already before the previous key is released, then the latency is negative. This can happen if the user types very fast, or if he uses special keys, like the shift key.

4. *Down-Down (DD) Time:* The DD time is the elapsed time between pressing one key and pressing the next key. In literature this is also referred to as latency or flight-time.

5. *Up-Up (UU) Time:* UU time is a defined as the time interval between successive key releases. In literature this is also referred to as latency or flight-time.

2.2.2 Mouse Dynamics

Mouse Dynamics (MD) has been defined as the way users are interacting with their system through the mouse. The basic assumption of MD is that every mouse user has some mouse usage patterns which are different from other users. Similar to KD, MD does not require any special hardware for data capture (sometimes researchers have used special mouse devices that have a fingerprint scanner [124] for experiments). From 2003, MD has become an interesting topic in the area of behavioural biometrics due to it's non-intrusiveness and convenience [39].

For MD based biometric authentication, we need to capture the mouse trajectory and mouse click data while users interact with their system. According to literature mouse features can be divided into two parts [51, 134]:

- ***Schematic features:*** These features characterize the constituents of mouse actions during GUI interactions such as the statistical distribution of mouse action types or mouse pointer positions. There are four different schematic features we can generate from the raw data that are generally used in the literature:

  1. *Mouse action histogram:* statistics of occurrences of various mouse action types.
  2. *Percentage of silent periods:* statistics of idle time of Mouse.
  3. Distribution of cursor positions on the screen.
  4. Distribution of movement distances per direction.

- ***Motor-skill features:*** This feature characterizes the efficiency, agility and motion habits of individual mouse actions such as the acceleration pattern or the speed of a double click. There are five different motor-skill features we can generate from the raw data that are generally used in the literature:

  1. *Elapsed time of single click:* time interval between down and up of left/right/middle button of a click.
  2. *Elapsed times of double click:* overall time and three internal intervals between downs and ups of left/right/middle button of a double click.
  3. *Average movement speed compared to the directions:* average movement speed calculated for different directions.
  4. *Average movement speed and acceleration compared to travelled distance:* average speed/accelerations calculated for different distance travelled.
  5. *Transition time of actions:* transition time between consecutive mouse actions.

2.2.3 Mobile Biometrics

Due to technological advances are we increasingly dependent on mobile devices. Such devices are widely used for banking transactions, therefore they contain highly sensitive information. The way users interact with a mobile device (*i.e.* tablet or smart phone) can be used as a biometric modality to authenticate the genuine users. User interaction with a mobile device can be achieved by tapping behaviour which is similar to KD and swipe gestures which are similar to MD. Due to the inbuilt pressure sensor in a mobile device we can also use pressure and area of interaction in both the cases. Similar to a PC, where KD can be used for password hardening [28] swipe gesture based patters can be used for static authentication on mobile devices [26, 33, 86, 132]. The applied KD based systems have used similar features as discussed in Section 2.2.1. The swipe gesture based systems have used some motor-skill related features and some trajectory related features as discussed in Section 2.2.2.

2.3 Related Work

After the initial introduction, we focus on CA systems using behavioural biometric modalities. There are two basic issues we would like to focus on:

- What is the motivation to choose behavioural biometrics?
  - Generally behavioural biometrics *i.e.* KD, MD or swipe gesture recognition do not require any additional hardware for data capture.
  - We expect that for these modalities, analysis will not be computationally complex when compared to other biological biometric modalities due to the limited amount of information.
  - We can collect user data without interrupting the normal daily work activity of the user. Also, we can collect the behavioural biometric data in a network environment in a covert manner.
- Why a combination of KD and MD?
  - As we know mouse and keyboard are the most common input devices for computers. We used a combination of KD and MD to avoid the situation where an attacker avoids detection by restricting as much as possible to one input device because the system only checks the other input device.
  - Combination of these two modalities can improve overall system performance.
  - We believe that it is very difficult to spoof more than one behavioural biometric modality simultaneously.

We will discuss the related work in our research domain. The literature survey is divided based on the used modality and devices.

2.3.1 CA using KD

KD relates to the way that a user types on his keyboard. The first article, as far as we know, referring to KD is by Umphress *et al*. [147] from 1985, but the majority of research in this area is from 2000 or later (before the article [147], there was a report published by RAND Corporation where they have shown some preliminary results on authentication by using KD [49]). The vast majority of research in KD focuses on timing information features, but a small number of works also includes other features, like pressure. For KD, most articles focus on static authentication and only a minority focusses on CA. According to our knowledge in 1995 the first research was published on CA by using KD [136].

We can divide CA research using KD into three groups based on the amount of data used and the data acquisition process followed. In the first group the researchers have used relatively small amount of data from individuals, *i.e.* 1000 keystrokes per sample or less [4, 31, 32, 44, 45, 58, 59,

60, 76, 87, 89, 107, 122, 137], in the second group researchers have used relatively large number of data from individuals, *i.e.* 6000 keystrokes per sample or less [48, 69, 82, 118, 142] and in the third group researchers have used data more than 6000 keystrokes per sample for each individual [3, 19, 37, 66, 117, 138]. We can also observe that the researches [4, 31, 32, 58, 137] were done on the same dataset where the owner of this dataset is Gunetti *et al*. [58] and the researches [45, 59] were done on the same dataset where the owner of this dataset is Filho *et al*. [45]. According to our knowledge except these two datasets *i.e.* [45, 58] all the other datasets are not publicly available. Most of the mentioned researches have used free text for the experiments except [48, 76, 118], where they have used fixed text (*i.e.* predefined text for all the participants) for their experiments. Except [48, 76, 82, 89, 122, 142], all the other experiments were conducted in an uncontrolled environment (*i.e.* no predefined PCs and no laboratory experimental setup) to represent the users' natural behaviour.

Almost every existing research on KD have used keystroke timing features for classification in some form or other as described in the Section 2.2.1. In some research, *n-graph* duration was used as a feature [4, 31, 32, 66, 44, 58, 76, 87, 138], while [31, 32, 44] have used *n-graph duration* along with DU and UD time as features and [66, 138] have used word specific *n-graph* features. In [69, 137], researchers have used all the features as described in the Section 2.2.1 on the other hand, in [3, 19, 60, 107, 117, 122] researchers have used only DU and UD time as features. In [45, 59, 89, 118] work researchers have used DD time as features, where as Monaco *et al*. [89] have used DU and UD time along with DD time in the feature vector. Furnell *et al*. [48], have used only DU time in their work on the other hand Dowland *et al*. [37] have used di-graph, tri-graph and word duration as features in their work. In [82, 142] researchers have used some extra features along with timing features. In [82], cognitive-centric features and in [142] stylomerty features were used for classification.

From the state of the art we found that the majority of the researches have used distance based classifiers for classification, but some of the researches have followed the machine learning approach. In [58, 69, 76, 87, 122], researchers have used relative distance (*i.e.* R-distance and A-distance [58]) where as in [19, 31, 32, 37, 44, 59, 66, 69, 138], researchers have used absolute distance measures (*i.e.* Euclidean distance or scaled Manhattan distance) for feature classification. Lazy learning approach *e.g. k-nearest neighbour* or *nearest neighbour* classifiers was used in [69, 89, 107, 142] and *Neural Network* as a machine learning approach was used in some of the researches [3, 48, 59]. Also, data clustering approach was used for classification in [4, 137]. Table 2.1 shows the summary of the related CA researches using KD with their applied methods and achieved performance.

2.3.2 CA using MD

There is a less amount of research that focuses on CA using MD when compared to KD [2, 41, 52, 81, 112, 131, 133, 155]. Most of these researches were conducted in an uncontrolled environment (except [133]) with uncontrolled tasks (except [52]). We can find from these studies that they have used 5 to 15 hours of data per user for the experiment, but some of these researchers did not report the amount of data used for the experiment, *i.e.* [41, 131, 155]. We also find that, except [131] all the other studies have used machine learning approach for classification. Table 2.2 shows the summary of the related CA researches using MD with their applied methods and achieved performance.

2.3.3 CA using a combination of KD and MD

Only few studies exist where researchers have used a combination of KD and MD for CA [10, 47, 63, 121, 146] in a multi-modal architecture [126]. In [10, 121], we can find that GUI interaction was also used as an additional modality. All of these studies were conducted in a controlled environment with some predefined tasks. Table 2.3 shows the summary of these researches with their applied methods and achieved performance.

Table 2.1: Summary of the related CA researches using KD.

Reference	Method	Users	Performance
[3]	Neural Network (NN)	53	EER of 2.13%
[4]	Clustering	14	Accuracy of 100%
[19]	Distance	35	182 Keystrokes
[31, 32]	Distance	12	FAR of 0.07% and FRR of 15.2%
[37]	Distance	35	FAR of 4.9% and FRR of 0%
[44]	Distance	60	EER of 1.4%
[45]	Markov Chain	15	EER of 12.7%
[48]	NN	30	FAR of 15% and FRR of 0%
[58]	'R' and 'A' Distance	40	FAR of 0.005% and FRR of 5%
[59]	Distance and NN	15	EER of 22.9%
[60]	One-Class Classification	10	FAR of 11.3% and FRR of 20.4%
[66, 138]	Distance and Naive Bayes	22	Accuracy of 70%-100%
[69]	12 different techniques	35	EER of 5.64%
[76]	'R' and 'A' Distance	10	FAR of 4.09% and FRR of 5.17%
[82]	Fisher Score	486	EER of 4.55%-13.37%
[87]	'R' and 'A' Distance	55	FAR of 2.02% and FRR of 1.84%
[89]	k-Nearest Neighbour (k-NN)	119	EER of 3.7%
[107]	Distance and k-NN	31	Accuracy of 23%
[117]	Chi-square	42	FAR of 0.8% and FRR of 0%
[118]	KS-test	35	EER of 0.08% - 0.09%
[122]	'R' and 'A' Distance	50	EER of 10%-15%
[137]	Clustering	21	FAR of 3.47% and FRR of 0%
[142]	k-NN	40	EER of 0.5%

Table 2.2: Summary of the related CA researches using MD.

Ref.	Method	Users	Performance
[2]	NN	22	EER of 2.46%
[41]	Random Forest (RF)	25	EER of 7.5%
[52]	Weibull distribution	50	EER of 0.06%
[81]	Support Vector Machine (SVM), k-NN and Decision Tree (DT)	20	Accuracy of 54%-95%
[112]	Learning Algorithm for Multivariate Data Analysis (LAMDA)	48	FAR of 0% and FRR of 0.36%
[131]	Distance	72	EER of 11.1%
[133]	NN, k-NN and SVM (one class)	28	FAR of 0.37% and FRR of 1.12%
[155]	SVM	30	FAR of 2.96% and FRR of 0.86%

Table 2.3: Summary of the related CA researches using a combination of KD and MD.

Reference	Method	Users	Performance
[10]	Bayesian network (BN), DT and SVM	31	FAR of 2.10% and FRR of 2.24%
[47]	Naive Bayes and SVM	67	FAR of 0.1% and FRR of 0.2%
[63]	NN, k-NN and PPMCC	20	Accuracy of 82.22%-96.4%
[121]	DT and SVM	61	Error Rate of 1.5%
[146]	BN	24	EER of 8.21%

2.3.4 CA on mobile devices

Nowadays CA system for mobile devices are also studied and some impressive results are shown that could be used as a motivation for continued study in this domain [17, 42, 43, 46, 56, 80, 129, 132, 135, 154]. Table 2.4 shows the state of the art research in the CA domain on mobile devices.

Table 2.4: Summary of the related CA researches on mobile devices.

Reference	Method	Users	Performance
[17]	SVM	10	EER $< 1\%$
[42]	k-NN and Dynamic Time Warping (DTW)	23	Accuracy of 90%
[43]	DT, RF and BN with Sliding window	40	FAR of 3.8% and FRR of 2.8%
[46]	k-NN and SVM	41	EER of 3%
[56]	Distance	41	EER of 22.5%
[80]	SVM	28	Accuracy of 79.74%-95.78%
[129]	NN and k-NN	40	EER of 3.3%
[132]	Distance and 8 machine learning approaches	190	EER of 13.8%-33.2%
[135]	SVM (one class)	51	FAR of 7.52% and FRR of 5.47%
[154]	Correlation Distance	30	EER of 2.62%

Table 2.5: Summary of the related CA researches with other biometric modalities.

Ref.	Method	Modalities
[9]	Trust Model with Fuzzy logic	face and fingerprint
[30]	SVM (one class)	Face on mobile devices
[40]	SVM	Screen recordings of PC interactions
[57]	Distance	Electrocardiogram (ECG)
[72]	RGB colour matching and Principal Component Analysis (PCA)	Face and skin color
[110]	Dynamic Bayesian Networks and SVM	Face and keystroke
[114]	Haar classifier and PCA	Face and soft biometrics
[127]	Clustering	Video based KD
[139]	Hidden Markov Model	Face and special mouse with fingerprint scanner
[153]	Multi-task Multivariate Low-Rank Representation (MLRR)	Face and swipe gesture on mobile devices

Some of these researches have used KD based CA (*i.e.* tapping behaviour) [43, 129] and some of them have used swipe gesture behaviour for CA [17, 42, 46, 56, 132, 135, 154]. We can also find the combination of tapping and the swipe gesture for CA [80]. Except for [42] are all the other experiments conducted in a controlled setting.

2.3.5 CA using other biometric modalities

In this section, we will discuss some researches that are not directly related to our research, but they have developed a CA system with other biometric modalities [9, 30, 40, 57, 72, 110, 114, 127, 139, 153]. Some of these researchers [9, 40, 57, 110, 114, 139, 153] are conducted in a controlled environment and others are conducted in an uncontrolled environment [30, 72, 127]. Except [57], all the other researches have used image processing techniques for analysis. In [9, 30, 72, 110, 114, 139, 153] biological biometrics were used *e.g.* face and fingerprint for CA. Also soft biometrics (*i.e.* skin color or clothes) were used along with biological biometrics for analysis [72, 114]. In [110], KD was used along with face biometrics and video based KD biometrics was used in [127]. A new biometric modality *i.e.* screen print was introduced [40]. Table 2.5 shows the summary of these researches with their applied methods.

2.4 Summary

Summary of the state of the art research:

- Most of the research is conducted in a controlled environment or with a predefined task. A controlled environment or predefined task does not represent the real world scenario for CA, due to the fact that users are concentrated towards completion of the tasks, which might influence their normal behaviour.

- Except [19], all of these researches were conducted in either periodic manner or in a traditional authentication manner with continuous data. In our work we will focus on an actual CA

system where each and every performed action by the user will be taken into consideration and according to the genuineness of the performed action(s) the system will decide whether the present user may continue to work or not.

- Most of these works represent their system performance in terms of EER of FMR and FNMR. According to our understanding this is not the proper way to report the CA system performance. We will provide an alternative CA system performance measure metric in our research (see Chapter 4).

Part II

Continuous Authentication

Chapter 3

Trust Model: A Computational Approach for Continuous Autentication

In this chapter, we describe the approach followed for CA. As we are trying to build a CA system that reacts immediately on every action or activity performed by the user. Therefore, a robust computational algorithm is required, that can handle the large variation of the user's behaviour. We developed a trust function that will be described in this chapter.

This chapter is based on the paper published in: [95] MONDAL, S., AND BOURS, P. A computational approach to the continuous authentication biometric system. *Information Sciences 304* (2015), 28 – 53.

3.1 Introduction

The basic idea is that the trust or confidence of the system in the genuineness of the current user depends on the deviations from the way this user performs various actions on the system. If a specific action is performed in accordance with how the genuine user would perform the task (*i.e.* as it is stored in the genuine user's profile/template), then the system's trust in the genuineness of this user will increase, which is called *Reward*. If there is a large deviation between the behaviour of the genuine user and the current user, then the trust of the system in that user will decrease, which is called *Penalty*. The amount of change of the trust level can be fixed or variable. A small deviation from the behaviour of the user, when compared to the template, could lead to a small decrease in trust, while a large deviation could lead to a larger decrease.

No single person will be able to always behave in exactly the same manner [15]. For the genuine user this means that he/she will also sometimes deviate more from his/her normal behaviour, which will lead to a decrease in trust. However, the majority of actions that a genuine user will perform will be close to his/her normal behaviour, *i.e.* lead to an increase of trust. Overall this would lead to a high level of trust. For an imposter however the opposite holds. In some cases he will behave as the genuine user, increasing his level of trust, but the majority of actions will lead to a decrease in trust due to the large deviation from the behaviour of the genuine user. This will then lead to a general decrease of the trust over time for an imposter user. Obviously an ideal system should perform in such a way that the trust in anyone other then the genuine user will decrease fast to a value below a predefined threshold $T_{lockout}$, then the system locks itself and will require static authentication of the user to continue working. In such an ideal system, also a genuine user would never reach a trust level that would result in a lockout, *i.e.* the genuine user would not notice the presence of the continuous authentication system in his daily activities.

In Figures 3.1 and 3.2 we elaborate the concept of *Trust Model* in more details. In Figure 3.1 we see how the trust level changes when we compare the profile of a genuine user with test data of that genuine user. In this figure, we see that the trust level, sometimes decreases due to penalties, but it never drops below the lockout threshold (*i.e.* $T_{lockout} = 90$, marked by a red line). Figure 3.2 shows that if the same user's profile is compared to test data of an imposter user, the trust will drop (in this example) 5 times below the lockout threshold within 500 user actions[1]. We set the upper limit of the system trust is 100, to prevent a situation where an imposter user benefits from the high system trust obtained by the genuine user, before he/she hijacks the system.

[1]Here, action means performed activity *e.g.* keystroke or mouse click.

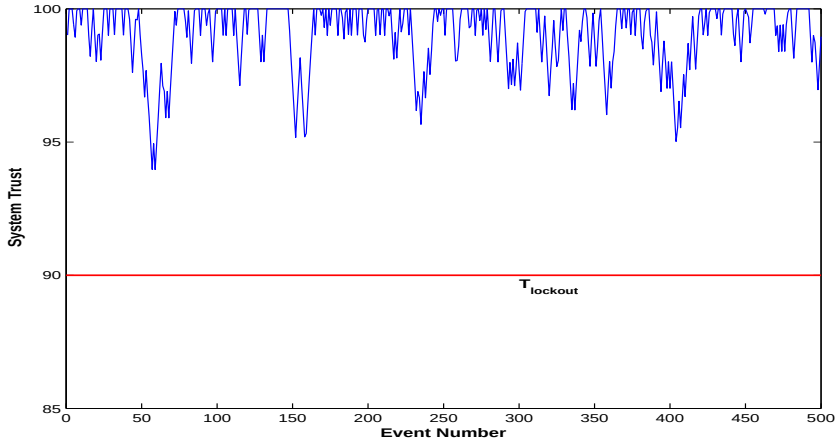


Figure 3.1: Trust value for genuine user tested with the genuine test data.

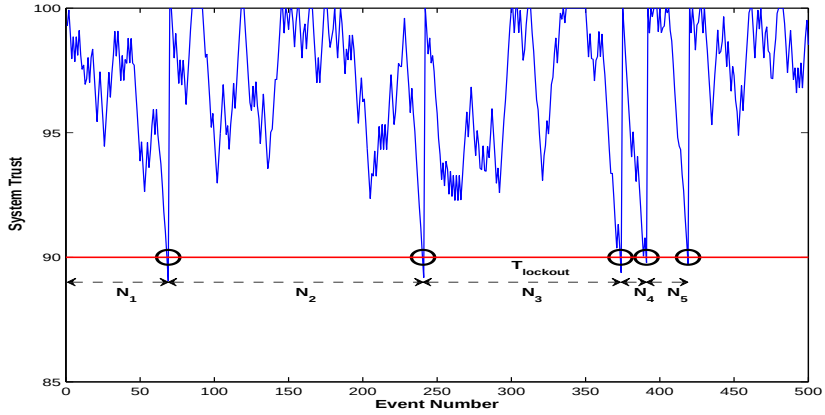


Figure 3.2: Trust value for genuine user tested with the imposter test data.

Bours [19] has described a trust model concept for continuous authentication using keystroke dynamics. He demonstrated that the trust level will increase or decrease according to the absolute distance (*i.e.* scaled Manhattan distance) between the genuine user's template and the current typing. We can also use classifier scores (*i.e.* the probability of the genuineness of that event or action) to increase or decrease the trust value. There are two major concerns we need to address related to the trust model:

- *The threshold for the Penalty or Reward:* The classifier score (*i.e.* $sc_i = P(x|H_1)$ where x_i is the feature vector of the i^{th} performed action and H_1 is the hypothesis for the genuine user) generally ranges from 0 to 1. Therefore, we have to set a threshold (Tr) where, $sc_i \geq Tr$ we will give *Reward* otherwise *Penalty*.

- *The amount of the Penalty or Reward the system will give corresponding to the classifier score:* The amount of penalty or reward can be fixed, for example if $sc_i \geq Tr$ increase the

Algorithm 3.1: Algorithm for 3-level Static Trust Model.

Data:
$sc_i \leftarrow$ Classification score for the i^{th} action
$Tr \leftarrow$ Threshold for Penalty or Reward
$Tr_P \leftarrow$ Threshold for 2^{nd} level Penalty
$Trust_{i-1} \leftarrow$ System trust after $(i-1)^{th}$ action
Result:
$Trust_i \rightarrow$ System trust on the user after i^{th} action

```
begin
    if sc_i ≥ Tr then
        Δ_T(sc_i) = f_Reward(sc_i)
    else
        if Tr_P ≤ sc_i < Tr then
            Δ_T(sc_i) = −f^1_Penalty(sc_i)
        else
            Δ_T(sc_i) = −f^2_Penalty(sc_i)
    Trust_i = min {max {Trust_{i−1} + Δ_T(sc_i) , 0} , 100}
```

trust value by 1 otherwise decrease the trust value by 1. On the other hand, the amount of penalty or reward can be variable depending upon the actual sc_i value.

Based on the above mentioned criteria different computational algorithms can be implemented in the *Trust Model*. Some of these are explained below.

3.2 Static Trust Model

The concept of a *Static Trust Model (STM)* comes where there is a discontinuous relationship between the penalty and reward value according to the classifier score. In literature we can find some implementation of this type of trust model where the single threshold value was used to decide whether the current action should get a penalty or a reward [19, 36]. We can call this type of trust model as 2-level STM. The main difference between the implementation by Bours [19] and Deutschmann *et al.* [36] is the function to calculate the penalty and reward value. In general this concept can be extended to n-level STM depending upon the number of levels used for rewards and penalties.

In Algorithm 3.1, we describe a 3-level STM implementation. This model has 1 level for a reward and 2 levels for penalties. In our research, we have used this trust model with different threshold parameters and also with different penalty and reward functions (*i.e.* $f_{Reward}(sc_i)$, $f^1_{Penalty}(sc_i)$ and $f^2_{Penalty}(sc_i)$).

In Algorithm 3.2 we describe a 4-level STM implementation. This model has 2 levels of rewards and 2 levels of penalties. We have used this trust model with different threshold parameters and also with different penalty and reward functions.

3.2.1 Limitations of STM

We have some limitations and drawbacks for STM. These are given below.

- It is very difficult to find optimal threshold parameters for different levels due to the large overlap between genuine and imposters score distributions;
- There can be a large difference in change in trust in case the classifier score is just below or just above the threshold;
- Very difficult to find what type of STM have to use on a system (*i.e.* 3-level or 4-level *etc.*);

Algorithm 3.2: Algorithm for 4-level Static Trust Model.

Data:
$sc_i \leftarrow$ Classification score for the i^{th} action
$Tr \leftarrow$ Threshold for Penalty or Reward
$Tr_R \leftarrow$ Threshold for 2^{nd} level Reward
$Tr_P \leftarrow$ Threshold for 2^{nd} level Penalty
$Trust_{i-1} \leftarrow$ System trust after $(i-1)^{th}$ action
Result:
$Trust_i \rightarrow$ System trust on the user after i^{th} action

```
begin
    if sc_i ≥ Tr then
        if sc_i ≥ Tr_R then
            Δ_T(sc_i) = f^1_Reward(sc_i)
        else
            Δ_T(sc_i) = f^2_Reward(sc_i)
    else
        if sc_i ≥ Tr_P then
            Δ_T(sc_i) = −f^1_Penalty(sc_i)
        else
            Δ_T(sc_i) = −f^2_Penalty(sc_i)
    Trust_i = min {max {Trust_{i−1} + Δ_T(sc_i) , 0} , 100}
```

- There is a discontinuous relation for penalty and reward values according to the classifier score.

3.3 Dynamic Trust Model

To overcome some of the limitations of the STM, we came up with a novel implementation of the trust model that we can call the *Dynamic Trust Model (DTM)*. Algorithm 3.3 explains the idea behind this implementation. This algorithm takes several parameters and returns the system trust of the genuineness of the user after the current action performed by the user. All the parameters for this algorithm can be different for different users. Also, we can change the parameters for different kind of actions performed by the users. For example, we can use three different sets of parameters for the same user for three different types of actions (*i.e.* keystroke, mouse move and mouse click).

Equation 3.1 shows the trust function for the DTM algorithm. Here, the change of trust (*i.e.* Δ_T) is calculated according to Equation 3.1, which is based on the classification score of the current action performed by the user as well as on 4 parameters. The parameter A represents the threshold value to decide between penalty and reward. If the classification score of the current action (*i.e.* sc_i) is exactly equal to this threshold *i.e.* $sc_i = A$ then $\Delta_T(sc_i) = 0$. If $sc_i > A$ then $\Delta_T(sc_i) > 0$, *i.e.* a reward is given and if $sc_i < A$ then $\Delta_T(sc_i) < 0$, *i.e.* the trust decreases because of a penalty. Furthermore, the parameter B is the width of the sigmoid function, while the parameters C and D are the upper limits of the reward and the penalty.

If the trust value after the i^{th} action is denoted by $Trust_i$, then the relation between the trust $Trust_{i-1}$ after $i-1$ actions and the trust $Trust_i$ after i actions when the particular i^{th} action had a classification score sc_i is given in Equation 3.2.

In Figure 3.3 we have shown the $\Delta_T(sc)$ produced by the Equation 3.1 based on the classification score (sc) of the current action for various sets of parameters. In this figure, we can see there is a continuous function between the penalty and reward and also there is a small difference in Δ_T when sc is just below or just above A. The major advantages of this implementation is that a single function can be used to calculate both penalty and reward where in case of penalty Δ_T is negative and in case of a reward it is positive.

Algorithm 3.3: Algorithm for Dynamic Trust Model.

Data:
$sc_i \leftarrow$ Resultant classification score for i^{th} action
$A \leftarrow$ Threshold for penalty or reward
$B \leftarrow$ Width of the sigmoid
$C \leftarrow$ Maximum reward
$D \leftarrow$ Maximum penalty
$Trust_{i-1} \leftarrow$ System trust after $(i-1)^{th}$ action
Result:
$Trust_i \rightarrow$ System trust on the user after i^{th} action

begin

$$\Delta_T(sc_i) = \min\{-D + (\frac{D \times (1 + \frac{1}{C})}{\frac{1}{C} + \exp(-\frac{sc_i - A}{B})}), C\} \tag{3.1}$$

$$Trust_i = \min\{\max\{Trust_{i-1} + \Delta_T(sc_i), 0\}, 100\} \tag{3.2}$$

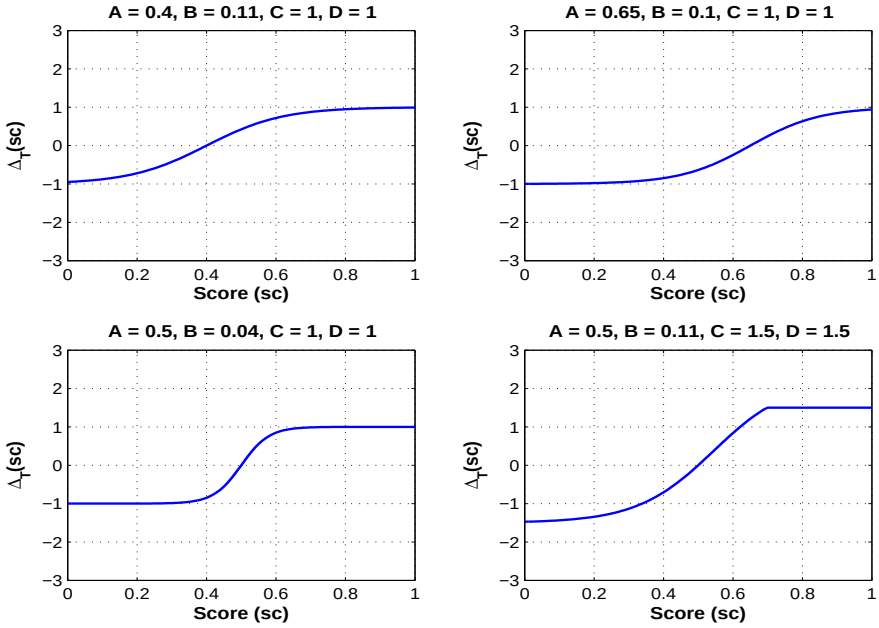


Figure 3.3: Score (sc) vs. $\Delta_T(sc)$ relation for different parameter values of Equation 3.1.

3.4 Discussion

In Figures 3.4 and 3.5 a comparison between CA and *Periodic Authentications (PA)* is shown, where the fixed number of 100 actions was chosen for PA. In Figure 3.4 we can see that both the CA and PA approach work fine when testing with test data of the genuine user, *i.e.* the CA approach did not

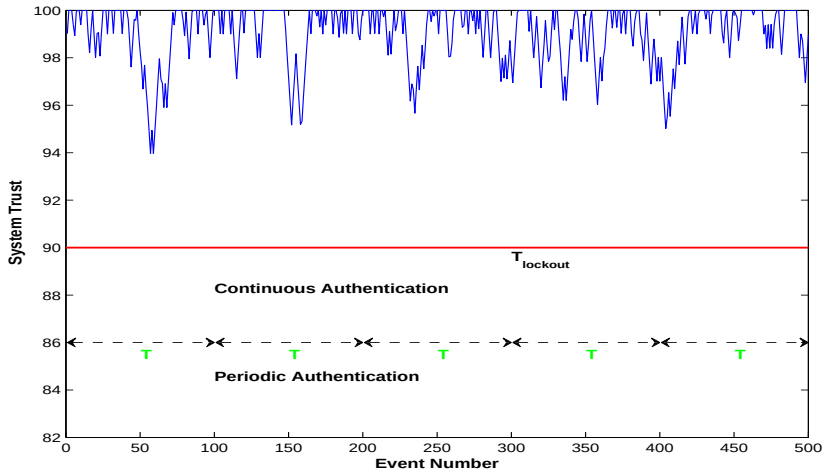


Figure 3.4: Comparison between CA and PA for genuine user tested with the genuine test data.

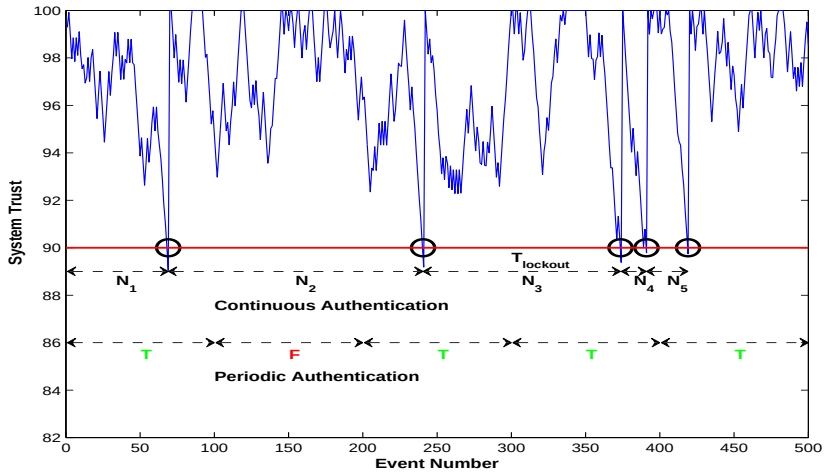


Figure 3.5: Comparison between CA and PA for genuine user tested with the imposter test data.

lock out the genuine user and also in the PA approach the genuine user was correctly verified in each block of 100 actions.

In Figure 3.5 we can see that the CA approach correctly recognizes the imposter user after N_1 actions (where $N_1 < 100$). In this case the PA system would wait until the user had completed the fixed number of 100 actions before the same decision on locking out the imposter was made. The PA system did not recognize the imposter user in the second set of 100 actions (marked as 'F' in red) and the CA approach also takes more than 100 actions to recognize the imposter user. The CA approach is again able to detect the imposter user before the end of the third set of 100 actions. From this small example we can observe the advantages of our CA approach over the PA approach.

In our analysis, we have restricted the upper limit of the trust level to 100. The reason for this is that if the genuine user would be working for a longer time, in principle the trust could rise to a high level, say for example to 1000. This means that if an imposter would take over the computer then, he would not be detected by the system until the trust level drops below the lockout threshold. In case of no upper limit, this means that the imposter could first profit from the trust that was build up by the genuine user, allowing him/her more time for malicious activities.

We have performed our analysis in the action domain rather than in time domain. Based on our understanding the number of actions performed by the users within a specific time period depends highly upon the individual's specific behaviour. Therefore, we have decided to report the performance of the continuous authentication system in the action domain.

3.5 Summary

In this chapter, we described the concept of the trust model for CA system. We provide different types of trust model implementations (*i.e.* STM and DTM). These algorithms are used in our CA system for different behavioural biometric modalities. We have also shown the advantages to use the trust model over state of the art person re-authentication approaches.

Chapter 4

Performance Evaluation of Continuous Authentication Systems

In this chapter, we describe how performance results for continuous authentication should be measured and reported. Most research on continuous authentication is in fact evaluating blocks of data, and performance is then reported in False Match and False Non-Match Rates. Here we will describe the Average Number of Imposter or Genuine Actions as the performance indicators, and will describe a more detailed performance reporting method.

This chapter is based on the paper published in: [22] BOURS, P., AND MONDAL, S. Performance evaluation of continuous authentication systems. *IET Biometrics* (2015), 1–7.

4.1 Introduction

Generally we use biometrics for getting access to a system. The identity of a user is checked, based on biological or behavioural biometrics, and if the correct features are presented to the system then the user will get access. Access is granted only at the start of a session and is valid for the full session. This means that it is possible for an imposter to hijack a session and take control of a system after the genuine user has been granted access. This type of authentication is referred to as *Static Authentication (SA)* in this thesis [19].

A different type of authentication, that overcomes the problem described above, is so called Continuous Authentication. Such a system will continuously check the genuineness of the user and, if a change of user is detected, then this system will lock itself or send out message to a security administrator. A *Continuous Authentication (CA)* system will still need some kind of static authentication to grant users initial access to the system or to prevent access after an imposter user has been locked out. DARPA has an active authentication program where they want to eliminate SA completely, but this is not the assumption for our CA system.

A good CA system should allow the user to work as normal, *i.e.* he or she should not be disturbed in his/her daily work because the identity is checked continuously. This means that for example a system where the user must present his or her fingerprint on a scanner, or a system where he or she needs to speak specific words, is not a good system, because these interfere with the normal daily routine. In most cases, when a CA system is applied to a computer, mouse dynamics [41, 112, 133, 155] and/or keystroke dynamics [3, 19, 37, 87, 90, 117] are used, although it is very well possible to include face recognition in case a webcam is available. For simplicity sake we will, in this thesis, often assume that the CA system is based on behavioural biometrics, to help explain the ideas more concretely. The ideas can however be easily applied to other biometric modalities.

A true CA system will determine the genuineness of the user after each and every separate actions. For example for Keystroke Dynamics this means that each and every keystroke will be considered by the CA system and a decision to lockout a user or not can be made after each keystroke. Alternatively, and in literature often mistaken for Continuous Authentication, are systems that use a set of actions of the user to determine the genuineness, *e.g.* in [41] the correctness of the user is based on a set of $N = 100$ consecutive mouse actions. Such a system does not perform continuous authentication because it only makes a decision about the genuineness of the user after each N actions. This implies that an imposter can always do N actions before the system checks his/her identity. In our research we will refer to such systems as *Periodic Authentication (PA)* systems.

The attack scenario we assume in our research is that a genuine user leaves his computer unattended and unlocked for a short period of time, *e.g.* to get a cup of coffee or to talk to a colleague. During this period an imposter could get access to the same resources as the genuine user and potentially perform harmful actions if no PA or CA system is active. In the proposed CA system an imposter can only do exactly as many actions as the system needs to determine that he/she is not the genuine user, while in a PA system the attacker can always perform at least a fixed number of actions.

In an SA system, the performance of the matching algorithm is reported in FMR and FNMR. Sometimes the system performance is reported even more condensed (using the EER) or more extensive (displaying the performance using a *Detection Error Tradeoff (DET)* Curve). For an SA system it is important to know the probability that the matching system makes a wrong decision, *i.e.* the probability that a genuine user has not been granted access (*i.e.* FNMR) or that an imposter user has been granted access (*i.e.* FMR) [149]. For a PA system the FMR and FNMR might also be usable performance indicators, but for a true CA system, they have no more meaning. It is of course important to know if an imposter user gets detected by the system and gets locked out, but it is far more important to know how much activity he/she can do before the system exposes him/her as an imposter. So, when comparing two CA systems that both detect all imposter users, the system that detects the imposter fastest (*i.e.* with less number of actions) is the better one. In this case, fastest does not refer to the actual clock time before an imposter is detected, because the level of activity of various users might be different. We do here refer to the number of actions that a user can do before being exposed as imposter. For example, in case of keystroke dynamics, the number of actions refers to the number of keystrokes, *i.e.* the amount of text that can be typed before the imposter is locked out.

We would like to mention that there are other biometric problems where traditional error reporting metric does not represent the proper system evaluation. DeCann *et al*. [34, 35] have introduced the *False De-Duplication (FDD)* and *False Non-Duplication (FND)* metrics for the biometric de-duplication problem and the *False Dynamic Match (FDM)* and *False Dynamic Non-Match (FDNM)* metrics for the biometric anonymous identification problem. Other studies also presented the supremacy of the bootstrap subset technique based DET curve calculation over the traditional *point-wise* estimation of a confidence interval [18, 119].

4.2 Continuous Authentication System

In this section we first describe in more detail a CA system, using the example of keystroke dynamics, and we will focus on the way that such a system works and how the performance can be evaluated. This will be described in Section 4.2.1 and then in Section 4.2.2 we describe 3 evaluation scenarios that we used in our analysis.

4.2.1 System Description

As mentioned in Section 4.1 a CA system will consider each and every action of the user to determine the genuineness of that user. It is however not the case that each and every action by itself will lead to a lockout of the current user, but only in combination with previous actions. In biometrics, and specifically in behavioural biometrics we do know that genuine users make errors [15]. The typing behaviour of a user is never so stable that all his/her typing rhythm is exactly correct. When considering *Keystroke Dynamics (KD)*, the features that are used are duration of a key (*i.e.* the time a specific key is held down) and the latency between two consecutive keys (*i.e.* the time between releasing one key and pressing down the next key). Due to normal variation in behaviour, as well as influences of external factors, like mood, tiredness, or distraction, will duration and latency vary. Furthermore we can notice that a specific user will be more stable in his/her typing behaviour of some keys and less stable for other keys. In general we also know that some users are more stable in their overall typing manner than others. Generally in KD the template of a user will contain mean (μ) and standard deviation (σ) for the durations and latencies of the various keys and key pairs. A

small σ value indicates stable behaviour of that user for that particular duration or latency, while a larger value implies the user is not so stable.

Often a normal distribution of the timing value (duration or latency) is assumed. Although this is not completely true, it does define a workable assumption. Under that assumption we know that 68%/95%/99.7% of all the timing values fall within 1/2/3 times σ difference from the mean μ. By default this implies that never 100% of all the timing values (duration or latency) fall within the range $[\mu - k \cdot \sigma, \mu + k \cdot \sigma]$ for small values of k. This implies that the genuine user will not, on each and every action, act as he/she should. But it is clear that on most of his actions, his behaviour will be considered normal and only on a minority of actions, his/her behaviour will deviate from normal.

When considering the behaviour of an imposter user however, it might happen that on a number of actions his behaviour seems to be the same as the normal behaviour of the genuine user. However, due to the fact that two people do not behave in exactly the same manner, we will see that on many actions the behaviour of the imposter deviates from the normal behaviour of the genuine user.

The above noticeable difference between genuine and imposter behaviour will be used in a CA system to differentiate between them and to lock out the imposter user and to leave the genuine user to do his daily business. The way this is done is by using the concept of "trust in the genuineness" of the current user [19]. If the system's trust in the genuineness of the user is above a specific value (threshold) then the user can continue his activity, while if the trust becomes too low, then the user is locked out and has to use the SA system again to get access to the system.

After the user has logged on using an SA system, the trust level is set to 100. Every single action of the current user will lead to an increase or decrease of this trust. More specific, if an action is performed in the normal manner (*i.e.* as described in the template of the genuine user), then the user gets a "reward" and the trust increases. Any action that is performed in a manner that deviates from the normal behaviour will lead to a decrease in trust, *i.e.* a "penalty". Given the above reasoning that the genuine user will perform most actions in a "normal" manner, his/her trust will in most cases increase and in some cases decrease, but generally will stay at a high level. An imposter user, whose behaviour will in most cases deviate from the normal behaviour of the genuine user, will have a negative trend in the trust value and will, after a certain number of actions, be locked out of the system due to the too low value of the trust. The goal is to minimize the number of actions that can be done by an imposter.

The way that the trust changes, *i.e.* the amount of penalty and reward, is a system setting that will determine how well the system performs. Penalty and reward can be fixed values, but can also depend on the actual correctness or deviation from the normal behaviour [20]. For example, a system could be implemented in such a way that if the current timing duration value of a letter deviates more than 2 standard deviations from the expected mean value, then the user would get a penalty, *i.e.* a decrease in trust. This change of trust Δ_T could be a fixed value (*i.e.* $\Delta_T = -1$) or could depend on the actual current time duration t and mean μ and standard deviation σ as given in the template (*e.g.* $\Delta_T = 2 - \frac{|t-\mu|}{\sigma}$).

The penalty and reward function plays the same role in CA as the distance metric or matching function plays in an SA system. Changing these functions will change the performance of the system. Unlike an SA system, where the distance value or matching score immediately determines the decision made by the system (either accept the user or reject him/her), there are 2 levels of decisions in a CA system. The first level is described above and is the change of the trust value based on the current action of the user. The second level will be the decision to either let the user continue working or lock him/her out. That decision is made based on the trust value after the latest action. A trust value that drops below a specific threshold will lock out a user, while otherwise the user can continue to work. Note that a user can never be locked out based on an action performed according to the normal behaviour of the genuine user, because such an action would in fact lead to an increase in trust and hence to a trust value that must be above the lock out threshold (otherwise the user would have already been locked out on the previous action). Note also that correct behaviour will lead to an increase of trust, but this trust value can never exceed 100. If a CA system would allow for the trust value to increase unlimited, then an imposter, when hijacking a current session, could profit from this build up trust of the genuine user.

	Tr_1	Te_1
IMP_1	2	
	3	
	...	
	$M - 1$	
	M	

Figure 4.1: Data separation for *VP-1*.

4.2.2 Evaluation Scenarios

When evaluating the performance of a system we can use various scenarios. We describe three general scenarios, but variations on these are possible. These scenarios will be named ”internal”, ”external”, and ”mixed” for reasons that will become clear shortly. All these scenarios are related to the use of *Machine Learning (ML)* tools. We assume that for each user of the CA system, we train a binary classifier that has the classes ”genuine user” and ”imposter user”. Each classifier is trained with a combination of genuine and imposter data in equal amounts, to avoid bias. If the number of data samples used from the genuine user equals n and data of k imposter users is used, then each of these imposter users will supply approximately n/k data samples for the training. Testing of the classifiers is never done with data that has already been used for training. The amount of training data of the genuine user was 50% of the total amount of data of that user, with a maximum of 20,000 actions[1].

In case of the ”internal” scenario, we assume that the system is used inside an organization where data of all participants is available. One may assume that, because data of all imposters is used during the training that this will influence the performance of the system in a positive manner. For that reason also the ”external” and the ”mixed” scenarios are designed. For the ”external” scenario we assume that the system might be attacked only by people of whom no data is available for training the classifier. The ”mixed” scenario is a combination of both the ”internal” and the ”external” scenario. Similar as for the ”external” scenario the classifier is trained with the data of the genuine user and the data of a subset of the imposter users.

4.2.2.1 Verification Process 1 (VP-1)

This verification process implies that data of all participants can be used to train binary classifiers. If we assume M participants within the organization, then each binary classifier is trained with the data of one genuine user and $M - 1$ imposter users. The data to test the performance of the system is the data of each participant that has not yet been used for training. Figure 4.1 explains this separation process for the first user, where $|Tr_1| \approx |IMP_1|$ to avoid classifier biassing. This verification process can be considered as the ”internal” scenario.

4.2.2.2 Verification Process 2 (VP-2)

In this case however, we do assume that M_1 imposter users whose data is used during the training are part of an organization that will employ the CA system. The remaining $M_2 = M - 1 - M_1$ imposter users are however considered as external to the organization or added to the organization after the training phase and no data of these users is assumed to be available for training purposes. Alternatively one could assume a large organization where there are too many participants to train the classifier of a genuine user with the data of all other participants. In this scenario will the testing

[1]We have used this maximum limit of 20,000 actions primarily because of two reasons. First is to keep a significant amount of data for the testing of all imposters and second is to reduce the classifier’s training time. We found that due to this maximum limit the classifier’s training accuracy also improved.

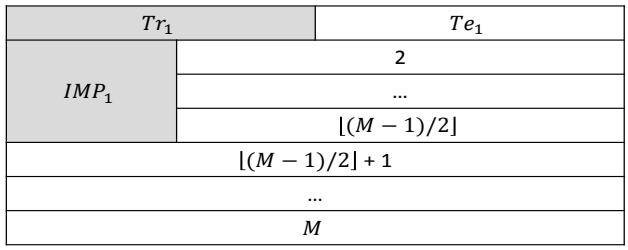


Figure 4.2: Data separation for *VP-2*.

data come from all participants, *i.e.* all imposter users are included in the testing. However, testing is never done with the same data that has been used for training already. Figure 4.2 explains this separation process for the first user, where again $|Tr_1| \approx |IMP_1|$ to avoid classifier biassing and $M_1 = \lfloor \frac{M-1}{2} \rfloor$. This verification process can be considered as the "mixed" scenario.

4.2.2.3 Verification Process 3 (VP-3)

In this verification process a set of standard imposter data should be supplied by the organization selling the CA system and that each user should train a classifier based on the supplied imposter training data and his/her own training data. This scenario can be tested by splitting the set of imposters in 2 sets of respectively M_1 and $M_2 = M - 1 - M_1$ users. Then, using the data of the M_1 imposters, plus the data of the genuine user for training (again assuring that the number of genuine and imposter data samples for training are approximately the same), the binary classifier can be trained. In this case however the testing will be done using the data of the genuine user that has not been used for training, and the data of the remaining M_2 imposter users. This procedure can be repeated multiple times for different splits of the set of imposter users.

Figure 4.3 explains this separation process for the first user, where again $|Tr_1| \approx \left|IMP_1^1\right| \approx \left|IMP_1^2\right|$ to avoid the classifier biassing, and $M_1 = \lfloor \frac{M-1}{2} \rfloor$. First, we trained the classifiers with the training data of the genuine user and the training data of the first set of $\lfloor \frac{M-1}{2} \rfloor$ imposter users, exactly as we have done in *VP-2* (see Figure 4.3(a)). Next we tested this system with the testing data of the genuine user and all of the data of the second set of imposter users. This process is then repeated with the second set of training data where the imposter users swapping roles (see Figure 4.3(b)). In this verification process imposter users are not known to the system during testing. This verification process can be considered as the "external" scenario.

4.3 Performance Indicators

As mentioned earlier FMR and FNMR are not the most suitable performance indicators for a CA system. It is more relevant to know *when* an imposter is detected then *if* he/she is detected. To be more precise, it is important to know how many actions can be performed by the imposter before he/she is exposed as an imposter. Similarly, for the genuine user, is it important that he/she does not get disturbed in his/her daily business, or at least as little as possible. Again here it is important to know how many actions a genuine user can perform before he/she is incorrectly locked out from the system. The conclusion is that we need to measure the performance in terms of *Average Number of Imposter Actions (ANIA)* and *Average Number of Genuine Actions (ANGA)*, where ANIA should be as low as possible, while ANGA should be high [93].

In the analysis of the CA system, the data of a user is split into 2 parts. The first part is used to create the template, *i.e.* either to train the classifier in case of use of ML tools or to build a statistical model. The second part of the data will be used for testing. The test data of a user is used action by action and each action will determine a change in trust. When starting the trust will be set to

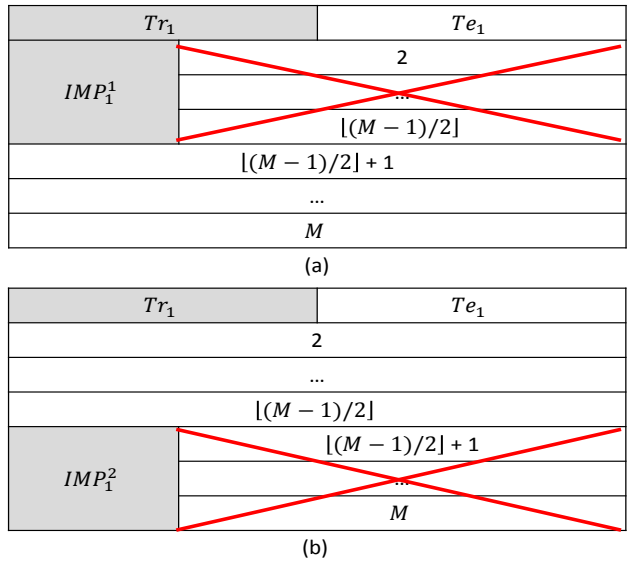


Figure 4.3: Data separation for *VP-3*.

100, indicating full trust in the current user. Any trust increase, *i.e.* any reward, will never lead to a trust above 100 because this might be misused by an imposter. If the trust drops below the lockout threshold, then we assume a user is locked out, but for the purpose of testing we will continue with the rest of his/her test data as if he/she just logged in again, *i.e.* the trust will be reset to 100. In Figure 4.4 we see an example of how the trust value changes when imposter test data is tested. We see that this imposter user is locked out 5 times within 415 actions. After every lock out the trust level is reset to 100 and unused test data is used for further testing.

In general if imposter user i, when tested against the template of genuine user g is locked out k times, after respectively $N_1, N_2, \ldots, N_k$ actions, then we define

$$ANIA_g^i = \frac{1}{k} \cdot \sum_{j=1}^{k} N_k$$

as the average number of imposter actions that imposter user i can perform against the template of genuine user g. From this we can derive that the average number of imposter actions against genuine user g equals

$$ANIA_g = \frac{1}{M-1} \cdot \sum_{i=1}^{M-1} ANIA_g^i,$$

where the summation runs over all imposter users, *i.e.* all users except the genuine user. Overall we find that the average number of imposter actions against any genuine user in this system equals

$$ANIA = \frac{1}{M} \cdot \sum_{g=1}^{M} ANIA_g,$$

where we average over all genuine users.

In the exact same manner we define $ANGA_g$ as the average number of genuine actions user g can perform when his/her test data is tested against his/her own template. The average over all users of the $ANGA_g$ values is again defined as the system's ANGA.

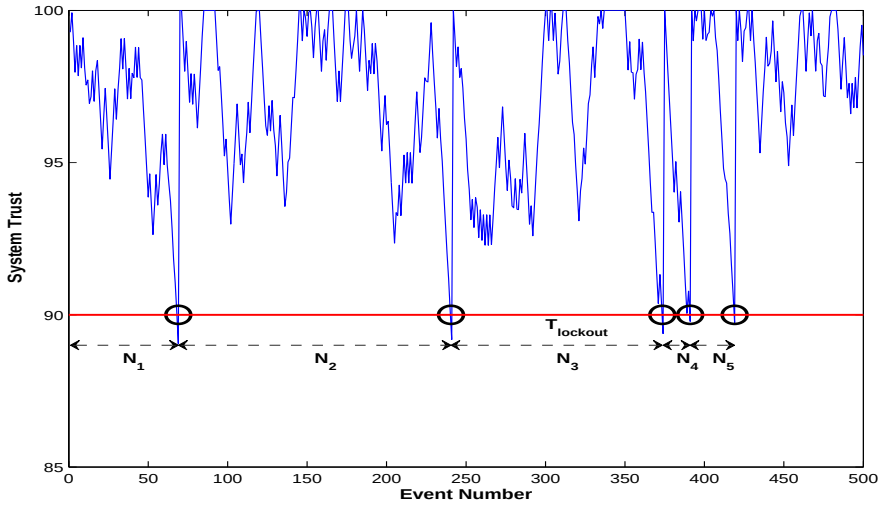


Figure 4.4: Trust level when testing of imposter 7 against the template of genuine user 1 data.

If we apply the above technique to the example in Figure 4.4 we find that $ANIA_1^7 = \frac{415}{5} = 83$, but in the example of Figure 4.5 we find that $ANIA_1^{22} = \frac{62}{3} = 21$. One could argue that the result is biased towards a lower ANIA value. Therefore, we can say that theoretically the above mentioned way of calculation of ANIA is possible when we have an infinite amount of data. But in practice due to the limited amount of data we have followed a slightly different approach where the total number of actions is taken into consideration *i.e.* $ANIA_1^7 = \frac{500}{5} = 100$ and $ANIA_1^{22} = \frac{500}{3} = 167$. In the latter approach the ANIA will always be higher that the previous approach.

4.3.1 Comparing CA and PA performance

The trust model described above works in two stages. In the first stage we see that each and every action leads to a change in the trust level. In the second stage this trust level is used to determine if the user can continue working or if the system is locked. Hence it is possible that each particular action (in combination with the past actions) can lead to a lock out of the current user because that particular action made the trust value drop below the lock out threshold. In a PA system a block of N consecutive actions is used to determine the genuineness of the user and the performance is reported in terms of FMR and FNMR. In this section we will show that we can turn the FMR and FNMR values of a PA system into ANIA and ANGA values, if the block size N is known.

In a PA system that acts on blocks of N actions, any user, genuine or imposter, can do N actions before his/her identity is checked. With a specific probability p the user is recognized as being genuine and allowed to perform the next N actions. After that, again on those last N actions a check is performed and with the same probability p the user is again allowed to continue working. Under the assumption that blocks are processed independently from each other, we derive that the expected number of actions E that this user can perform equals

$$E = N + p \cdot N + p^2 \cdot N + \cdots = \sum_{i=0}^{\infty} p^i \cdot N = \frac{N}{1-p}.$$

From this formula we can easily derive that in a PA system, where the system works on blocks of N actions and the FMR equals p, then the

$$ANIA = \frac{N}{1-p} = \frac{N}{1-FMR}.$$

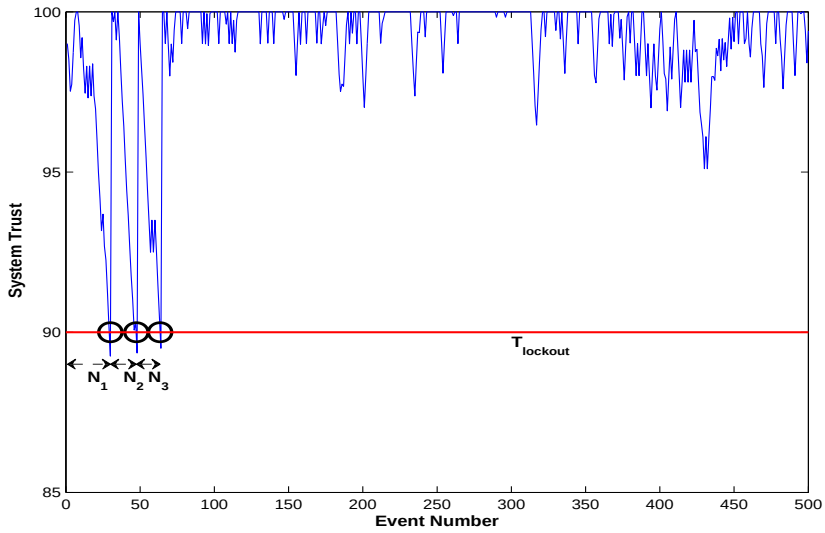


Figure 4.5: Example of a change of trust level when testing imposter data.

On the other hand, if in the same system the FNMR equals q, then this means that a genuine user is not matched against his/her own template with probability q, meaning that he is matched with probability $p = 1 - q$, so again p is the probability that the user can continue in this PA system. Now we find that

$$ANGA = \frac{N}{1-p} = \frac{N}{q} = \frac{N}{FNMR}.$$

As an example, in [112] the authors used blocks of $N = 2000$ mouse actions and they obtained an FMR of 0% and a FNMR of 0.36%. Using the above formulas we find that ANIA equals 2000 (even though FMR=0%) and ANGA is well over 550000. In [41] the authors used different block sizes. For $N = 1$ they obtained an EER of 26.25%, *i.e.* an ANGA of 3.81 and an ANIA of 1.36, while for $N = 100$ the EER dropped to 7.5%, *i.e.* ANGA=1333.3 and ANIA=108.1.

4.4 Performance Reporting

For reporting the performance of a CA system ANIA and ANGA will give a good first impression. Such a performance reporting is similar to reporting FMR and FNMR or reporting EER in an SA system: it gives an impression, but more information can also be provided. For an SA system a more complete presentation of the performance can be done with the DET curve. Similarly we can have a more extensive report for a CA system. When looking at a particular genuine user, the best that can happen is that (1) this user is not locked out of the system given his/her test data; and (2) all imposter users are locked out from the system. The lockout of an imposter user should of course happen as quickly as possible, *i.e.* after as few actions as possible.

However, there might be situations that are not as ideal as described above. It might happen that the genuine user is locked out by the system, or that imposter users are not recognized as such and are not locked out. We therefore have 4 different categories that need to be considered:

+/+ : This is the optimal situation where all imposter users are locked out and genuine users are not;

Table 4.1: Example of extended performance reporting for a CA system.

Category	# Users	ANGA	ANIA	# Imp. ND
+ / +	41		99	
+ / -	4		807	7
- / +	3	4630	164	
- / -	1	90936	512	1
Summary	49	45929	329	8 (0.3%)

+/- : In this case genuine users are never locked out of the system but some imposters are not detected;

-/+ : In this category all imposters are detected and locked out, but some genuine users are also at some point locked out;

-/- : The worst situation is when both imposter users are not detected and genuine users are locked out by the system.

The more extensive results are presented in a matrix, where the rows represent the 4 categories. Then the first column represents the number of genuine participants for each of the 4 categories (*i.e.* this column sums up to the number of participants M). The second column represents the value of ANGA, *i.e.* the average number of actions of the genuine user for the genuine users in that particular category. In case the genuine users are not locked out (so in the '+ / +' and '+ / -' categories), no number is represented here, but it is clear to be higher than the number of actions in the test data. The third column represents the value of ANIA, *i.e.* the average number of actions of the locked out imposter users for the genuine user of the particular category (the imposters that are not locked out from the system are not being part of this calculation), while the last column shows the number of imposter users that are not detected. This last number should be seen in relation to the number of genuine participants (*i.e.* the number in the first column). The final row in the matrix sums up the results for each of the columns.

In Table 4.1 an example is presented of the performance of a CA system. We see that in total 49 users participated in the analysis (numbers in the first column sum up to 49). Most of those are never locked out as genuine user (45 out of 49), but 4 genuine users are locked out of the system, *i.e.* 8.2% of the genuine users is wrongfully locked out. Even though the terminology is not entirely correct, we could artificially define the FNMR in this case as 8.2%. One can see that actually the user in the '- / -' category is locked out (on average) after over 90000 actions, so this means in practise less than once a day. The column ANGA is left blank for the first two categories, because the genuine user did not get locked out, so we cannot calculate ANGA for these users. The system ANGA can be calculated from the average number of actions that are used for testing. In this case the average number of actions for testing per user was $m = 47682$. Note that the genuine user in the '- / -' category can do 90936 actions before he/she gets locked out by the system, so even more than the average number of actions available for testing. This is because the number of actions for testing per user varied greatly from a minimum of 2908 to a maximum of 288450. For the users that do not get locked out by the system we will use the average number of actions for testing in the calculation of the system ANGA and ANIA. In case of the results of Table 4.1 we can calculate ANGA as:

$$\frac{[(41+4)\times m] + [3\times 4630] + [1\times 90936]}{49} = 45929$$

In the column ANIA the numbers vary significantly per category. The number 807 for example in the +/- category is based on the imposter users that are locked out against the 4 genuine users. Given that per genuine user there are 48 imposter users, we find that in this category we considered $4 \times 48 = 192$ imposter data sets in total. Of these, 7 are not detected (see the last column), *i.e.* not locked out. The number 807 is the ANIA averaged over the other $192 - 7 = 185$ imposter test sets.

The last column in the table represents the number of imposters sets not detected in that category. In total we see that 8 imposter sets have not lead to a lock out of that imposter user. So, similar to the artificial definition of FNMR, we could artificially define FMR as 8 out of $48 \times 49 = 2352$ or 0.34%. Note that the first and the third entry in the last column are left blank because for these two categories all imposter users are detected. The system ANIA can now be calculated as:

$$\frac{[41\times48\times99]+[(4\times48-7)\times807+7\times m]+[3\times48\times164]+[(1\times48-1)\times512+1\times m]}{48\times49} = 329$$

4.5 Summary

In this chapter, we have described ANIA and ANGA as the reporting metric for a CA system comparable to FMR and FNMR. Similar to the extension to a DET curve, we have shown that we can also report more details on the performance of a CA system. Due to the fact that our CA system reacts on every single action performed by the user, we come up with this novel performance metric for a CA system. We also show how the FMR and FNMR performance values of a PA system can be converted to ANIA and ANGA for inter-system performance comparison.

Chapter 5

Description of the Datasets and Feature Extraction

In this chapter, we will describe the datasets used in this research. In total four datasets ware used in this research where three of them are publicly available datasets and one is collected by ourself.

5.1 Dataset-1: Mouse Dynamics Dataset

To the best of our knowledge there is no publicly available dataset which contains the biometric data for continuous mouse dynamics with a significant number of users in order to provide the statistical significance of the analysis, except the dataset from the research conducted by Nakkabi *et al*. [112]. This dataset was built based on free, continuous computer mouse usages. The dataset contains the mouse dynamics data collected from 49 volunteers, but in the analysis Nakkabi *et al*. [112] have used only the data of 48 users. The volunteers were asked to use their computer and mouse in a normal, everyday fashion, without any restrictions on the tasks they had to perform. There is a huge variation in the number of samples per user (minimum 3736, maximum 333789, average 60701). The data collection software stored the following four raw information for each mouse action from a volunteer:

1. Type of action, *i.e.* mouse move, silence period, point-click, or drag-drop.
2. Travelled distance (d) in the form of the number of pixels.
3. Elapsed time (t) in seconds with a 0.25 second *Sampling Interval* or granularity.
4. Direction of movement represented by a value between 1 and 8. See Figure 5.1 for which direction corresponds to which value.

5.1.1 Limitations of Dataset-1

We encountered the following limitations for this dataset:

- This dataset shows only mouse move distance not the trajectory of the mouse move. The trajectory of the mouse move could be useful to derive some more features which might improve the system performance.
- There is no separation between left mouse button click and right button click.
- The time granularity of the capture software is 0.25 second which is very coarse and we can miss some important behavioural information from the dataset.
- There is no information about mouse click *i.e.* mouse button press and release time.
- Due to limited information about the number of sessions and separation of the sessions for each users we are unable to separate the data based on the sessions[1]. This information could be useful to separate the dataset for analysis in a training set, test set, and parameter adjustment set.

[1]Here, session means different day of work or a long pause during the work.

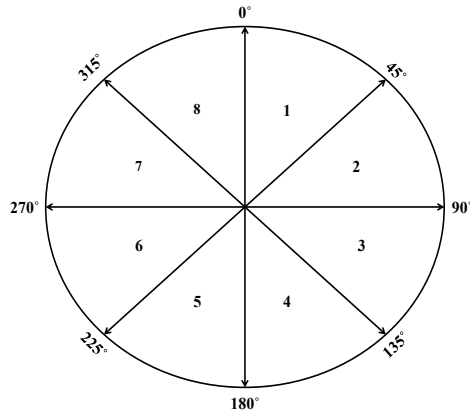


Figure 5.1: Direction of the mouse movements [2].

5.1.2 Feature Extraction

In [2, 112], as well as in many other works on CA, statistical features are extracted from the raw data. In our scheme we want to verify the identity of the user from every single mouse action. Therefore, we cannot use statistical features derived from the raw data, but we are looking for single event based features. We have extracted the following five features from the raw data:

1. *Type of action:* We have explicitly removed the *Silence* actions from the raw data because we want to focus on the behaviour of the user. We therefore, only used the other actions that were recorded *i.e.* MM, PC and DD.
2. *Direction:* Taken directly from the raw data.
3. *Speed:* This equals, $s = \frac{d}{t}$.
4. *Reciprocal of the Acceleration:* This equal, $r = \frac{t}{s}$.
5. *Travelled distance in bins:* We did not use the travelled distance in its raw form, but decided to use a limited number of bins for the travelled distance range. The first 20 bins contained a 50 pixel range each, for example if the travelled distance was between 1 and 50 pixels then we assigned bin 1, if the travelled distance was within 51-100 pixels we have assigned bin 2 and so on. After that the bins grew in range, in particular we used 38 bins according to the following schedule:
   - From 1 to 1000 pixels: Bin size is 50 pixels, so 20 bins in total;
   - From 1001 to 2000 pixels: Bin size is 100 pixels, so 10 bins in total;
   - From 2001 to 3000 pixels: Bin size is 200 pixels, so 5 bins in total;
   - From 3001 to 4000 pixels: Bin size is 500 pixels, so 2 bins in total;
   - More than 4001 pixels: Treated as a separate bin.

We first tried to use the *Acceleration* (*i.e.* $r = \frac{s}{t}$) as a feature of the mouse action, but we got much better results when using the reciprocal of the acceleration. Figure 5.2 shows the *Empirical Cumulative Distribution* plot for *Acceleration* and *Reciprocal of the Acceleration* feature for two users. From this pattern we can immediately understand that the *Reciprocal of the Acceleration* is highly separable in compression with the *Acceleration* feature. The mentioned formulas for the

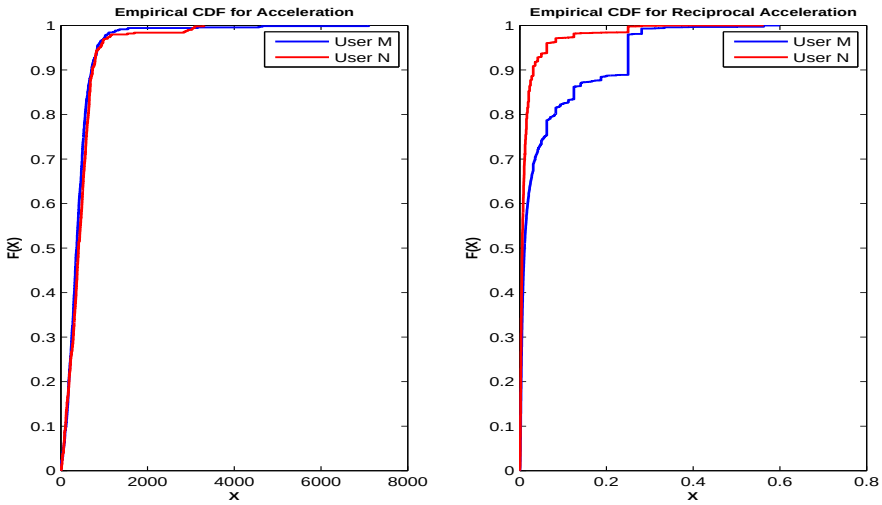


Figure 5.2: Empirical Cumulative Distribution for *Acceleration* and *Reciprocal of the Acceleration* features for two users.

calculation of speed and acceleration are not the exact formula to calculate the speed and acceleration of mouse actions [41]. As we discussed in the section 5.1.1, there are some limitations of this dataset. Due to inaccessibility of the small distance change with time within a mouse move trajectory, we are unable to estimate the precise speed and acceleration of the mouse actions. Based on our calculation of speed and acceleration, we have found that the acceleration has a very high correlation with the speed of the action. Therefore, we got worse results when using acceleration.

5.2 Dataset-2: Combination of Mouse and Keystroke Dynamics Dataset

We created a Windows operating system based logging tool, which captures the Keystroke and Mouse interaction data continuously. Log data is stored locally in a CSV file or can be transmitted to a server over a secure channel. Privacy of the users and confidentiality of the sensitive data is taken into account throughout the development of the tool. Our software is mainly designed for behavioural biometrics research, but the data we can capture could also be useful for proactive forensics and intrusion detection. The complete description of this data collection tool can be found in Appendix A. Here we will only give a brief description of the keystroke and mouse events.

5.2.1 Keystroke Events

Table 5.1 shows the data format for keystroke events. Sequence (*i.e.* Seq.) represents the sequential occurrence of the events. The *Evt. Type* is always 'K' (*i.e.* keystroke related events). Keystroke events have only two types of *actions*, key press (*i.e.* 'D') and key release ('U'). The *Value* field states which key was pressed or released in UTF-8-encoding. The *time-stamp* was recorded in milliseconds when the event occurred with a 16ms sampling interval. The *relation* attribute contains a corresponding sequence number of a previous event. *Flag* is an Integer indicating which alternate/system key was active. *Additional fields* indicates the occurrence of the pressed key for key release events.

Table 5.1: Data structure for keystroke events.

Seq.	Evt. Type	Action	Value	Time	Relation	Flag	Additional fields
n	'K'	'D' 'U'	String	ms	evt. ID	Int	n/a Count

Table 5.2: Data structure for mouse events.

Seq.	Evt. Type	Action	Value	Time	Relation	Flag	Additional fields
n	'M'	'M'	x_y	ms	evt. ID	n/a	n/a
		'U'	x_y			Int	Rectangle
		'D'	x_y				Rectangle
		'W'	Delta			n/a	n/a

Table 5.3: Data structure for mouse events.

5.2.2 Mouse Events

The data format for mouse events is shown in Table 5.3. Sequence (*i.e.* Seq.) represents the sequential occurrence of the events. The *Evt. Type* is always 'M' (*i.e.* mouse related events). Mouse events can have four types of *actions*, mouse move ('M'), mouse wheel use ('W'), mouse button press ('D') and mouse button release ('U'). The *Value* field contains the x_y mouse pointer coordinates concatenated by an '_' underscore character. In case of mouse wheel use, *Value* is the corresponding delta value indicating how much the wheel was scrolled; positive values are upward scrolls, negative are downward scrolls. The *time-stamp* was recorded in milliseconds when the event occurred, with a 16ms sampling interval. The *relation* attribute contains a corresponding sequence number of a previous event. *Flag* is an Integer indicating which mouse button was pressed/released. *Additional fields* indicates the active rectangle area for mouse button press and release events.

5.2.3 Sample log

The sample data collected with our logging software recorded in Table 5.4 shows a user typing *NISlab* in the *notepad* application. Events in the log can be interpreted as follows: From seq. 5 to 54 indicate a mouse move action, where the relation field in seq. 5 is 0, which indicate the starting of a mouse move action. Seq. 55 (mouse button down) and 59 (mouse button up) indicate a mouse single click action, where the relation field in seq. 59 is 55, which indicate the mouse button up event occurred in seq. 59 is related to the mouse down event that occurred in seq. 55. In seq. 58 the software focus changes to *notepad.exe*, which indicates opening of the notepad software. From seq. 60 to 63 *Left Shift* was pressed make the letter 'N' as a capital letter. The uses of shift key also indicate in the flag field (seq. 60 to 62) which shows the number 4. The use of backspace to remove 'i' was indicated in seq. 66 and 67. From seq. 68 to 79 we can see that the user continues typing to complete ISlab to complete NISlab. We can gather some more information related to the user's computer settings like this user has two screens (seq. 3 and 4) with the resolution of those screens and also we can understand that these events occurred on the screen 1 by the seq. 57.

5.2.4 Data Collection

Despite a high degree of privacy concern, we still got 53 volunteers to participate in our experiment. The volunteers installed our data logging software and collected data continuously for 5 to 7 days. All the participants of this data collection process are university students and staff members and they are regular computer users. We followed the data protection and privacy law according to the guidelines provided by *The Norwegian Data Protection Authority*[2]. Our data collection software for

[2] https://www.datatilsynet.no/English/

Table 5.4: Sample of data captured with our logging software.

Seq.	Evt. Type	Action	Value	Time	Relation	Flag	Additional fields					
1	start	18208296										
2	H	KEY	1033	7	18208296							
3	H	SCR_Info	0	0	1600	900	1	18208296				
4	H	SCR_Info	1600	0	2880	1024	2	18208296				
5	M	M	607_312	18208343	0							
6	M	M	607_316	18208389	5							
			...									
53	M	M	405_307	18210308	52							
54	M	M	405_303	18210324	53							
55	M	D	405_296	18210495	0	1	-1	-1	-1	-1		
56	S	OCS	belt_main.exe	18210511	55	1	Pause	-	552	303	629	326
57	H	SCR	1	18208296								
58	S	FC	notepad.exe	18210542	55	4	$\|empty\|$	15	305	239	1219	765
59	M	U	405_295	18210620	55	1	305	239	1219	765		
60	K	D	$\|Lshift\|$	18213288	58	4						
61	K	D	N	18213600	58	4						
62	K	U	N	18213709	61	4	1					
63	K	U	$\|Lshift\|$	18213896	60	0	1					
64	K	D	i	18213927	58	0						
65	K	U	i	18213990	64	0	1					
66	K	D	$\|backspace\|$	18215893	58	0						
67	K	U	$\|backspace\|$	18215971	66	0	1					
68	K	D	$\|Lshift\|$	18216704	58	4						
69	K	D	I	18217328	58	4						
70	K	U	I	18217406	69	4	1					
71	K	D	S	18217765	58	4						
72	K	U	S	18217921	71	4	1					
73	K	U	$\|Lshift\|$	18218561	68	0	9					
74	K	D	l	18220105	58	0						
75	K	U	l	18220167	74	0	1					
76	K	D	a	18220323	58	0						
77	K	U	a	18220464	76	0	1					
78	K	D	b	18223088	58	0						
79	K	U	b	18223150	78	0	1					
80	M	M	407_311	18224429	0							
81	M	M	407_319	18224445	80							
			...									
180	M	M	676_320	18226614	179							
181	M	M	676_316	18226629	180							
182	M	D	676_315	18226863	122	1	552	303	629	326		
183	S	OCS	belt_main.exe	18226863	182	1	Pause	-	552	303	629	326
184	S	FC	belt_main.exe	18226879	182	0	Stop	1003	637	303	705	326
185	M	U	676_315	18226957	182	1	637	303	705	326		
186	stop	18226957										

the experiment is fully compatible with these guidelines.

From the various previous studies we learned that collecting experimental data in a controlled setting, with a specific task on a specific computer, has major disadvantages. In this case the user will be focused more on completing the task, and the captured behaviour will not represent their normal behaviour [37, 112]. For that reason it is not possible to easily extend the results from experiments in a controlled setting to predicted results in an uncontrolled or real world setting. To address this issue our data is collected under the following conditions:

1. No specific task or mandatory use of specific applications was imposed on the user;

2. The data collection was done in a completely uncontrolled environment to represent the users' natural computer usage behaviour;

3. All of our participants installed this software on their own system to remove the hardware changes effect on the natural behavioural pattern.

Table 5.5: Data comparison with previous research.

Modality	Ref.	# User	Time period/user	Environment	Task	Applications
Keystroke	[3]	53	209 hr	Uncontrolled	Uncontrolled	Uncontrolled
	[13]	44	5 times	Controlled	Fixed	Predefined
	[37]	35	3 months	Uncontrolled	Uncontrolled	Predefined
	[87]	55	12 months	Uncontrolled	Uncontrolled	Predefined
	[90]	30	4 session	Controlled	Fixed	Predefined
	[117]	26	several days	Uncontrolled	Uncontrolled	Uncontrolled
Mouse	[41]	25	Not given	Uncontrolled	Uncontrolled	Uncontrolled
	[112]	48	14 hr	Uncontrolled	Uncontrolled	Uncontrolled
	[133]	28	15 hr (30 sessions)	Controlled	Uncontrolled	Uncontrolled
	[155]	30	5 hr	Controlled	Uncontrolled	Uncontrolled
Keystroke & Mouse	[1]	10	5 days	Controlled	Fixed	Predefined
	[10]	31	1 day	Controlled	Fixed	Predefined
	[63]	20	Not given	Controlled	Fixed	Predefined
	[146]	24	8 weeks	Uncontrolled	Uncontrolled	Web Browser
	Our	53	5 - 7 days	Uncontrolled	Uncontrolled	Uncontrolled

Table 5.5 shows the quantitative and qualitative comparison between our collected data and the data collected from previous research on continuous authentication.

Due to the completely uncontrolled data collection process that was carried out in this research, there are large differences in the amount of data collected from the various participants. We avoided providing any task or any specific instruction given to the participants, because that might lead to the participants being more focused on completing that task or follow those instructions, rather than exhibiting their own natural computer usage behaviour. We collected on an average of 7×10^5 events from each participants. Within that on average 12.4% ($\pm$ 7.7%) are keystroke events and 83.3% ($\pm$ 8.2%) are mouse related events for each of the participants. The remaining 4.3% events are software and hardware related events that are ignored in our analysis.

5.2.5 Keystroke Dynamics Features

In our research, we convert the keystroke events into two different actions.

1. *Single Key Action*, where the feature is the *key hold time* of a given key. From Table 5.4 we can see that seq. 61 and 62 are the *Single Key Action* related to entering the key 'N'. Therefore, the feature for the key press 'N' will be time difference between key down event and the key up event.

2. *Key Digraph Action*, where the features are the *Total Duration*, the time between first key press and second key press (*Down-Down Time*), the time between first key release and second key press (*Up-Down Time*) and the time between first key release and second key release (*Up-Up Time*) of a particular key digraph [8].

Figure 5.3 is a graphical representation of the keystroke dynamics feature extraction process. In Table 5.4, seq. 69 $\sim$ 72 are the two consecutive keys 'I' and 'S'. Note that the time difference between key 'I' release and key 'S' press is 359ms. In our analysis, we applied a constraint for *Key Digraph Action* that the latency between two consecutive keys should be below 2000ms. For example in Table 5.4 we see that the latency between 'a' and 'b' in events 77 and 78 equals 2624ms, hence this latency is above the threshold of 2000ms and hence disregarded. The reason why long latencies are disregarded is because they do not represent the normal behaviour of the user. For example a user can pause to think, drink coffee, or read something and then the time elapsed between releasing the last key and pressing a new key is not representative for his normal typing behaviour.

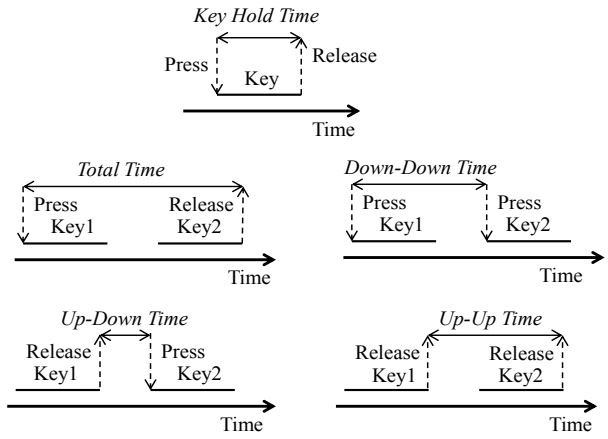


Figure 5.3: Keystroke Dynamics Features

5.2.6 Mouse Dynamics Features

In our research, we have converted the mouse events into four different actions.

1. *Mouse Single Click Action*, where the feature is similar to a *Single Key Action*. In Table 5.4, we can see that seq. 55 and 59 form a *Mouse Single Click Action*. Therefore, the feature for this *Mouse Single Click Action* will be a time difference between the mouse down event (seq. 55) and the mouse up event (seq. 59).

2. *Mouse Double Click Action*, where the features are the same as those of a *Key Digraph Action*. Two consecutive mouse clicks are considered to be a double click when the *Up-Down* time is below a threshold of 1000ms.

3. *Mouse Move Action* can be formed by the sequence of mouse move events. In Table 5.4, we can see that seq. 5 to 54 form a sequence of a *Mouse Move Action* and the features for this action are calculated according to Table 5.6.

4. *Mouse Drag-Drop Action* is very similar to the *Mouse Move Action*. For this action first there has to be a mouse click down event followed by mouse move sequences and then mouse click up event. Similar to mouse move features we have calculated *Mouse Drag-Drop Action* according to Table 5.6 and also added to the *Mouse Single Click Action* feature.

Our data collection software follows an efficient compression technique where it only records relevant mouse move actions in mouse move sequence. This means that we can reconstruct the mouse curve with negligible error. Based on the information provided by our data capture software we can extract a variety of trajectory related features for mouse move and mouse drag-drop actions [41, 83, 130]. Table 5.6 shows the mouse move and drag-drop features used in this research.

Figure 5.4 shows the *Empirical Cumulative Distribution* for a subset of mouse trajectory related features. From this figure, we can clearly understand that the users are separable from each other. These features are used for the first time in the continuous authentication research (marked in bold in Table 5.6).

5.3 Swipe Gesture based Biometric Datasets for Mobile Devices

In our research, we have used two publicly available swipe gesture based mobile biometric datasets for experiment [7, 46]. The detailed description of these datasets is given below.

Table 5.6: Mouse trajectory features for Mouse Move and Drag-Drop. Here P_i is the x-y coordinate of the mouse move sequence where $i = 0, 1, 2, \ldots, n$.

Features	Formula
Direction bin	Divided into 8 bins ($45°$)
Actual distance	$\sqrt{(x_0 - x_n)^2 + (y_0 - y_n)^2}$
Actual distance bin	Divided into 20 bins
Curve length	$\sum_{i=0}^{n-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$
Curve length bin	Divided into 20 bins
Length ratio	$\frac{Curve\ length}{Actual\ distance}$
Actual speed	$\frac{Actual\ distance}{\Delta t}$
Curve speed	$\frac{1}{n} \sum_{i=0}^{n-1} \frac{\sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}}{t_{i+1} - t_i}$
Curve acceleration	$\frac{curve\ speed}{\Delta t}$
Mean movement offset	$\frac{1}{n} \sum_{i=1}^{n} \det \begin{vmatrix} P_n - P_0 \\ P_i - P_0 \end{vmatrix} / norm(P_n - P_0)$
Mean movement error	$\frac{1}{n} \sum_{i=1}^{n} \left\lvert \det \begin{vmatrix} P_n - P_0 \\ P_i - P_0 \end{vmatrix} / norm(P_n - P_0) \right\rvert$
Mean movement variability	$\sqrt{\sum_{i=1}^{n-1} \frac{(y_i - movement\ offset)^2}{n-2}}$
Mean curvature	$\frac{1}{n} \sum_{i=0}^{n} \frac{\angle P_i(x_i, y_i) P(0,0) P_i(x_i, 0)}{\sqrt{x_i^2 + y_i^2}}$
Mean curvature change rate	$\frac{1}{n} \sum_{i=0}^{n} \frac{\angle P_i(x_i, y_i) P(0,0) P_i(x_i, 0)}{\sqrt{(x_n - x_i)^2 + (y_n - y_i)^2}}$
Mean curvature velocity	$\frac{mean\ curvature}{\Delta t}$
Mean curvature velocity change rate	$\frac{mean\ curvature}{\Delta t^2}$
Mean angular velocity	$\frac{1}{n} \sum_{i=0}^{n-2} \frac{\angle P_i P_{i+1} P_{i+2}}{t_{i+2} - t_i}$

5.3.1 Dataset-3

During this data collection process a client-server application was deployed to 8 different Android mobile devices (screen resolutions ranging from 320x480 to 1080x1205) and touch gesture data was collected from 71 volunteers (56 male and 15 female with ages from 19 to 47) [7]. To the best of our knowledge this dataset contains the largest number of users compared to other publicly available datasets. The data was collected in four different sessions with two different types of tasks. One type of task was reading an article and answering some questions about it and the other was surfing an image gallery.

This dataset is divided into two sets (*i.e.* `Set-1` and `Set-2`) were the first set consisted of all 71 users with vertical and horizontal touch gestures. The second set consisted of a subset of 51 users (42 male and 9 female) who have more than 100 horizontal gestures. The second set only contains

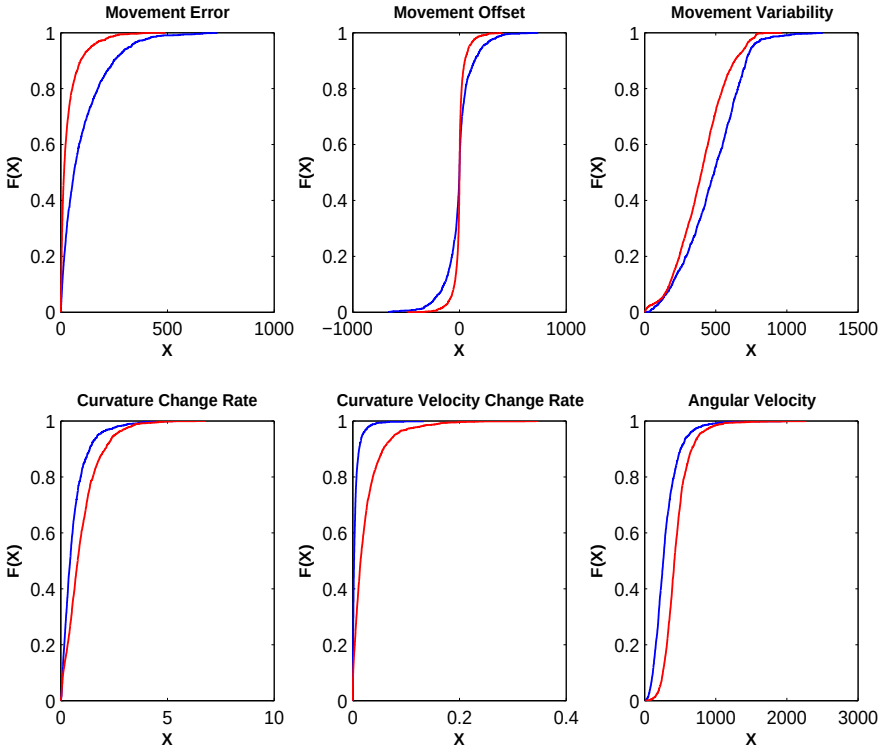


Figure 5.4: Empirical Cumulative Distribution for mouse trajectory related features. Here, blue color represents the genuine user and red color represents the imposter users.

horizontal gesture data which were not present in first subset.

Every swipe action S is encoded by a sequence of vectors $s_k = (x_k, y_k, t_k, o_k^{pl}, p_k, a_k)$, $k \in \{1, 2, \ldots M\}$, where M is the total number of sequences, x_k, y_k are the coordinates of the swipe position, t_k is the time-stamp, o_k^{pl} is the orientation of the phone (*i.e.* portrait or landscape), p_k is the finger pressure on the screen, and a_k is the area covered by the finger during swipe. In the analysis, we divided the sequence of consecutive tiny movement data into actions (*i.e.* strokes). From the encoded raw data, 15 features were calculated for each swipe action s_k. The details of this feature extraction have been described by Antal *et al*. [7]. These extracted features are:

1. *Action Duration:* The total time taken to complete the action (*i.e.* in milliseconds);
2. *Begin X:* X-Coordinate of the action starting point;
3. *Begin Y:* Y-Coordinate of the action starting point;
4. *End X:* X-Coordinate of the action end point;
5. *End Y:* Y-Coordinate of the action end point;
6. *Distance End-To-End:* Euclidean distance between action starting point and end point;
7. *Movement Variability:* The average Euclidean distance between points belonging to the action trajectory and the straight line between action starting point and end point;

8. *Orientation:* Orientation of the action (*i.e.* horizontal or vertical);
9. *Direction:* Slope between action starting point and end point;
10. *Maximum Deviation from Action:* The maximum Euclidean distance between points belonging to the action trajectory and the straight line between action starting point and end point;
11. *Mean Direction:* The average slope of the points belonging to the action trajectory;
12. *Length of the Action:* The total length of the action;
13. *Mean Velocity:* The mean velocity of the action;
14. *Mid Action Pressure:* The pressure calculated at the midpoint of the action;
15. *Mid Action Area:* The area covered by finger at the midpoint of the action.

5.3.2 Dataset-4

In this dataset, a custom application was deployed during the data collection process to 5 different Android mobile devices and the touch gestures from 41 volunteers with 5 to 7 sessions per participant were collected [46]. The data were collected for 7 different tasks, *i.e.* 4 different Wikipedia Reading articles and 3 different Image Comparison Games. Every swipe action S is encoded by a sequence of vectors $s_k = (x_k, y_k, t_k, o_k^{pl}, p_k, a_k, c_k)$, $k \in \{1, 2, \dots M\}$, where the encoding is the same as for *Dataset-3* (see Section 5.3.1), except that c_k is added. Here c_k represents the task, *i.e.* it is a value between 1 and 7. We can compute 31 different features for each stroke or action, that was discussed in more details by Frank *et al*. [46]. These extracted features are:

1. *Inter Stroke Time:* Time between two consecutive strokes;
2. *Stroke Duration:* The total time taken to complete the action or stroke;
3. *Start X:* X-Coordinate of the stroke starting point;
4. *Start Y:* Y-Coordinate of the stroke starting point;
5. *Stop X:* X-Coordinate of the stroke end point;
6. *Stop Y:* Y-Coordinate of the stroke end point;
7. *Direct End-To-End Distance:* Euclidean distance between action starting point and end point;
8. *Mean Resultant Length:* This represents how directed the stroke is;
9. *Up/Down/Left/Right Flag:* Orientation of the stroke (*i.e.* horizontal, vertical, up or down);
10. *Direction of End-To-End Line:* Slope between action starting point and end point;
11. *20% Pairwise Velocity:* 20% percentile of the stroke velocity;
12. *50% Pairwise Velocity:* 50% percentile of the stroke velocity;
13. *80% Pairwise Velocity:* 80% percentile of the stroke velocity;
14. *20% Pairwise Acceleration:* 20% percentile of the stroke acceleration;
15. *50% Pairwise Acceleration:* 50% percentile of the stroke acceleration;
16. *80% Pairwise Acceleration:* 80% percentile of the stroke acceleration;
17. *Median Velocity at Last 3 Points:* This represents the velocity before stop the stroke;

18. *Largest Deviation from End-To-End Line:* The maximum Euclidean distance between points belonging to the action trajectory and the straight line between action starting point and end point;
19. *20% Deviation from End-To-End Line:* 20% percentile of the stroke deviation;
20. *50% Deviation from End-To-End Line:* 50% percentile of the stroke deviation;
21. *80% Deviation from End-To-End Line:* 80% percentile of the stroke deviation;
22. *Average Direction:* The average slope of the points belonging to the stroke trajectory;
23. *Length of Trajectory:* The total length of the stroke;
24. *Ratio End-To-End Distance and Length of Trajectory:* Self explanatory;
25. *Average Velocity:* The mean velocity of the stroke;
26. *Median Acceleration for First 5 Points:* Self explanatory;
27. *Mid-Action Pressure:* The pressure calculated at the midpoint of the stroke;
28. *Mid-Action Area Covered:* The area covered by finger at the midpoint of the stroke;
29. *Mid-Action Finger Orientation:* Self explanatory;
30. *Change of Finger Orientation:* Self explanatory;
31. *Phone Orientation:* Self explanatory.

5.4 Summary

In this chapter, we describe four different behavioural biometrics based datasets that are used in this thesis. Three of these datasets are publicly available and one dataset is collected by us. We also discussed the set of features extracted from the raw data for all the datasets. One major difference between a CA and a PA system is that a CA system acts on each single action, while a PA system acts on a block of actions and in that way has the possibility to use statistical features. Due to this limitation we have explored the action based features that can be found in this chapter in our research.

Chapter 6

Continuous Authentication using Mouse Dynamics

In this chapter, we will investigate the performance of an mouse dynamics based CA system under various different analysis techniques. We will test these on a publicly available continuous mouse dynamics database *i.e. Dataset-1* (see Section 5.1), but the techniques can be applied to other biometric modalities in a continuous setting also. We have tested all different combinations of fusion techniques, threshold settings, score boosting techniques and static versus dynamic trust models. We show that the optimal performance we can reach with our new techniques improves significantly over the best known performance on the same dataset.

This chapter is based on the papers published in: [93] MONDAL, S., AND BOURS, P. Continuous authentication using mouse dynamics. In *Int. Conf. of the Biometrics Special Interest Group (BIOSIG'13)* (2013), IEEE, pp. 1–12 and [95] MONDAL, S., AND BOURS, P. A computational approach to the continuous authentication biometric system. *Information Sciences 304* (2015), 28 – 53.

6.1 Background Knowledge

For our analysis, we tested two different classification algorithms in a multi-modal architecture. These were *Support Vector Machine (SVM)* and *Artificial Neural Network (ANN)*. Some details of SVM and ANN and the basic understanding of multi-classifier fusion are presented below.

6.1.1 Support Vector Machine

Support Vector Machine (SVM) is a very well-known supervised learning algorithm which can be used for classification problems [25]. This classifier is capable of creating a linear decision margin that is as wide as possible, depending on the *Support Vectors (SV)*. The SV are those data points from the different classes that are closest to the decision line. In this research, we have used the LibSVM software distribution for the SVM classifier [27]. The main motivation to use LibSVM is not only because it is a well implemented optimization technique for the cost function of SVM and widely used in the research community, but also because it will provide the classification score along with the class label. In our research, we are going to use the classification score for our analysis. Initially we tried SVM with a *Linear Kernel*, but found that the classifier did not perform well due to the small feature set (see Section 5.1.2). We decided to use *Gaussian Kernel* as a similarity measure function in this research.

6.1.2 Artificial Neural Network

Artificial Neural Network (ANN) is a combination of multiple artificial neurons which can be used for classification and regression analysis [16]. In our research, the neurons consist of a *linear* activation function with a 2-layer *Feed-Forward* neural network. In this research, we have used the NETLAB software distribution for the ANN classifier [111]. NETLAB has a good implementation of the *Scaled Conjugate Gradient* algorithm which is efficient to optimize the cost function and also it will reduce the ANN training time. We have tested different numbers of hidden nodes, and different

regularization parameter values (α) for different users to maintain the trade of between *Bias* and *Variance* and the training time of the classifier model.

6.1.3 Multi Classifier Fusion

Multi Classifier Fusion (MCF) is a technique used to combine multiple classifiers on the same biometric modality to improve the performance of that modality [64, 75]. Researchers generally prefer to use multiple classifiers on a same modality when the modality is considered to be a weak modality. The architecture of the MCF technique is very much similar to the multi-modal biometric technique but, with the MCF technique only score level and decision level fusion are possible [126]. For our analysis, we have used SVM and ANN classification algorithms in a multi-modal architecture with score level fusion to improve the performance of the mouse dynamics modality for continuous authentication.

6.2 Contributed Algorithms

In this section we describe two additional algorithms that are used in the analysis. These algorithms are related to combining classification scores of different classifiers and boosting the raw score of a classifier.

6.2.1 Score Boost

In this section, we are going to discuss our score boosting algorithm which we have applied. As we are dealing with the classification score of the current action to decide the genuineness of the current user, we have found that there is a huge overlap between genuine and imposter scores on the training set. Therefore, we are trying to find some way which can significantly improve the performance of the system in this situation. The score boost technique will significantly boost the score in a particular range. Therefore, the genuine action will get a higher reward and the imposter action will get a higher penalty. In Algorithm 6.1, we show the score boost technique. This algorithm is capable to boost the classification score of the current action based on some parameters. The score will boost towards a higher order if the score is above the value of parameter C and if the score is below this value, it will boost towards a lower order. The parameter W is the width where the score will remain unchanged from $C - \frac{W}{2}$ to $C + \frac{W}{2}$. The parameter P was used for the order of the score boost. In Figure 6.1 we have shown the nature of this algorithm for different parameters.

6.2.2 Weighted Fusion Scheme

In Algorithm 6.2, we describe the weighted fusion scheme. This algorithm takes several parameters and computes the weights for the fusion of the SVM (WT_{SVM}) and ANN (WT_{ANN}) classifiers. Based on the accuracy difference of the two single classifiers this algorithm will decide the weights for score fusion. We use $Tolerance$ to neglect the amount of accuracy difference and put the same weight for both the classifiers. The slope of the function can be adjusted by the $Slope$ parameter. According to the primary principals of a weighted fusion scheme [141], the algorithm should satisfy the two criteria, $0 \leq WT_i \leq 1, \forall i$ and $\sum_{i=1}^{n} WT_i = 1$, where n is the number of classifiers used in the system. The maximum weight on a classifier can be adjusted by $UpperLimit$ parameter. If we can set the $UpperLimit$ parameter with 1, this algorithm can be used as a classifier selection. In Figure 6.2 we have shown the Accuracy difference vs. Weight calculation from the Equation 6.1 in Algorithm 6.2 for the different parameters.

6.3 System Architecture

In this section, we will discuss the methodology of our analysis. The CA system is divided into two basic phases (see also Figure 6.3):

Algorithm 6.1: Algorithm for Score Boost.

Data:
$sc_i \leftarrow$ Classification score i^{th} action
$C \leftarrow$ Center
$W \leftarrow$ Width of left unchanged
$P \leftarrow$ Power of boost
Result:
$X_i \rightarrow$ Boosted classification score i^{th} action

begin
  if $sc_i < (C - \frac{W}{2})$ **then**
    if $sc_i \geq 0$ **then**
      $X_i = \frac{(sc_i)^P}{C - \frac{W}{2}}$
    else
      $X_i = -|sc_i|^{\frac{1}{P}} - (C - \frac{W}{2})$
  else
    if $(C - \frac{W}{2}) \leq sc_i \leq (C + \frac{W}{2})$ **then**
      $X_i = sc_i$
    else
      $X_i = C + \frac{W}{2} + ((1 - c - \frac{W}{2}) \times (sc_i - c - \frac{W}{2}))^{\frac{1}{P}}$

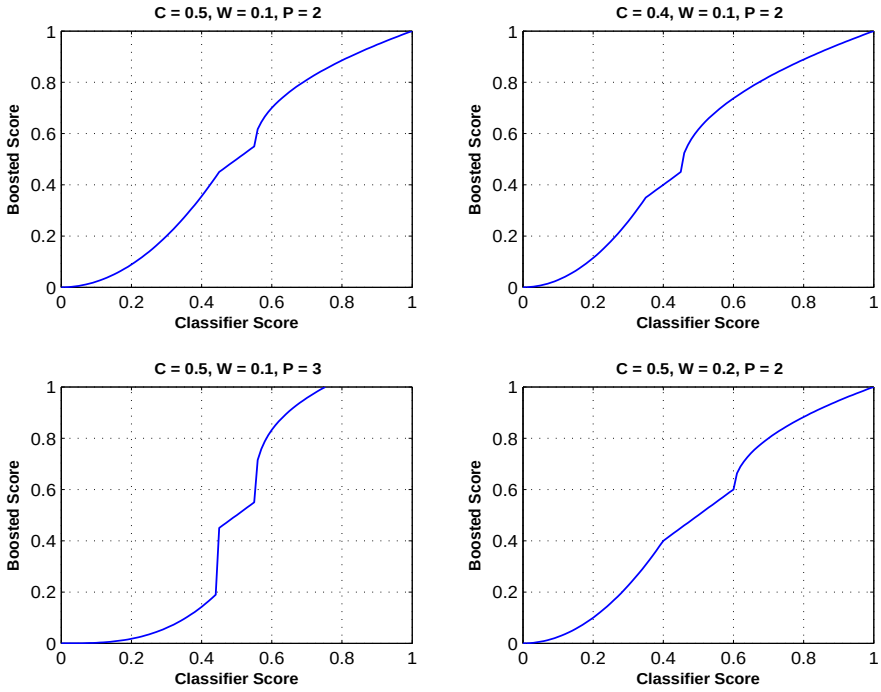


Figure 6.1: Classifier score vs. Boosted score from Algorithm 6.1 with different parameters.

I. Training Phase: In the training phase, the training data (see Section 4.2.2) is used to build the

Algorithm 6.2: Algorithm for Weighted Fusion Scheme.

Data:
$Accuracy_{svm} \leftarrow$ Accuracy of SVM classifier
$Accuracy_{ann} \leftarrow$ Accuracy of ANN classifier
$Slope \leftarrow$ Slope of the function
$Tolerance \leftarrow$ Parameter for tolerance of the classifiers accuracy difference
$UpperLimit \leftarrow$ Upper limit of the classifier weight ($0.5 < UpperLimit \leq 1$)
Result:
$WT_{SVM} \rightarrow$ Weight for SVM classifier
$WT_{ANN} \rightarrow$ Weight for ANN classifier

begin
  $Abs_{diff} = |Accuracy_{svm} - Accuracy_{ann}|$
  $Diff_{acc} = Abs_{diff} - Tolerance$

$$WT = \min\{0.5 + \max\{Diff_{acc} \times Slope \times 0.1, 0\}, UpperLimit\} \tag{6.1}$$

  if $Accuracy_{svm} > Accuracy_{ann}$ **then**
    $WT_{SVM} = WT$
    $WT_{ANN} = 1 - WT$
  else
    $WT_{SVM} = 1 - WT$
    $WT_{ANN} = WT$

classifier models and store the models in a database for use in the testing phase. Each genuine user has his/her own two classifier models (SVM and ANN). This means that we build two sets of 49×2 different classifier models for *VP-1* and *VP-2* and one set of 49×4 different classifier models for *VP-3*. Recall that *Dataset-1* has 49 users.

II. Testing Phase: In the testing phase we use test data, which was separated from the training data, for comparison. In the comparison, we will use the models stored in the database and obtain the classifier score on each sample of the test data. This score will then be used to update the trust value $Trust$ in the trust model (see Sections 3.2 and 3.3). Finally, the trust value $Trust$ is used in the decision module, to determine if the user will be locked out or can continue using the PC. This decision is made based on the current trust value and the lockout threshold ($T_{lockout}$).

In Figure 6.4 we show the extended block diagram for the Trust Calculation module (see Figure 6.3) without any classifier score normalization and score boosting as discussed in Section 6.2.1. Figure 6.5 shows the extended block diagram for the *Trust calculation* module with classifier score normalization and/or score boosting technique. The score normalization or score boost is used for the pre-processing of the raw scores as they are produced by the classifiers. We have made this step optional and done separate performance analysis with and without the pre-processed score, which we will discuss in the Section 6.4. We also measured the performance of the system by applying the min-max score normalization techniques [141] and our proposed score boosting technique (see Section 6.2.1). After this step, both classifier scores will go to the Fusion Rules module. In this module, we applied different state of the art fusion rules like, average, min, max [126] and our proposed weighted fusion scheme (see Section 6.2.2). Finally the fused score will go to the *Trust Model* (see Chapter 3) and the current system trust in the genuineness of the user will be calculated. Based on the current system trust the system will decide if the user can continue his/her work or if that user should be locked out from the system.

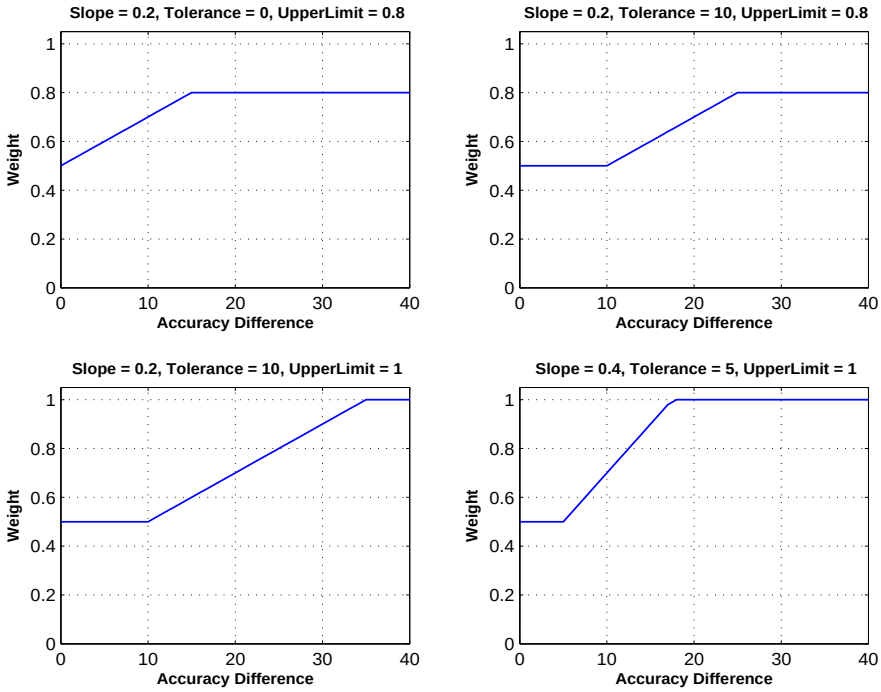


Figure 6.2: Accuracy difference vs. Weight from Equation 6.1 with different parameters.

6.4 Result Analysis

In this section, we analyse the results that we got by applying the algorithms discussed in Section 6.2. We divide our analysis into two major parts based on the type of trust model (see Sections 3.2 and 3.3). As we have mentioned in Section 5.1, in the used dataset the total number of genuine users is 49 and hence the total number of data sets of imposter users is 2352 (49×48). We will report the results from a zero effort attack scenario *i.e.* one-hold-out test in terms of ANIA and ANGA along with the total number of imposters not detected for a fixed lockout threshold ($T_{lockout}$) as well as for a user specific lockout threshold (T_{us}), where the threshold for lockout will be $T_{us} = \max(50, \min(Trust_{genuine}))$.

Besides reporting the performance in terms of ANIA and ANGA and in terms of the 4 categories described below, we will also report the results in terms of FMR and FNMR. We do realize that this is a slight abuse of terminology in the context of continuous authentication with our analysis methods, but we decided to do so to clarify the results in known terminology. These results can be found in Section 6.4.3.

6.4.1 Analysis of Static Trust Model

We applied the 3-level and 4-level STM with different penalty and reward thresholds (see Section 3.2 for the algorithms). Also, we applied different functions to calculate the change in trust Δ_T. We have performed a separate analysis with the boosted classifier scores and without boosted classifier scores.

We applied the STM without the Score Boost and with Score Boost technique followed by the Trust calculation architecture given in Figure 6.4. After applying different threshold parameters for

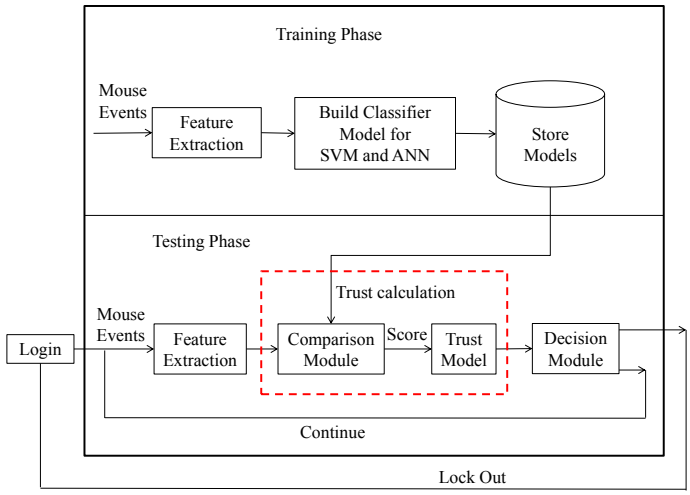


Figure 6.3: Block diagram of the system.

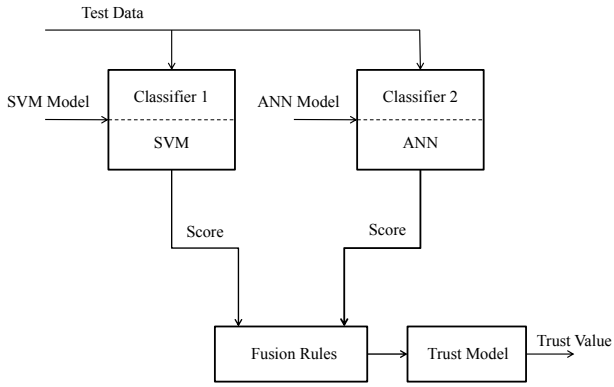


Figure 6.4: Block diagram of the *Trust Calculation* module (see Figure 6.3) without score normalization or score boosting.

3-level and 4-level trust models, we got the best results from the 3-level STM. We have applied different fusion rules and found the best results from the Average Fusion rules

$$sc_i = \frac{SCORE_{SVM} + SCORE_{ANN}}{2}$$

and our proposed Weighted Fusion scheme

$$sc_i = WT_{SVM} \times SCORE_{SVM} + WT_{ANN} \times SCORE_{ANN}.$$

We also tested the different parameter values for our proposed weighted fusion scheme (see Section 6.2.2 for the algorithm) to optimize the system performance.

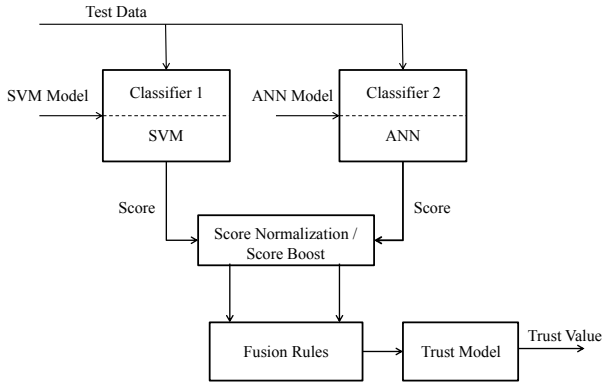


Figure 6.5: Block diagram of the *Trust Calculation* module (see Figure 6.3) with score normalization and/or score boosting.

Optimal algorithmic parameters for the above results are given below:

3-level STM: The parameters for the Algorithm 3.1 are:

- Threshold parameters, $Tr = 0.5$ and $Tr_P = 0.3 \sim 0.4$.
- Reward function, $f_{Reward}(sc_i) = 1 \times sc_i$, where sc_i is the fused classification score.
- Penalty functions, $f^1_{Penalty}(sc_i) = 1 - sc_i$ and $f^2_{Penalty}(sc_i) = 1$, where sc_i is the fused classification score.

Weighted Fusion: The parameters for Algorithm 6.2 are, $Slope = 0.4$, $Tolerance = 10$ and $UpperLimit = 1$.

6.4.1.1 Without Score Boost

Table 6.1 shows the optimal result we got from this analysis for *VP-1* (see section 4.2.2.1). The table is divided into two parts based on the fusion rules. In the Average fusion rules, only 43 users fall into the best case category (*i.e.* '+ / +' category) for the user specific lockout threshold; whereas in weighted fusion, this number is 45. We can clearly observe from the table that, if we go from user specific lockout threshold (T_{us}) to fixed lockout threshold ($T_{lockout} = 90$), the results are getting worse. In the fixed lockout threshold, the number of best performing users are decreasing, and the numbers for *Positive vs. Negative* and *Negative vs. Positive* are increasing. We observed from the analysis that, some of the users have $T_{us} > 90$ and those users are contributing to the *Positive vs. Negative* category and some of the users have $T_{us} < 90$ and those users are contributing to the *Negative vs. Positive* category.

Table 6.2 shows the optimal result, we got from this analysis for *VP-2* (see section 4.2.2.2), and Table 6.3 shows the optimal result for *VP-3* (see section 4.2.2.3). By comparing all these three tables we can observe that *VP-2* gives better results compared to the other VPs. Generally we have noted that weighted fusion with user specific thresholds produces the best result where all the genuine users never locked out from the system and 2% of the imposters are not being detected for one user.

We noticed furthermore that the results for *VP-1* are not the best, although this might have been unexpected beforehand because the models are trained with data of all of the imposters.

Table 6.1: Results for *VP-1* with the analysis method of STM without Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	43		68		45		84	
	+ / -	3		440	3	4		1326	32
	- / +								
	- / -	3	2066	2076	42				
	Summary	49	44889	1088	45(1.9%)	49		816	32(1.4%)
90	+ / +	35		92		34		88	
	+ / -	6		1000	22	5		600	13
	- / +	7	4216	103		9	7585	84	
	- / -	1	323	602	5	1	5782	2171	10
	Summary	49	40506	752	27(1.1%)	49	39462	636	23(1%)

Table 6.2: Results for *VP-2* with the analysis method of STM without Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	48		109		48		131	
	+ / -	1		366	1	1		366	1
	- / +								
	- / -								
	Summary	49		134	1(0.04%)	49		156	1(0.04%)
90	+ / +	35		171		43		210	
	+ / -	6		781	11	3		543	3
	- / +	8	32311	138		3	35009	98	
	- / -								
	Summary	49	45172	460	11(0.5%)	49	46906	284	3(0.1%)

Table 6.3: Results for *VP-3* with the analysis method of STM without Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	47		130		47		91	
	+ / -	2		1026	2	2		592	4
	- / +								
	- / -								
	Summary	49		206	2(0.09%)	49		192	4(0.2%)
90	+ / +	34		172		41		171	
	+ / -	8		662	16	6		597	8
	- / +	5	31450	93		2	54401	158	
	- / -	2	81004	960	2				
	Summary	49	47386	636	18(0.8%)	49	47956	383	8(0.3%)

6.4.1.2 With Score Boost

Next we applied the STM with the Score Boost technique followed by the Trust calculation architecture given in the Figure 6.5. Similar to the previous settings we have found that the 3-level STM with average fusion and weighted fusion performed better than other settings. The Algorithm 6.1 parameters are, $C = 0.5$, $W = 0.00 \sim 0.05$ and $P = 2$.

We have also applied the min-max score normalization techniques on *VP-1* and found that the number of best performing users is low (30 users) and the ANIA is high (129 actions) for the user specific lockout threshold.

Table 6.4 shows the optimal results we obtained from this analysis for *VP-1*. Here, the trust model and weighted fusion parameters are the same as in the previous analysis. The results we got from this analysis are very much comparable to the previous settings. According to this analysis we

Table 6.4: Results for *VP-1* with the analysis method of STM with Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	43		79		45		87	
	+ / -	4		1231	28	4		1837	25
	- / +								
	- / -	2	366	1085	10				
	Summary	49	45751	965	38(1.6%)	49		717	25(1.1%)
90	+ / +	32		87		34		85	
	+ / -	7		777	31	5		598	15
	- / +	9	2575	94		9	4807	76	
	- / -	1	1890	1401	10	1	12037	3543	13
	Summary	49	38463	1029	41(1.7%)	49	39080	751	28(1.2%)

Table 6.5: Results for *VP-2* with the analysis method of STM with Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	48		127		49		109	
	+ / -	1		589	1				
	- / +								
	- / -								
	Summary	49		156	1(0.04%)	49		109	0(0%)
90	+ / +	29		142		41		150	
	+ / -	8		669	12				
	- / +	12	13626	133		8	2880	72	
	- / -								
	Summary	49	39342	466	12(0.5%)	49	40367	137	0(0%)

Table 6.6: Results for *VP-3* with the analysis method of STM with Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	41		99		44		77	
	+ / -	4		807	7	4		523	4
	- / +	3	4630	164		1	34457	185	
	- / -	1	90936	512	1				
	Summary	49	45929	329	8(0.3%)	49	47412	196	4(0.2%)
90	+ / +	26		89		34		88	
	+ / -	4		1351	9	2		948	4
	- / +	18	6971	100		13	14008	94	
	- / -	1	80279	383	1				
	Summary	49	33392	399	10(0.4%)	49	38748	204	4(0.2%)

can say that the previous setting performs slightly better than this system due to the nature of the STM.

Tables 6.5 and 6.6 show the optimal results we obtained from this analysis for *VP-2* and *VP-3* respectively. Similar to the previous analysis, we have found that *VP-2* performs better than the other two and more precisely, weighted fusion with user specific threshold performs best where all the genuine users never locked out and all the imposters were detected by the system.

6.4.2 Analysis of Dynamic Trust Model

To overcome the limitations of the STM (see Section 3.2) and to further improve on the performance of the system, we have applied the DTM. The algorithm of the DTM was provided in Section 3.3. This model is very similar to the 2-level trust model except that we used an efficient function to

Table 6.7: Results for *VP-1* with the analysis method of DTM without Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	46		67		47		69	
	+ / -					2		445	2
	- / +								
	- / -	3	1716	3988	70				
	Summary	49	44868	1608	70(3%)	49		125	2(0.09%)
90	+ / +	39		95		15		287	
	+ / -	7		384	11	34		872	69
	- / +								
	- / -	3	850	279	16				
	Summary	49	44815	691	27(1.1%)	49		2066	69(2.9%)

calculate the Δ_T for this model. The main advantages of this model are that (1) the penalty and reward values are defined via a continuous function; and (2) there is not a large difference in the trust change for classification scores just below or above the threshold. Due to the limited number of features we can extract from an action, there is a huge overlap between imposter and genuine actions scores. For this reason the DTM model worked extremely well on the given dataset.

Algorithmic parameters for the above results are given below,

Dynamic Trust Model: The parameters for the Algorithm 3.3 are,

$$A = \frac{1}{2n}\sum_{i=1}^{n}(T_i^{svm} + T_i^{ann}),$$

where T^{svm} and T^{ann} are the training score sets for the SVM and ANN classifiers and n is the number of samples in the training set, $B = 0.01 \sim 0.5$, $C = 1$ and $D = 1$.

Weighted Fusion: The parameters for the Algorithm 6.2 are, $Slope = 0.5$, $Tolerance = 10$, $UpperLimit = 1$.

6.4.2.1 Without Score Boost

In this section, we present the results we got from the DTM (see Section 3.3) without applying the Score Boost technique followed by the Trust calculation architecture given in the Figure 6.4. After exploring all the options for fusion rules, we found that the average and our proposed weighted fusion rules perform better than the other fusion rules.

Table 6.7 shows the optimal result we have found from this analysis for *VP-1*. This result was obtained after exploring all the parameters of Equation 3.1. These experimental results clearly indicate the potential improvement on the performance of the system over the analysis discussed above. From these results we can clearly conclude that all the users have $T_{us} > 90$ for weighted fusion. Note specifically also that in case we apply weighted fusion no genuine user is locked out by the system.

Tables 6.8 and 6.9 show the optimal results we obtained from this analysis for *VP-2* and *VP-3* respectively. Again we found that *VP-2* outperforms the other two for both the fusion techniques with user specific threshold performs best where all the genuine users never locked out and all the imposters were detected by the system. However, a fixed threshold with weighted fusion technique also produces similar result on *VP-2* which was not seen in the previous analysis.

In this analysis, there are significant differences between *VP-1* (see Table 6.7) and *VP-3* (see Table 6.9). It can be observed that all the techniques in *VP-3* produce results where genuine users are never locked out, but in case of *VP-1* only weighted fusion produces such results. Also 69 imposters are undetected for 34 users in case of *VP-1* with weighted fusion and a fixed threshold whereas the same techniques produce only 3 undetected imposters for 3 users in the case of *VP-3*.

Table 6.8: Results for *VP-2* with the analysis method of DTM without Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	49		83		49		79	
	+ / -								
	- / +								
	- / -								
	Summary	49		83	0(0%)	49		79	0(0%)
90	+ / +	48		86		49		86	
	+ / -	1		1665	1				
	- / +								
	- / -								
	Summary	49		138	1(0.04%)	49		86	0(0%)

Table 6.9: Results for *VP-3* with the analysis method of DTM without Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	46		74		48		101	
	+ / -	3		390	3	1		337	1
	- / +								
	- / -								
	Summary	49		154	3(0.1%)	49		126	1(0.04%)
90	+ / +	46		81		46		68	
	+ / -	3		509	3	3		440	3
	- / +								
	- / -								
	Summary	49		167	3(0.1%)	49		151	3(0.1%)

6.4.2.2 With Score Boost

In this section, we are going to show the results we have got from the DTM with applying the Score Boost technique followed by the Trust calculation architecture given in the Figure 6.5. The parameters for Algorithm 6.1 are, $C = 0.5$, $W = 0.00 \sim 0.05$ and $P = 2$.

Table 6.10 shows the optimal results we found from this analysis for *VP-1*. These results clearly show that this method outperforms all other methods previously discussed in this chapter. The total number of users for the '+/+' category equals 48 and for one genuine user, there was one imposter not detected.

Tables 6.11 and 6.12 show the optimal results we obtained from this analysis for *VP-2* and *VP-3* respectively. Overall, we have found this analysis performs better than all the other previous analyses where every verification process produces better results over other analyses. In this analysis, *VP-2* produces best results where all the users fall into the '+/+' category for both the fusion techniques with both the thresholds.

6.4.3 Performance reporting in FMR and FNMR

In this section we report the overall system performance in terms of FMR and FNMR, although we are aware that this is a slight abuse of terminology in the context of continuous authentication with our analysis methods. In this case we consider FMR as the probability that an imposter user is not detected when his test data is compared to the classification model of a genuine user. Each of the 48 imposter data was tested against 49 genuine users, which means that the total number of imposter tests equals $49 \times 48 = 2352$. Similarly we also define the FNMR here as the probability that a genuine user is falsely locked out by the system.

We will report here the FNMR and FMR values for all the tests done in Sections 6.4.1 and 6.4.2. We will restrict ourselves to only the optimal settings, *i.e.* the user dependent threshold T_{us} and the

Table 6.10: Results for *VP-1* with the analysis method of DTM with Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	46		62		48		71	
	+ / -					1		805	1
	- / +								
	- / -	3	10498	14462	114				
	Summary	49	45405	2554	114(4.8%)	49		106	1(0.04%)
90	+ / +	26		227		28		311	
	+ / -	20		654	42	21		662	31
	- / +								
	- / -	3	1925	5002	76				
	Summary	49	44881	2913	118(5%)	49		1081	31(1.3%)

Table 6.11: Results for *VP-2* with the analysis method of DTM with Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	49		87		49		70	
	+ / -								
	- / +								
	- / -								
	Summary	49		87	0(0%)	49		70	0(0%)
90	+ / +	49		92		49		75	
	+ / -								
	- / +								
	- / -								
	Summary	49		92	0(0%)	49		75	0(0%)

Table 6.12: Results for *VP-3* with the analysis method of DTM with Score Boost.

$T_{lockout}$	Category	Average Fusion				Weighted Fusion			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	47		82		48		87	
	+ / -	2		540	2	1		364	1
	- / +								
	- / -								
	Summary	49		141	2(0.09%)	49		113	1(0.04%)
90	+ / +	47		87		48		93	
	+ / -	2		733	2	1		364	1
	- / +								
	- / -								
	Summary	49		153	2(0.09%)	49		119	1(0.04%)

Weighted Fusion. For example, we see in Table 6.6 that in that case one genuine user is locked outed (see category '- / +') and that four imposter users are not detected (category '+ / -'). This implies that $FNMR = 1/49 = 2.0\%$ and $FMR = 4/2352 = 0.17\%$. All the results are presented in Table 6.13. In this table the verification processes are represented in the columns and the rows represent the evaluation scenarios. Here, SB means Score Boost and WSB stands for Without Score Boost.

6.4.4 Result Discussion

It is not surprising that from Tables 6.1 to 6.12 we consistently see that the personal threshold performs better than the fixed system threshold. When only considering the personal threshold, we can also see in almost all of the tables that the Average Fusion does not perform as well as the Weighted Fusion. The differences in performance between using Score Boost and not using Score Boost are

Table 6.13: Results in terms of (FNMR, FMR).

	VP-1	*VP-2*	*VP-3*
STM-WSB	(0%, 1.36%)	(0%, 0.04%)	(0%, 0.17%)
STM-SB	(0%, 1.06%)	(0%, 0%)	(2.04%, 0.17%)
DTM-WSB	(0%, 0.09%)	(0%, 0%)	(0%, 0.04%)
DTM-SB	(0%, 0.04%)	(0%, 0%)	(0%, 0.04%)

Table 6.14: Best performance for all the verification processes.

Verification Process	Type-II Error		Type-I Error				
	# User	ANGA	# User	ANIA	SD	t-value	p-value
VP-1	49	∞	48	71	47	0.92	0.46
			1	805 (# Imp. ND=1)			
VP-2	49	∞	49	70	56	0.82	0.5
VP-3	49	∞	48	87	57	0.93	0.4
			1	364 (# Imp. ND=1)			

not that large. We can see this from comparing the results in Section 6.4.1.1 and Section 6.4.2.1 with the results in Section 6.4.1.2 and Section 6.4.2.2. On the other hand, we can see that the DTM outperforms the STM.

We can consistently see that *VP-2* performs better than the other verification processes and also *VP-3* performs better than *VP-1*. Due to the varied nature of the behavioural biometric paradigm, we can clearly assume that the increment of the number of imposters on the training phase can also affect the classifier models to recognize the actions. We can validate this assumption by looking at the classifier's training accuracy where we have noted that the classifier's training accuracy was improved for *VP-2* and *VP-3* when compared to *VP-1*. Similar findings were reported in the research conducted by Pusara [121].

In CA systems, the objective is not just to achieve 0% FMR and FNMR but also reduce the number of actions performed by the imposters before getting detected by the system and to increase the number of actions performed by the genuine users before falsely locked out by the system. In total, we can conclude that the best performance that can be reached, is on average 70 actions required to detect an imposter when using personal thresholds in combination with boosted score and weighted fusion in a DTM, where none of the genuine users is falsely locked out from the system.

Table 6.14 shows the best performance for all the verification processes. In this table, we can see that none of the genuine users are lockout from the system for all the three verification processes (*i.e. VP-1*, *VP-2* and *VP-3*). We can also observe that for *VP-1*, 48 users are the best performing users (*i.e.* +/+ category) with an ANIA of 71 (*Standard Deviation (SD)* = 47, t-value = 0.92, p-value = 0.46) and one user has an ANIA of 805 with one undetected imposter. For *VP-2* we have all the 49 users are best performing users with an ANIA of 70 (SD = 56, t-value = 0.82, p-value = 0.5). In case of *VP-3*, we have found that 48 users are the best performing users with an ANIA of 87 (SD = 57, t-value = 0.93, p-value = 0.4) and one user has an ANIA of 364 with one undetected imposter. In all cases we see that the p-values indicate that the results found in this research are statistically significant [79].

6.5 Discussion

6.5.1 Significance of this new scheme

We found that the state of the art research on continuous authentication used either the whole test set or over a large, fixed number of actions for the analysis. This implies that an imposter at least can perform that fixed number of actions before the system checks the identity. This is then in fact

no longer continuous authentication, only at best periodic authentication. As for our understanding actual continuous authentication should react on every single action performed by the user. By doing this the challenges are eminently because, the system cannot apply any *Schematic features* (*i.e.* Mouse action histogram, Percentage of silent periods, Distribution of cursor positions *etc.*) for the analysis, which might be helpful to recognize the user. However, still we believe that this new scheme should be the correct approach for continuous authentication systems.

6.5.2 Significance of Dynamic Trust Model

The proposed DTM trust model (see Section 3.3) is a very significant contribution to the CA research. This is a powerful algorithm to handle the huge overlap between genuine user scores and the imposter users scores. In Figure 6.6 we have shown the distribution of the classifier scores after average fusion for the best performing user where on an average 9 actions were required to detect an imposter for that user. We see in this figure that the median score of this user is higher than 75% of all scores of any other user. But for other users the distribution of the classifier scores after average fusion looks more like Figure 6.7. In this case the score distribution of the genuine user does not differ a lot from the distributions of scores of other users. However, the algorithm described in our research will still produce satisfactory results. Please note that these two figures were generated by the classifier scores without using the score boost technique.

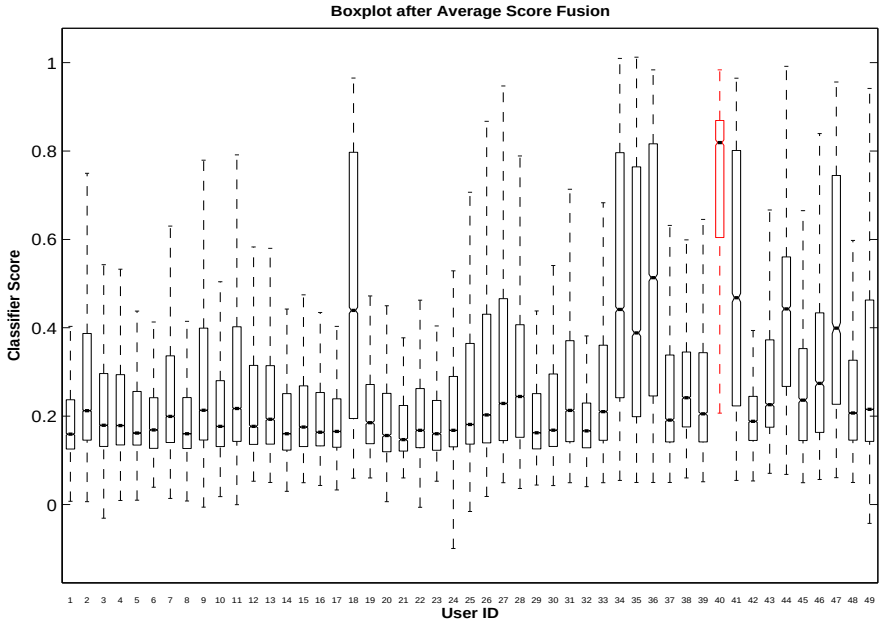


Figure 6.6: Distribution of the classifier score after average fusion for the best performing user for *VP-2*. Red mark of this box-plot is the genuine user's score distribution.

6.5.3 Algorithmic parameters

In this research, all the algorithmic parameters were optimized based on the training dataset, more precisely, the score distribution of the training data. Apart from that have we applied sequential searching for the parameters to obtain the optimized results. We first tried different fixed thresholds

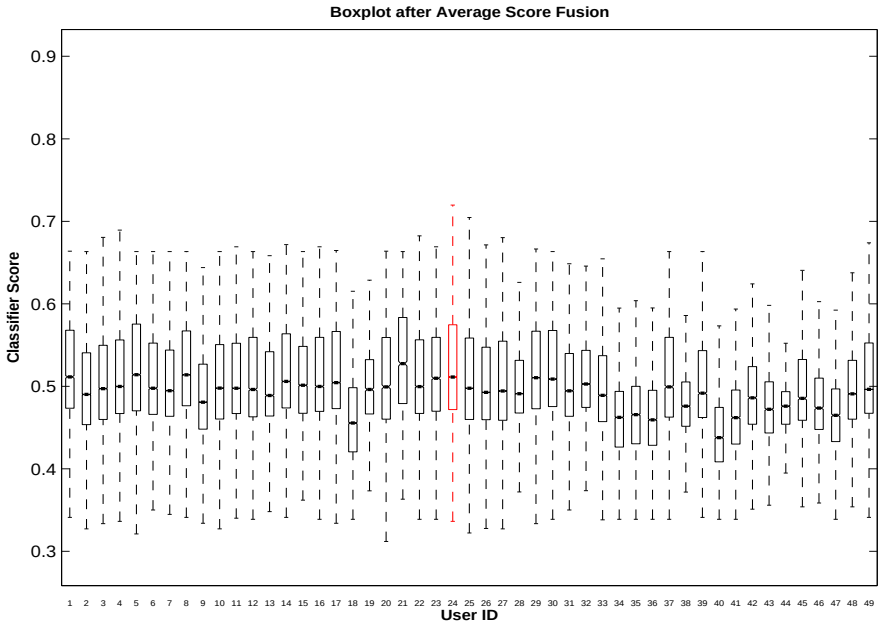


Figure 6.7: Distribution of the classifier score after average fusion for the below average performing user for *VP-2*. Red mark of this box-plot is the genuine user's score distribution.

as $T_{lockout}$ ranging from $50 \sim 90$ and then choose the user specific lockout threshold (T_{us}), where we have tried to avoid the genuine user from being locked out from the system.

We would like to mention that all the algorithms used in this research are sensitive to the parameters, meaning that the choice of one wrong parameter can lead to a disappointing result. We can understand this by observing the Figure 6.7, where the room for wrong parameter selection is huge. Using a cross validation set with optimization algorithms (*i.e.* Genetic Algorithm, Particle Swarm Intelligence *etc.*) for algorithmic parameters optimization can improve the results, but due to some limitations of this dataset (see Section 5.1.1), we are unable to build a validation set. Because of the unavailability of any other proper publicly available continuous authentication dataset on mouse dynamics, we were bound to apply our research on this dataset. However, the algorithms and methods applied in this research can be applied on any continuous authentication dataset, irrespective of the biometric modality, as long as classification scores are available.

6.5.4 Discussion on Context Dependency

As stated previously, the used dataset was collected in an uncontrolled environment in a continuous manner. On average this dataset has 14 hours of data per user. However, we do not know how much variation of the data has been introduced during data collection because this dataset was collected in a completely uncontrolled environment. In our research, we have used a single biometric signature per user, but we do feel that performance can be improved by using multiple biometric signatures for the same user, each related to a specific situation. Data clustering can be used to produce multiple signatures for the same user, but we need more experimental evidence to validate this argument which is beyond the scope of this dataset.

6.5.5 Significance of the multiple classifiers

We applied 3-level STM for *VP-2* with SVM as a classifier on the same dataset and obtained an ANIA of 136 for 46 best category users when using a user specific lockout threshold (T_{us}). We then applied ANN with the same analysis technique and obtained an ANIA of 103 for 43 best category users. Table 6.15 shows the result obtained from this analysis for *VP-2* with the analysis method of STM without Score Boost. During this analysis, we observed that some users perform better with one classifier than with the other classifier. Therefore, we have used multi classifier fusion technique to obtain the optimum result. We can compare this result with the result shown in Table 6.2 where we can observe the clear improvement on the result by using multi-classifier fusion.

Table 6.15: Results for *VP-2* with the analysis method of STM without Score Boost.

Category	SVM				ANN			
	# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
+ / +	46		136		43		103	
+ / -	3		1958	9	6		412	8
- / +								
- / -								
Summary	49		423	9(0.4%)	49		302	1(0.3%)

6.5.6 Comparison with previous research

Nakkabi *et al*. [112] achieved an FMR of 0% and an FNMR of 0.36% for 48 users with the same dataset that we have used in our analysis. The number of actions they have used in this research is a bit unclear; but, in their previous research they mentioned blocks of 2000 actions for the analysis [2]. Therefore, if we consider the same number of actions and convert it in terms of ANIA/ANGA (discussed in section 4.3.1) then $ANIA = 2000$ and $ANGA \approx 555556$. The ANGA is comparable with our result, meaning that the genuine user is in practice never locked out from the system but, the ANIA is very high compared to our result. Also, we need to mention that they have gotten this result on 48 out of 49 users.

6.6 Summary

Our analysis was complete in the sense that we tried three different verification processes and all different combinations of the 4 different settings that we used (threshold setting, score boosting, dynamic vs. static trust model, and fusion). We applied the analysis techniques to a dataset with continuous mouse dynamics data, but the techniques are general enough to be applied to other biometric data for continuous authentication, such as keystroke dynamics data. We have shown that our results improve significantly over the previously known performance results on this dataset, even though it is hard to compare the results in a straightforward manner.

Our analysis was focused on the Trust calculation module (see the red marked section on Figure 6.3). We would like to mention that different pre-processing techniques or different classifier selection techniques can influence the results.

Chapter 7

Continuous Authentication using Keystroke Dynamics

In this chapter, we discuss how keystroke dynamics can be used for true continuous authentication. We have collected keystroke dynamics data of 53 participants who used the computer freely and we have analysed the collected data. We will describe a system that decides on the genuineness of the user based on each and every single keystroke action of the current user.

This chapter is based on the papers published in: [94] MONDAL, S., AND BOURS, P. Continuous authentication using fuzzy logic. In *7th Int. Conf. on Security of Information and Networks (SIN'14)* (2014), ACM, pp. 231–238 and [21] BOURS, P., AND MONDAL, S. *Continuous Authentication with Keystroke Dynamics*. Science Gate Publishing, 2015, ch. Recent Advances in User Authentication Using Keystroke Dynamics Biometrics, pp. 41–58.

7.1 Data Processing

In this chapter, we have used *Dataset-2* for the analysis. The data collection software collected both keystroke information as well as mouse related activities, but we have only used the keystroke data in this study. On average a user provided 47600 keystroke related actions. The extracted features from the raw data are discussed in the Section 5.2.5.

We have separated our data to build and train the system (training dataset), parameter adjustment of the algorithms used in this research (validation dataset) and finally to test the system performance (test dataset). 35% of the total data was used as a training dataset, 10% was used as a validation dataset and the remaining 55% of the data was used for testing. In fact, if a user provided a large amount of keystroke related data then the amount used for training was limited to a maximum of 20000, which means that even more data of that user would be available for testing.

7.2 Matching Module

We have followed two different approaches for classification of keystroke actions (see Section 5.2.5 about the two different types of actions). The complete description of these approaches are given below.

7.2.1 Statistical Approach

In this approach, no imposter data was used in the template building phase or training phase and 50% of the imposter users (*i.e.* validation data of 26 subjects) is used for algorithmic parameter adjustments. The template of each individual was built from the training data of that individual and we applied distance based metric in the matching module (see Figure 7.5). Two separate approaches were applied for score calculation and the descriptions of these approaches are given below.

7.2.1.1 Statistical Approach - 1 (SA-1)

For a *Single Key Action* classification, we have used pairwise *Scaled Euclidean Distance (SED)* for particular key observations for this verification process. For example, the key `a` has n observations in the training data, then we will get n SED values for a test sample of `a`. The distance metric

vector used is (f_1, f_2, f_3) = (mean, minimum, maximum) of these n distance values. From these 3 attributes we are going to calculate a score that is going to be used in the *Trust Model* (see Section 3.3) in the following way:

$$sc = 1 - \frac{f_1 - f_2}{f_3 - f_2}$$

For a *Key Digraph Action*, we have four attributes in the feature vector. Here we have used two distance metrics, *i.e.* pairwise SED and *Correlation Distance (CD)* for particular key observations. For example, assume the key digraph `ab` has n observations in the training data. Now, we will get n SED values and n CD values for a test sample of digraph `ab`. Then we define the distance metric vector as (f_1, f_2, f_3) = (mean of SED, minimum of SED, maximum of CD). From this we calculate the score sc used in the *Trust Model* of Section 3.3 in the following way:

$$sc = \frac{f_1 \times f_3}{f_2}.$$

7.2.1.2 Statistical Approach - 2 (SA-2)

In this approach, we have used same the distance metric for the two keystroke actions as discussed above, but applied fuzzy logic for the score calculation that is going to be used in the *Trust Model*.

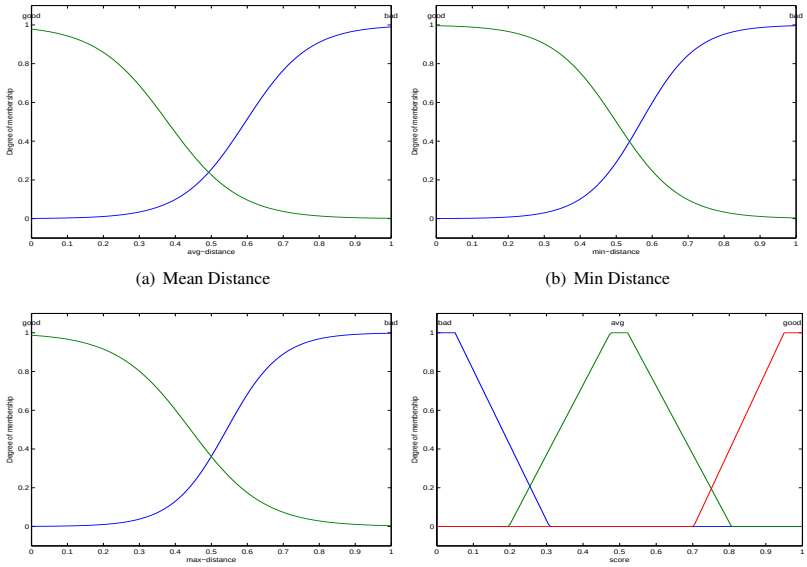
Fuzzy Logic is a mathematical approach to compute the approximate value based on the multi-valued logic [152]. In our research, we have applied fuzzy logic technique to convert the distance metric into score value. Also, we would like to mention that the same fuzzy logic controller was used for all the participants and two different types of keystroke actions. In Figure 7.1 we have shown the membership functions used in this research. We have followed *Sigmoid* membership functions for input variables and more precisely the distance metrics described in previous section. The output variable was used as *Trapezoidal* membership function. Figures 7.2, 7.3, and 7.4 are the surface plot representation of the input and output variables by applying the fuzzy logic rules described below:

1. (mean-distance=bad) $\wedge$ (min-distance=bad) $\wedge$ (max-distance=bad) $\Longrightarrow$ bad
2. (mean-distance=bad) $\wedge$ (min-distance=bad) $\wedge$ (max-distance=good) $\Longrightarrow$ bad
3. (mean-distance=bad) $\wedge$ (min-distance=good) $\wedge$ (max-distance=bad) $\Longrightarrow$ average
4. (mean-distance=bad) $\wedge$ (min-distance=good) $\wedge$ (max-distance=good) $\Longrightarrow$ bad
5. (mean-distance=good) $\wedge$ (min-distance=bad) $\wedge$ (max-distance=bad) $\Longrightarrow$ good
6. (mean-distance=good) $\wedge$ (min-distance=bad) $\wedge$ (max-distance=good) $\Longrightarrow$ average
7. (mean-distance=good) $\wedge$ (min-distance=good) $\wedge$ (max-distance=bad) $\Longrightarrow$ good
8. (mean-distance=good) $\wedge$ (min-distance=good) $\wedge$ (max-distance=good) $\Longrightarrow$ good

7.2.2 Machine Learning Approach

We also followed a *Machine Learning Approach (MLA)* with three verification processes discussed in Section 4.2.2. In this analysis, we used three different classifiers *i.e. Artificial Neural Network (ANN)*, *Counter-Propagation Artificial Neural Network (CPANN)* and *Support Vector Machine(SVM)* with a *Multi-Classifier Fusion (MCF)* architecture. The description of the SVM and ANN are provided in Sections 6.1.1 and 6.1.2 respectively, while the description of the CPANN is given below.

Counter-Propagation Artificial Neural Network (CPANN) is a hybrid learning mechanism based on an Artificial Neural Network to handle the supervised problems. In CPANN, the output layer is added to the *Kohonen* layer which is very similar to *Self Organizing Maps* and provides both the advantages of supervised and unsupervised learning. It can also guarantee to find the correct network



(a) Mean Distance (b) Min Distance

(c) Max Distance (d) Output Score

Figure 7.1: Membership functions of our fuzzy logic system.

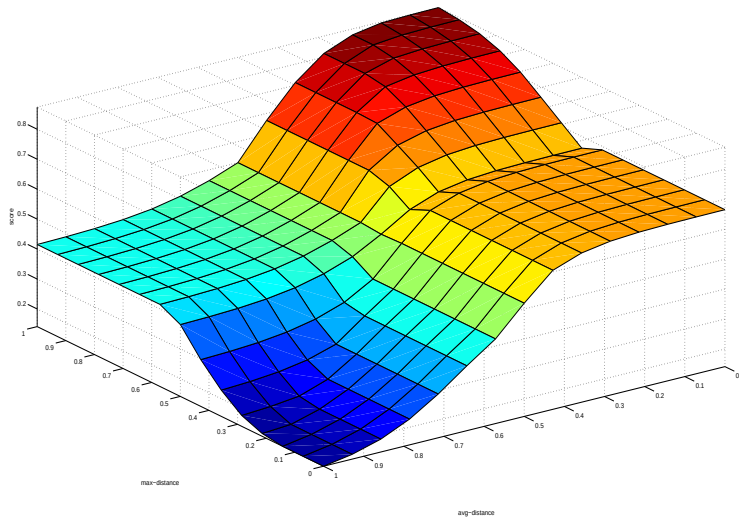


Figure 7.2: Surface plot for Score vs. Mean and Max distance.

weights, which can be seen as a drawback of regular back-propagation networks. We have used a free CPANN software distribution for our research [84]. The number of neurons is optimized for different users.

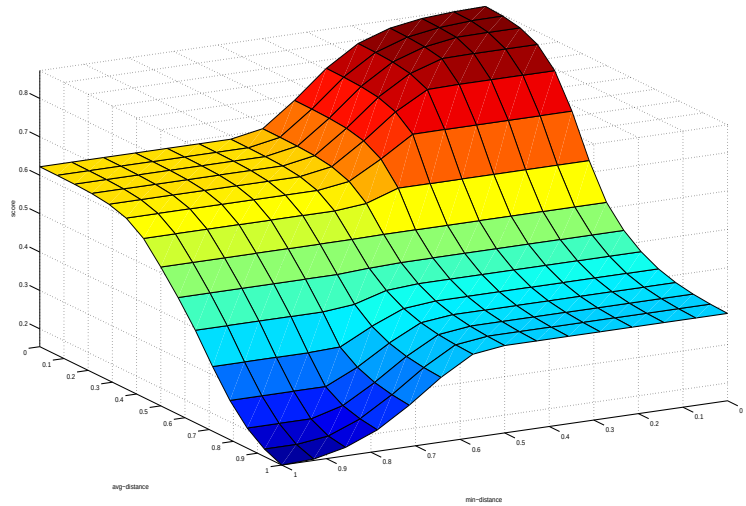


Figure 7.3: Surface plot for Score vs. Mean and Min distance.

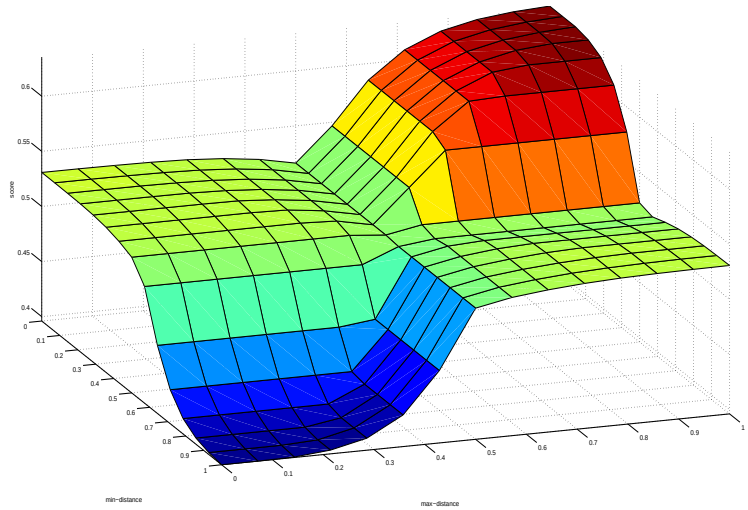


Figure 7.4: Surface plot for Score vs. Max and Min distance.

The score vector we use for further analysis is $(f^1, f^2, f^3) = (Score_{ann}, Score_{svm}, Score_{cpann})$. From this score vector, we calculate a score that is used in the *Trust Model* (see Section 3.3) in the

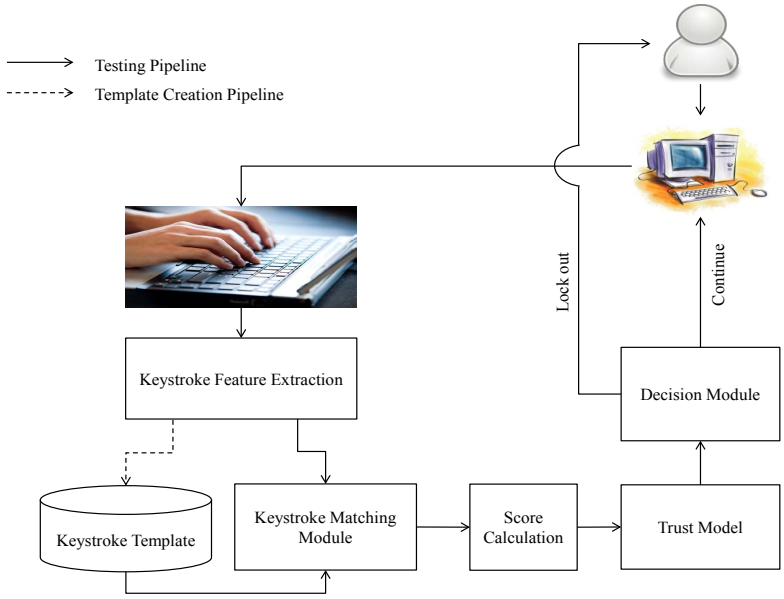


Figure 7.5: Block Diagram of the proposed system.

following way:

$$sc = (\sum_{j=1}^{3} w_j f^j)/(\sum_{j=1}^{3} w_j)$$

The weights were optimized using *Genetic Algorithm (GA)* [88] with a validation dataset.

7.3 System Architecture

In this section, we discuss the methodology of our system. The system was divided into two phases (see Figure 7.5).

In the training phase, the training data is used to build the classifier models and store the models in a database for use during the testing phase (marked as dotted arrow in Figure 7.5). Each genuine user has his/her own classifier models and training features.

In the testing phase, we use the test data, which was separated from the training data, for comparison. In the comparison, we will use the models and training features stored in the database and obtain the classifier score (probability) on each sample of the test data according to the performed action. This score will then be used to update the trust value $Trust$ in the trust model (see Section 3.3). Finally, the trust value $Trust$ is used in the decision module, to determine if the user will be locked out or can continue to use the PC. This decision is made based on the current trust value and the lockout threshold ($T_{lockout}$).

For each action performed by the current user, the system calculates the score for that action (see for details the subsections in Section 7.2) and used that to calculate the change in trust according to Equation 3.2. The parameters A, B, C, and D in Equation 3.1 are dependent on the type of action *i.e. Single Key Action* or *Key Digraph Action* and were optimized by GA optimization technique, where the cost function was to maximize $[ANGA - ANIA]$.

Table 7.1: Results obtained from statistical approach.

Category	SA-1				SA-2			
	# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
+ / +	30		443		12		402	
+ / -	11		1665	23	10		3575	50
- / +	10	2389	614		21	1073	471	
- / -	2	13499	1127	3	10	4922	1823	35
Summary	53	21228	987	26(0.9%)	53	12229	2016	85(3.1%)

Table 7.2: Results obtained from machine learning approach.

	Category	# User	ANGA	ANIA	# Imp. ND
VP-1	+ / +	28		381	
	+ / -	12		1099	28
	- / +	11	3851	383	
	- / -	2	3800	1282	5
	Summary	53	20716	878	33(1.2%)
VP-2	+ / +	30		235	
	+ / -	14		950	29
	- / +	5	3381	331	
	- / -	4	20185	875	9
	Summary	53	23593	830	38(1.4%)
VP-3	+ / +	35		269	
	+ / -	12		766	28
	- / +	4	2466	300	
	- / -	2	20128	1242	8
	Summary	53	24180	751	36(1.3%)

7.4 Result Analysis

In this section, we analyse the results that we got from this research. We performed a zero effort attack scenario, *i.e.* leave-one-out testing. Therefore, we have test data of 1 genuine user and 52 imposter users. The total number of data sets of genuine users is 53 and the total number of data sets of imposter users is $53 \times 52 = 2756$. We report the results in terms of ANIA and ANGA along with the total number of imposters not detected for person based lockout threshold ($T_{lockout}$) which was optimized by using GA.

Table 7.1 shows the result we got from our analysis with the statistical approaches. We observe from the table that 30 participants qualify the *'+ / +'* category, where the ANIA is 443 actions for *SA-1* and 12 participants qualify the *'+ / +'* category, where the ANIA is 402 actions for *SA-2*. In the category *'+ / -'* we find 11 participants for *SA-1* and 10 participants *SA-2*, were genuine users that are not locked out, but the ANIA related to these users is relatively high (*i.e.* 1665 actions with 23 imposters undetected for *SA-1* and 3575 actions with 50 imposters undetected for *SA-2*). The genuine users in the *'- / +'* category were locked out at least once by the system (*i.e.* 10 participants for *SA-1* and 21 participants for *SA-2*). The remaining participants, both for *SA-1* and *SA-2* are falling into the *'- / -'* category. We can clearly observe from this table that the *SA-1* approach performs better than the *SA-2* approach.

Table 7.2 shows the result we got from our analysis with machine learning approach for *VP-1*, *VP-2* and *VP-3* (see Section 4.2.2 for the description of *VP-1*, *VP-2* and *VP-3*). *VP-2* seems to give better results than other verification processes, where the ANIA for the first category is the primary concern (*i.e.* *'+ / +'* category). On the other hand, we see that more genuine users fall into the *'+ / +'* category for *VP-3*. Overall, we can say that *VP-3* has secured the best results from our analysis.

Table 7.3: Results obtained from statistical approach with harmful actions.

Category	SA-1				SA-2			
	# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
+ / +	30		299		12		272	
+ / -	11		1238	23	10		2429	50
- / +	10	1583	411		21	751	317	
- / -	2	8942	771	3	10	3225	1218	35
Summary	53	14096	686	26(0.9%)	53	8129	1352	85(3.1%)

Table 7.4: Results obtained from machine learning approach with harmful actions.

	Category	# User	ANGA	ANIA	# Imp. ND
VP-1	+ / +	28		255	
	+ / -	12		733	28
	- / +	11	2518	250	
	- / -	2	2382	852	5
	Summary	53	13745	584	33(1.2%)
VP-2	+ / +	30		155	
	+ / -	14		620	29
	- / +	5	2299	220	
	- / -	4	13172	578	9
	Summary	53	15656	547	38(1.4%)
VP-3	+ / +	35		180	
	+ / -	12		503	28
	- / +	4	1742	201	
	- / -	2	13134	816	8
	Summary	53	16057	499	36(1.3%)

Table 7.5: Results obtained from MLA for *VP-2* with Score Boost.

	Category	# User	ANGA	ANIA	# Imp. ND
VP-2	+ / +	32		374	
	+ / -	11		1053	21
	- / +	9	8683	757	
	- / -	1	1717	904	2
	Summary	53	15624	726	23(0.8%)

7.4.1 Harmful Action

We can understand that a *Key Digraph* action derived from two consecutive keys is considered in the analysis (see Section 5.2.5). Therefore, it is not an activity that can cause any damage on the PC, because the *Single Key Actions* are already considered. Tables 7.3 and 7.4 show the results after excluding the *Key Digraph* actions.

We also applied *Score Boost* algorithm discussed in Section 6.2.1 for *VP-2* and the obtained results shows in Table 7.5. We can see that there is not a significant improvement in the results, even though 32 participants qualify the *'+ / +'* category as compared to 30 participants in case of without score boost, but, the ANIA value is high (*i.e.* 374 as compared to 155). Due to the complexity to optimize the parameters of Algorithm 6.1, we are unable to obtain significant improvement in the results.

7.5 Summary

The summary of the major findings from this research is as follows:

- The machine learning approach performs better in terms of ANIA than the statistical approaches. Also when we consider *VP-3* the number of participants falls into the *'+ / +'* category is highest (*i.e.* 35 participants with ANIA of 269 of which 180 actions can be considered as potentially harmful).
- For the statistical approaches, no imposters user data was used for template creation. In this respect the *SA-1* provide us very good results *i.e.* 30 participants fall into the *'+ / +'* category with ANIA of 443 (of which 299 actions can be considered as potentially harmful).
- The fuzzy logic approach does not provide good results. The major challenge we had during this analysis was to optimize the membership function's parameters for the fuzzy logic controller. The process is very time consuming and extremely resource hungry. Therefore, we have used same fuzzy controller for all the participants and both actions. The performance could maybe improved by applying optimized parameters of fuzzy logic membership functions for the different participants and more precisely for the different actions.

Chapter 8

Continuous Authentication using a Combination of Keystroke and Mouse Dynamics

In this chapter we focus on context independent continuous authentication system that reacts on every separate action performed by a user. The experimental data was collected in a complete uncontrolled condition from 53 users by using our data collection software. We considered both keystroke and mouse usage behaviour patterns to prevent a situation where an attacker avoids detection by restricting to one input device because the continuous authentication system only checks the other input device. The complete description and the extracted features for different modalities are given in the Section 5.2 and the description of our data collection software can be found in Chapter A.

This chapter is based on the papers published in: [98] MONDAL, S., AND BOURS, P. Continuous authentication in a real world settings. In *8th Int. Conf. on Advances in Pattern Recognition (ICAPR'15)* (2015), IEEE, pp. 1–6; [96] MONDAL, S., AND BOURS, P. Context independent continuous authentication using behavioural biometrics. In *IEEE Int. Conf. on Identity, Security and Behavior Analysis (ISBA'15)* (2015), IEEE, pp. 1–8 and [104] MONDAL, S., AND BOURS, P. A study on continuous authentication using a combination of keystroke and mouse biometrics. Under Review in Neurocomputing, 2016.

8.1 Data Processing

In this experiment, we have used *Dataset-2* for our analysis (see Section 5.2 for more details). Data separation for the user profile creation, algorithmic parameter adjustment and system performance testing will be discussed in this section.

8.1.1 Data Separation

We separated our data in three parts for every biometric subject: (1) to build and train the system (training dataset M); (2) to adjust parameters of the algorithms used in this research (validation dataset V), and (3) to test the system performance (test dataset T). Approximately 35% of the total data was used as a training dataset for building the classifier models or template creation, 10% of the total data was used as a validation dataset for parameter adjustment of the algorithms and the remaining 55% of the data was used as a test dataset.

Let $S = [s_1, s_2, s_3, \ldots, s_n]$ be the set of data of the n biometric subjects (*i.e.* $n = 53$), and let $M = [m_1, m_2, m_3, \ldots, m_n]$, $V = [v_1, v_2, v_3, \ldots, v_n]$ and $T = [t_1, t_2, t_3, \ldots, t_n]$ represent the training, parameter adjustment, and testing dataset. Figure 8.1 shows the pictorial representation of these three sets where $m_i \approx 35\%$, $v_i \approx 10\%$ and $t_i \approx 55\%$ of the total data of the i^{th} user.

8.1.2 Verification Processes

1. Verification Process 1 (VP-1): A detailed description of this process can be found in Section 4.2.2.1. Figure 8.2 shows the data representation of *User-1* for this verification process. In this figure, we can see that m'_j is the amount of data that not been used in the training phase. A similar approach was adopted for other users also.

M	V	T
m_1	v_1	t_1
m_2	v_2	t_2
...	...	...
m_{27}	v_{27}	t_{27}
m_{28}	v_{28}	t_{28}
...	...	...
m_{53}	v_{53}	t_{53}

Figure 8.1: Pictorial representation of the followed data separation process.

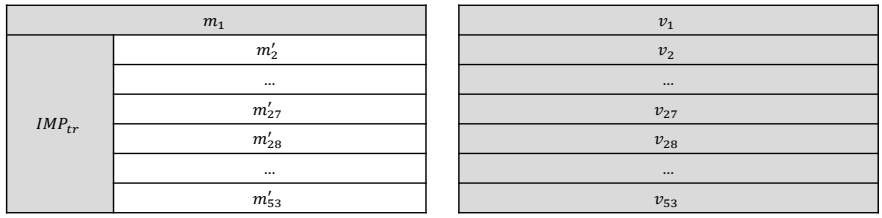


Figure 8.2: Data representation of *User-1* for *VP-1*.

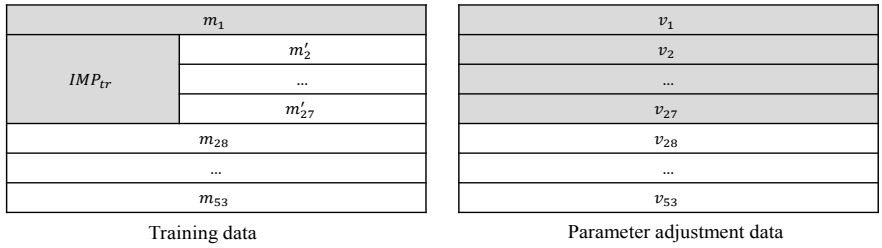


Figure 8.3: Data representation of *User-1* for *VP-2*.

2. Verification Process 2 (VP-2): A detailed description of this process can be found in Section 4.2.2.2. Figure 8.3 shows the data representation of *User-1* for this verification process. A similar approach was adopted for other users also.

3. Verification Process 3 (VP-3): A detailed description of this process can be found in Section 4.2.2.3. Figure 8.4 shows the data representation of *User-1* for this verification process. A similar approach was adopted for other users also.

4. Verification Process 4 (VP-4): Beside the three verification processes that are introduced before we now also do use a new one. In this verification process, we use no imposter data

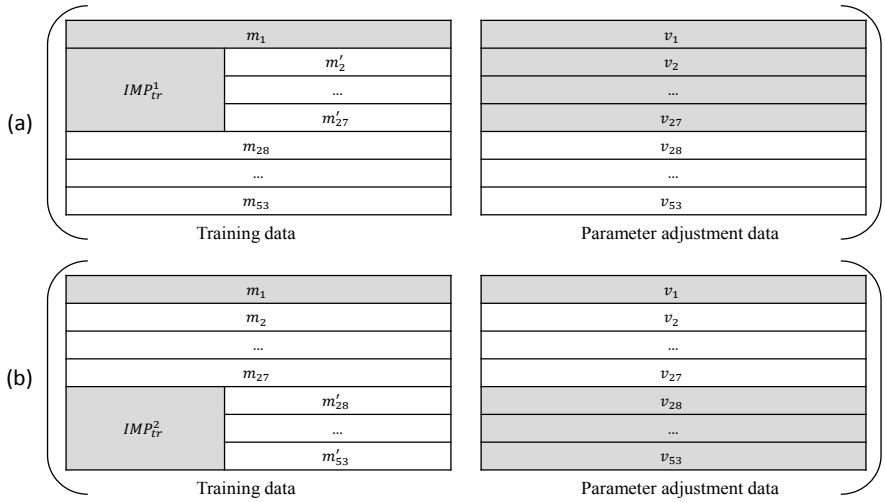


Figure 8.4: Data representation of *User-1* for *VP-3*.

during the training phase for keystroke actions and mouse single and double click actions. For mouse move and drag-drop we followed the same approach as *VP-1* and *VP-2*.

8.2 Classification Techniques

We applied separate classification techniques for different modalities along with different verification processes. The description of these techniques is given below.

8.2.1 Classification for VP-1, VP-2 and VP-3

We use three different classifier models in a *Multi Classifier Fusion (MCF)* architecture for all the different actions and for *VP-1*, *VP-2* and *VP-3*. The regression models are *Artificial Neural Network (ANN)* and *Counter-Propagation Artificial Neural Network (CPANN)* and the prediction model is *Support Vector Machine (SVM)*. The score vector we use is

$$(f_1, f_2, f_3) = (Score_{ann}, Score_{svm}, Score_{cpann}).$$

From these three classifier scores we calculate an overall score that is used in the *Trust Model* in the following way: $sc = (\sum_{j=1}^{3} w_j f_j)/(\sum_{j=1}^{3} w_j)$ where w_j are the weights for the weighted fusion technique. The weights are optimized using *Genetic Algorithm* techniques.

8.2.2 Classification for VP-4

Single Key Classification

We use pairwise *Scaled Euclidean Distance* for particular key observations for this verification process. For example, the key *a* has n observations in the training data, then we will get n scaled Euclidean distances for the new test sample of *a*. The distance metric vector we use is now (f_1, f_2, f_3) = (mean, minimum, maximum) of these n distances. From these 3 attributes we calculate a score that is used in the *Trust Model* in the following way: $sc = 1 - (f_1 - f_2)/(f_3 - f_2)$.

Key Digraph Classification

We used two distance matrices: *Scaled Euclidean Distance* (SED) and *Correlation Distance* (CD) for particular key observations. For example, the key digraph *ab* has n observations on the training data. Now, we get n Scaled Euclidean Distances and n Correlation Distances for the new test sample of digraph *ab*. Then we take the distance vector as (f_1, f_2, f_3) = (mean of SED, minimum of SED, maximum of CD). From this we calculate the score sc used in the *Trust Model* in the following way: $sc = (f_1 \times f_3)/f_2$.

Mouse Single Click

We only have duration of the click as a feature. We calculate pairwise *Scaled Euclidean Distance* (SED) from all the training observations after removing outliers by using *Interquartile Range (IQR)*. In this verification process, the training sample (x) was removed from the training set when $x < (Q_1 - 1.5 \times IQR)$ or $x > (Q_3 + 1.5 \times IQR)$ where, Q_1 and Q_3 are the first and the third quartile and $IQR = Q_3 - Q_1$. Then we construct the distance vector as (f_1, f_2, f_3) = (mean of SED, minimum of SED, standard deviation of SED). From this we calculate the score sc used in the *Trust Model* in the following way: $sc = w/(f_1 + f_3) + (1 - w) \times (1 - (f_1 - f_3)/(f_2 - f_3))$ where, w is the weight for the weighted fusion technique and optimized by using *Genetic Algorithm*. We applied the same classification technique as for *Single Key* for this action, but we got better results when applying the above technique.

Mouse Double Click

A similar classification technique was followed for mouse double click as for *Mouse Single Click*. We first applied the same classification technique as for *Single Key* for this action, but we got better results when applying the above mentioned technique.

Mouse Move and Drag-Drop

For these two actions similar classification approach was adopted for *VP-4* as other verification processes, *i.e. VP-1*, *VP-2* and *VP-3*.

8.3 Feature Selection

8.3.1 Feature Selection Method-1

Before building the classifier models, we first apply the feature selection technique [78] for Mouse Move and Drag-Drop actions. The feature subset is based on $S_n > Threshold$ where $S_n = sup\left|CDF(x_g^n) - CDF(x_i^n)\right|$, $CDF()$ is the *Cumulative Distribution Function*, x_g^n is the genuine user's n^{th} feature and x_i^n is the imposter's n^{th} feature. Figure 8.5 shows the Empirical Cumulative Distribution plot of mouse move features after feature selection for *User-12*.

8.3.2 Feature Selection Method-2

Another feature selection process we have followed is based on the maximization of the separation (*i.e. Kolmogorov –Smirnov (KS)* test [140]) between two *Multivariate Cumulative Distributions*.

Let $F = \{1, 2, 3, \ldots, m\}$ be the total feature set, where m is the number of feature attributes. The feature subset $A \subseteq F$ is based on the maximization of FS with *Genetic Algorithm* as a feature subset searching technique where, $FS = sup\left|MVCDF(x_i^A) - MVCDF(x_j^A)\right|$, $MVCDF()$ is Multivariate Cumulative Distribution Function, x_i^A is i^{th} user feature subset data and x_j^A is imposter user(s) feature subset data. Figure 8.6 shows the Empirical Cumulative Distribution plot of the selected features for *User-12*. We also tested the feature selection technique proposed by Ververidis *et al*. [148] and the obtained results can be found in the result analysis section.

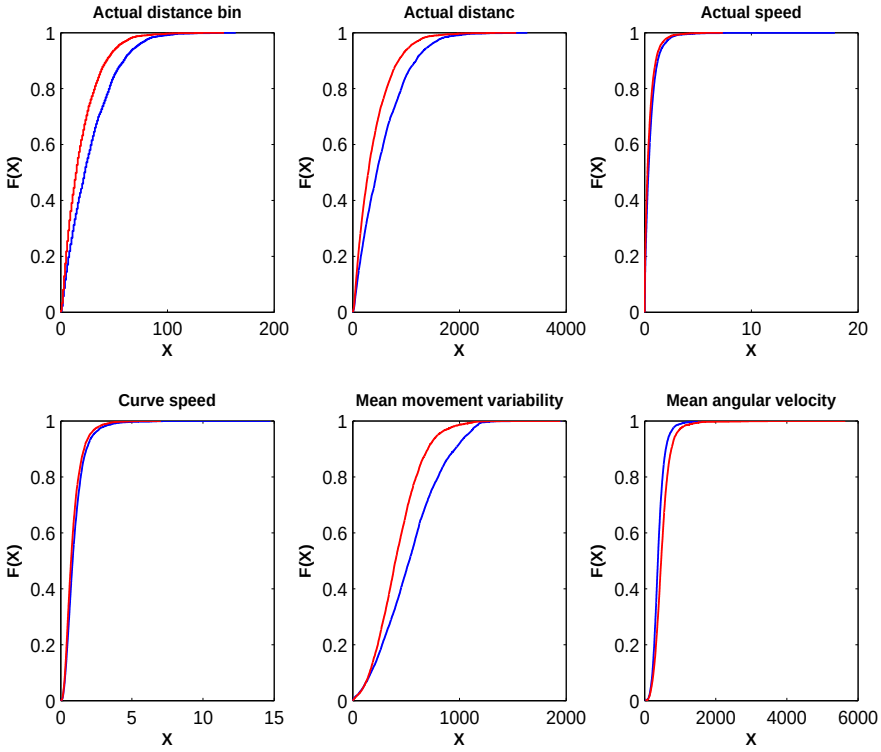


Figure 8.5: Cumulative Distribution of mouse move features after feature selection *Method-1* for *User-12*. Here, the blue lines are generated from the genuine user data (*i.e. User-12*) and the red lined are generated from the data of the imposters for this user.

8.3.3 Discussion

The major difference between two of our proposed feature selection methods (*i.e. Method-1* and *Method-2*) is that in *Method-1* individual feature contributions are taken into consideration and in *Method-2* the total feature subset contribution was taken into consideration. As we know in pattern recognition, the best separable features do not always perform the best when we combine them. If we compare the selected features after applying these two methods for the same user, we can see in Figure 8.5 at least three features have a high separable information where as in Figure 8.6 we have only one such feature. But, we find that combination of all the features selected by *Method-2* has more classifiable information than the features selected by *Method-1*.

8.4 System Architecture

Figure 8.7 shows the complete system architecture for this analysis. We check every action performed by the user and based on the way this action is performed the trust level is adjusted. We would like to mention that at any given time the user can only perform either a mouse action or a key action but not both at the same time. Therefore, we do not need any multi-modality fusion architecture for our analysis as was recently seen in [1, 10, 63, 146]. We would also like to mention that the system trust will be adjusted based on any six actions performed by the user. Figures 8.8

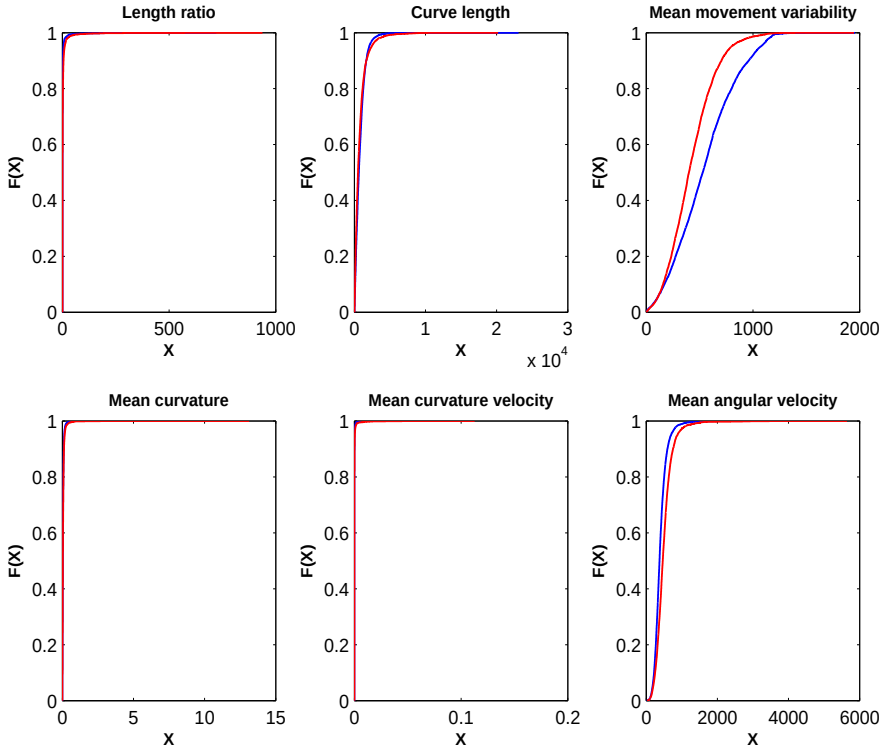


Figure 8.6: Cumulative Distribution of selected features *Method-2* for *User-12*. The blue lines are generated from the genuine user data (*i.e. User-12*) and the red lined are generated from the data of the imposters for this user.

and 8.9 show the expanded matching module for the keystroke and mouse dynamics, respectively.

For each action performed by the current user, the system calculates the score for that action (see for details in Section 8.2) and uses that score to calculate the change in trust according to Equation 3.2 of Algorithm 3.3. Parameters A, B, C, and D in Equation 3.2 depend on the type of action and are optimized by *Genetic Algorithm (GA)* optimization technique [115] with the parameter adjustment dataset, where the cost function is to maximize $ANGA - ANIA$.

8.5 Result Analysis

In this section, we analyse the results that we obtained by applying the analysis methods discussed above. We performed zero effort imposter attack, therefore, we followed leave-one-out testing where we have the test data of 1 genuine user and 52 imposter users. Table 8.1 shows number of actions on an average we have tested for each users to measure our system performance.

The total number of data sets of genuine users is 53 and the total number of data sets of imposter users is 53 $\times$ 52 = 2756. We report the results in terms of ANIA and ANGA along with the total number of imposters not detected, based on a user dependent lockout threshold ($T_{lockout}$) which was optimized by using GA.

Tables 8.2 and 8.3 show the results we got from our analysis for *VP-1* and *VP-2* respectively, with the proposed feature selection *Method-1*. We can see that the results for *VP-2* has been improved

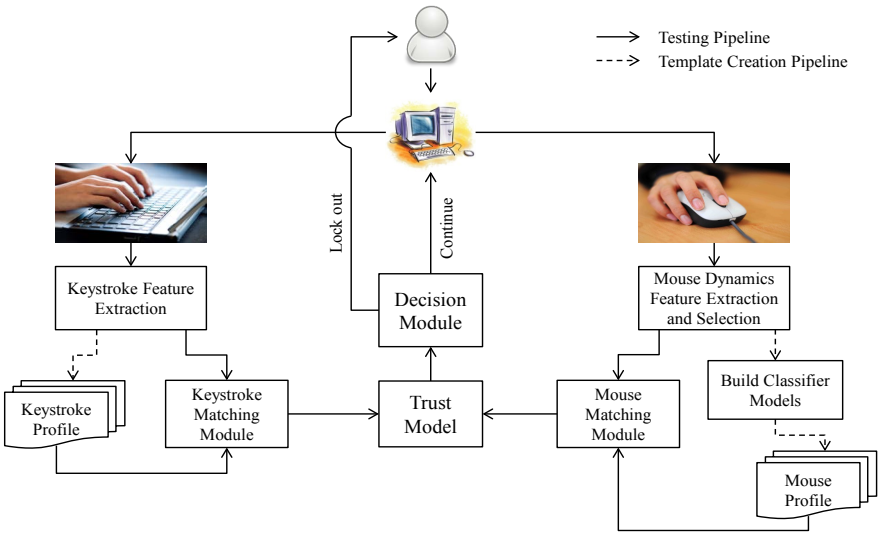


Figure 8.7: Block diagram of the proposed system.

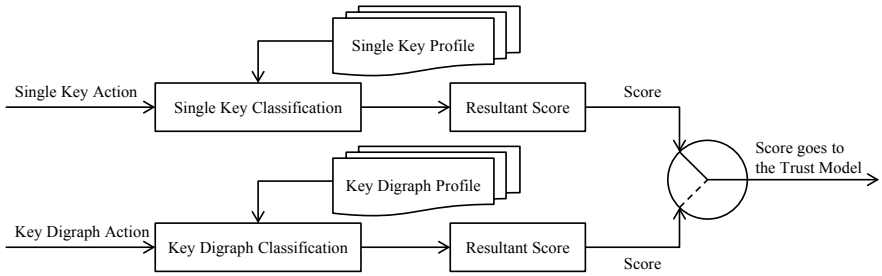


Figure 8.8: Block diagram of the *Keystroke Matching Module*.

Table 8.1: Average number of actions tested for each users.

Action Type	Genuine Actions	Imposter Actions
Single Key Action	17.4×10^3	90.4×10^4
Key Digraph Action	8.8×10^3	45.5×10^4
Mouse Single Click Action	6×10^3	31.1×10^4
Mouse Double Click Action	1.8×10^3	9.2×10^4
Mouse Move Action	8.2×10^3	42.5×10^4
Mouse Drag-Drop Action	0.1×10^3	0.4×10^4
Total	42.3×10^3	219.1×10^4

when compared to *VP-1 i.e.* number of participants qualify '+ / +' category increased from 39 to 44 and also the value of ANIA reduced from 439 to 297. Now, we have applied feature selection technique *Method-2* for *VP-2*. The results obtained from this analysis shows in Table 8.4, and we can observe that the results did not improve. We performed the same analysis with *Penalty-Reward*

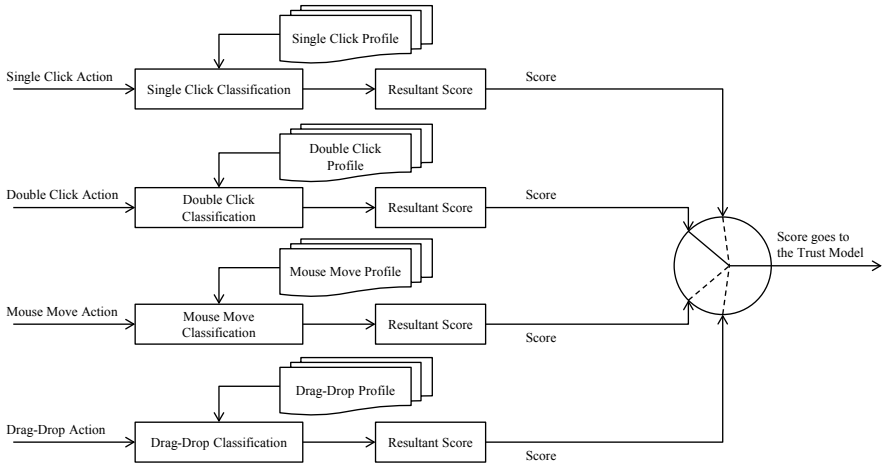


Figure 8.9: Block diagram of the *Mouse Matching Module*.

Table 8.2: Results obtained for *VP-1* with feature selection *Method-1*.

Category	# User	ANGA	ANIA	# Imp. ND
+ / +	39		439	
+ / -				
- / +	14	9716	517	
- / -				
Summary	53	33693	460	0(0%)

Table 8.3: Results obtained for *VP-2* with feature selection *Method-1*.

Category	# User	ANGA	ANIA	# Imp. ND
+ / +	44		297	
+ / -	4		1077	4
- / +	5	4062	558	
- / -				
Summary	53	38693	440	4(0.2%)

Fusion (PRF)[1] and observed the improvement on the results in Table 8.5 *i.e.* number of participants qualify for the *'+ / +'* category increased from 44 to 46.

Table 8.6 shows the results we got from our analysis for *VP-3* with proposed feature selection *Method-2*. In this table we show the results obtained from both *Score Fusion (SF)* as well as *Penalty-Reward Fusion (PRF)* techniques. We can observe from this table that the SF performs better than PRF *i.e.* number of participants that qualify for *'+ / +'* category increased from 44 to 47 and also ANIA value for that category was reduced slightly from 288 to 275.

Table 8.7 shows the results we got from our analysis for *VP-4* with feature selection technique proposed by Ververidis *et al*. [148] and the number of imposters used for training the classifiers (*i.e.* for *Mouse Move Action* and *Mouse Drag-Drop Action*) and for parameter adjustment is $j = n - 1$,

[1]In this fusion technique, from the 3 classifier scores we have individually calculate the penalty-reward from Equation 3.1. So, $\Delta_T(sc^1)$ represents the penalty-reward from Classifier-1, $\Delta_T(sc^2)$ stands for the penalty-reward from Classifier-2 and $\Delta_T(sc^3)$ is the penalty-reward from Classifier-3. Then we combine these with weighted fusion to calculate the system trust from Equation 3.2. Therefore, $\Delta_T(sc) = (\sum_{j=1}^{3} w_j \Delta_T(sc^j))/(\sum_{j=1}^{3} w_j)$ were, w_j are the weights for the weighted fusion techniques. The weights are optimized using *Genetic Algorithm* with the parameter adjustment dataset.

Table 8.4: Results obtained for *VP-2* with feature selection *Method-2*.

Category	# User	ANGA	ANIA	# Imp. ND
+ / +	44		374	
+ / -	3		487	3
- / +	6	5322	542	
- / -				
Summary	53	38114	445	3(0.1%)

Table 8.5: Results obtained for *VP-2* with feature selection *Method-2* and Penalty-Reward Fusion.

Category	# User	ANGA	ANIA	# Imp. ND
+ / +	46		381	
+ / -	2		1241	3
- / +	5	9920	642	
- / -				
Summary	53	39245	483	3(0.1%)

Table 8.6: Results obtained for *VP-3* with feature selection *Method-2*.

Category	Score Fusion (SF)				Penalty-Reward Fusion (PRF)			
	# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
+ / +	47		275		44		288	
+ / -	3		1580	3	5		2361	16
- / +	3	4037	726		4	8876	405	
- / -								
Summary	53	40134	419	3(0.1%)	53	39777	723	16(0.6%)

Table 8.7: Results obtained for *VP-4* with feature selection proposed by [148] and $j = n - 1$.

Category	# User	ANGA	ANIA	# Imp. ND
+ / +	17		634	
+ / -	5		2742	10
- / +	30	5854	585	
- / -	1	8337	5974	1
Summary	53	21029	10619	11(0.4%)

Table 8.8: Results obtained for *VP-4* with feature selection *Method-1* and $j = n - 1$.

Category	# User	ANGA	ANIA	# Imp. ND
+ / +	25		548	
+ / -	1		2684	1
- / +	27	6378	462	
- / -				
Summary	53	24000	559	1(0.04%)

i.e. $j = 52$. We can observe from the table that only 17 participants qualify for the '+ / +' category, where the average ANIA is 634 actions. Only five participants qualify the '+ / -' category where ANIA is relatively high (*i.e.* 2742 actions) and in total 10 imposters were not detected. A total of 30 participants qualify for the '- / +' category where ANGA and ANIA are 5854 and 585 actions respectively. Only one participants falls into the '- / -' category where the ANGA and ANIA are 8337 and 5974 actions respectively, and also one imposter is not detected.

Table 8.8 shows the results we got from our analysis for *VP-4* with proposed feature selection *Method-1* and $j = n - 1$, *i.e.* $j = 52$. We can clearly observe the improvement on the results from this table *i.e.* 25 participants qualify for the '+ / +' category, where the average ANIA is 548 actions.

Table 8.9: Results obtained for *VP-4* with feature selection *Method-1* and $j = \lfloor \frac{n-1}{2} \rfloor$.

Category	# User	ANGA	ANIA	# Imp. ND
+ / +	36		435	
+ / -	3		2918	5
- / +	13	8389	750	
- / -	1	14075	901	2
Summary	53	33450	763	7(0.3%)

Only one participants qualify for the *'+ / -'* category where ANIA is 2684 actions and one imposter was not detected and total 27 participants qualify for the *'- / +'* category where ANGA and ANIA are 6378 and 462 actions respectively. No participants fall into the *'- / -'* category.

Table 8.9 shows the result we got from our analysis for *VP-4* with proposed feature selection *Method-1* and $j = \lfloor \frac{n-1}{2} \rfloor$ *i.e.* $j = 26$. We can again observe the improvement on the results from this table when compare to Table 8.8, *i.e.* 36 participants qualify for the *'+ / +'* category, where the average ANIA is 435 actions. Only three participants qualify for the *'+ / -'* category where ANIA is 2918 actions and total five imposters were not detected and 13 participants qualify for the *'- / +'* category where ANGA and ANIA are 8389 and 750 actions respectively. Only one participant falls into the *'- / -'* category where the ANGA and ANIA are 14075 and 901 actions respectively also, two imposters are not detected.

From the above experimental results we observe that *VP-4* has a lesser performance when compared to *VP-1* and *VP-2*. This is understandable because no imposter users were considered to train the system and parameter adjustment process for four significant actions. But surprisingly, we got a better performance for *VP-2* and *VP-3* compared to *VP-1*. In *VP-1* all the imposter users are considered in the training of the system and in the parameter adjustment process for all actions, while for *VP-2* only 50% imposter users are considered and for *VP-3* the training and testing was done by two mutually exclusive sets of imposter users. Due to the varied nature of the behavioural biometric paradigm, we can assume that the increment of the number of imposters on the training phase can also affect the system performance. We can validate this assumption by looking at the classifier's training accuracy where we have noted that the classifier's training accuracy was improved for *VP-2* when compared to *VP-1*. Similar findings are reported in the research conducted by Pusara [121].

We have studied the data of the genuine users in the *'- / +'* category participants in some more detail. For one of those genuine users we noted that the trust level when tested against his/her own model dropped significantly for a period of time. See Figure 8.10 for a graphical impression of the trust level of one of these user. This figure was generated without applying any lockout threshold for better understanding. We can see from the figure that the user performed very well up to approximately action number 1.9×10^4, when a sudden drop in the trust level appears. The trust level remains low until approximately action number 2.1×10^4, after which the trust returns to a high level. Although we have no proof of this, it seems valid that the computer of this particular user was used by somebody else during this period, *i.e.* the owner of the computer allowed another person to use his/her PC, *e.g.* to play a game, search the web, or alike.

8.6 Discussion

Our major focus in this chapter was to develop a proper CA system, which reacts on every single action performed by the user. A change of the pre-processing can influence the results (*e.g.* a different feature extraction process, a different feature selection technique and a different choice of classifiers). In this section we discuss some general concerns related to this research and provide some recommendations related to the usefulness of our research for the future aspects.

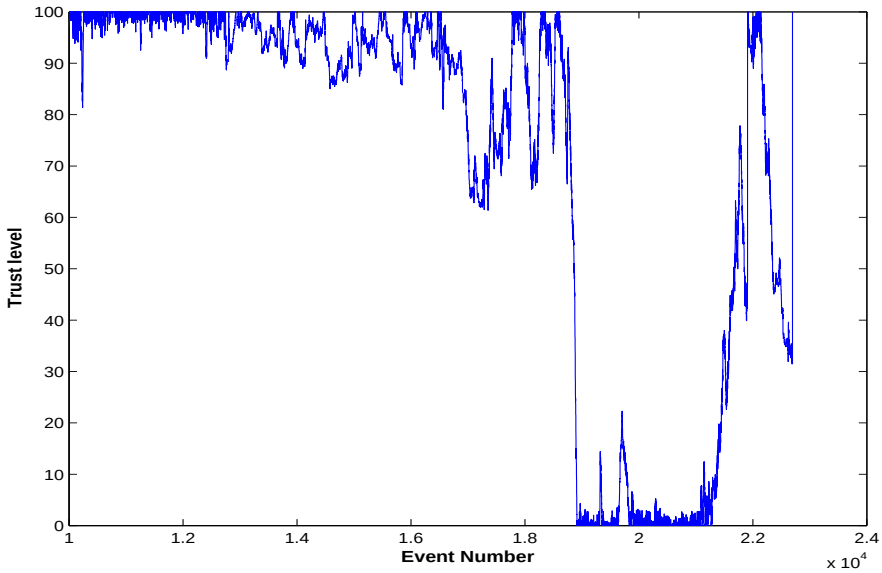


Figure 8.10: Change of the system trust for the one of the genuine user from *'- / +'* category for *VP-4*.

Table 8.10: Results obtained for *VP-2* by using only KD and MD.

Category	Keystroke Dynamics				Mouse Dynamics			
	# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
+ / +	30		235		36		186	
+ / -	14		950	29	11		929	27
- / +	5	3381	331		4	5001	338	
- / -	4	20185	875	9	2	1097	570	3
Summary	53	23593	840	38(1.4%)	53	14696	532	30(1.1%)

8.6.1 Significance of a combination of KD and MD

In our research, we considered keystroke and mouse dynamics both, not only to improve the system performance due to fusion, but also to prevent the situation where an attacker avoids detection by restricting to one input device because the system only checks the other input device. Table 8.10 shows the performance results if we would have considered only keystroke or only mouse actions. In both cases we performed the analysis only for *VP-2* with feature selection *Method-1* and SF. We can also see the improvement in the results when a combination of keystroke and mouse dynamics is used by comparing Tables 8.10 and 8.3.

8.6.2 Significance of the Score Boost algorithm

We applied the *Score Boost* algorithm discussed in Section 6.2.1 for *VP-1* and *VP-2* with SF. The obtained results from this analysis are shown in Table 8.11. We can see that there is an improvement in terms of ANIA in *'+ / +'* category users for *VP-1 i.e.* ANIA is reduced from 439 to 309 when we compare this result with Table 8.2, but the overall performance is lower. We also see a lower performance for *VP-2* when comparing this result with Table 8.3. Due to the complexity to optimize the parameters of Algorithm 6.1, we are unable to obtain significant improvement in the results.

Table 8.11: Results obtained for *VP-1* and *VP-2* by using *Score Boost* and SF.

Category	*VP-1*				*VP-2*			
	# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
+ / +	39		309		40		218	
+ / -					3		1852	3
- / +	14	5640	484		10	20067	513	
- / -								
Summary	53	32616	355	0(0%)	53	38105	410	3(0.1%)

Table 8.12: Comparison with previous research.

Ref.	# Users	FNMR	FMR	Block size	ANGA	ANIA
[1]	10	3%	3%	?	?	?
[10]	31	2.24%	2.10%	1100	49107	1124
[63]	20	17.80%	17.80%	?	?	?
[146]	24	8.21%	8.21%	40	487	44

8.6.3 Comparison with previous research

We would also like to test and compare our obtained results on another dataset used in the state of the art research [1, 10, 63, 146] but unavailability of these datasets limits the comparison with these previous works. In Table 8.12 we show the results results from previous research that combined keystroke and mouse dynamics for continuous authentication. The reported results are converted to ANIA and ANGA (when the block size was given) according to the conversion method given in Section 4.3.1.

Table 8.13 shows the comparison between our CA approach with state of the art *Periodic Authentication (PA)* approach with the same preprocessing and classification technique, where the block size varies from 100 to 1000. In this table, *False Non-Lockout Rate (FNLR)*[2] means the percentage of a block of actions from imposter user is not locked-out by the system and *False Lockout Rate (FLR)*[3] means the percentage of a block of actions from the genuine user is wrongly locked-out of the system. We can see that our system performs better when comparing the ANGA values *i.e.* the CA system's ANGA value is much higher than the comparable PA system's ANGA value (marked as bold), when the ANIA values are approximately the same. We can do a similar comparison when ANGA values are approximately the same for both CA and PA system. Then we can see that the CA system's ANIA values are much lower than the PA system's ANIA value. Due to unavailability of the comparable ANGA values for *VP-2* and *VP-3*, we marked these in as bold for only *VP-1*.

8.6.4 Significance of our dataset

We found that all the state of the art experiments that combine keystroke and mouse dynamics, were conducted in a controlled lab environments. Participants had to complete a predefined task or use a set of predefined applications [1, 10, 63, 146]. Based on our understanding these experiments are the starting point to understand the science behind continuous authentication, but not enough to extend it to a real world situation. Therefore, we built a dataset which represents the real world continuous authentication data for our experiments.

Our dataset consist of both keystroke and mouse information and includes user application information. The user application information could be useful to improve the system performance.

[2]This terminology can be compared to FMR.
[3]This terminology can be compared to FNMR.

Table 8.13: Comparison between PA and our CA.

Approach	Block Size	*VP-1*				*VP-2*				*VP-3*			
		FNLR	FLR	ANIA	ANGA	FNLR	FLR	ANIA	ANGA	FNLR	FLR	ANIA	ANGA
PA	100	14.41	6.49	117	1541	17.03	15.88	121	630	20.67	20.21	126	495
	150	13.66	5.44	174	2759	16.45	15.54	180	965	20.41	20.09	188	747
	200	13.04	4.74	230	4216	16.36	14.84	239	1348	20.17	19.67	251	1017
	250	12.47	4.39	286	5698	16.16	14.36	298	1741	20.41	19.10	314	1309
	300	12.15	4.45	341	6739	15.74	14.57	356	2059	**19.62**	**19.82**	**373**	**1514**
	350	11.69	4.27	396	8192	15.61	14.06	415	2489	**20.10**	**19.08**	**438**	**1834**
	400	**11.58**	**4.03**	**452**	**9919**	**15.51**	**13.82**	**473**	**2895**	19.50	19.16	497	2088
	450	**11.18**	**3.77**	**507**	**11943**	**15.30**	**13.59**	**531**	**3311**	19.75	18.90	561	2381
	500	11.03	3.62	562	13796	15.36	13.10	591	3817	19.49	18.52	621	2700
	550	10.87	3.51	617	15691	15.23	13.41	649	4103	19.74	18.63	685	2952
	600	10.92	3.33	674	18016	15.27	12.50	708	4800	19.27	18.55	743	3234
	650	10.67	3.33	728	19493	14.88	12.62	764	5152	19.19	18.40	804	3532
	700	10.71	3.08	784	22737	15.02	12.25	824	5715	19.09	18.37	865	3811
	750	10.48	3.15	838	23806	14.77	12.51	880	5998	18.92	18.17	925	4128
	800	10.18	2.62	891	30566	14.83	12.19	939	6562	19.48	17.86	994	4480
	850	10.61	2.37	951	35883	14.65	12.61	996	6742	18.92	18.34	1048	4634
	900	10.21	2.93	1002	30736	15.00	12.01	1059	7492	18.93	18.36	1110	4901
	950	**9.97**	**3.03**	**1055**	**31304**	14.50	12.02	1111	7905	18.69	18.20	1168	5221
	1000	**9.94**	**2.90**	**1110**	**34462**	14.47	12.42	1169	8051	18.72	17.94	1230	5574
Our CA				**460**	**33693**			**483**	**39245**			**419**	**40134**

Table 8.14: Results obtained for *VP-2* without MCF.

Category	ANN				SVM				CPANN			
	# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
+ / +	29		534		28		357		30		608	
+ / -	3		697	3	6		1800	10	6		1342	12
- / +	16	5989	704		17	6912	513		14	8333	664	
- / -	5	5730	2029	11	2	9764	952	2	3	38626	1384	3
Summary	53	27888	941	14(0.5%)	53	29721	769	12(0.4%)	53	33120	972	15(0.5%)

Table 8.15: Results obtained for *VP-3* without MCF.

Category	ANN				SVM				CPANN			
	# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
+ / +	36		377		36		386		34		379	
+ / -	9		1542	18	7		2013	13	9		1388	13
- / +	8	6287	639		8	8457	680		9	7858	590	
- / -					2	4517	883	2	1	5072	1938	1
Summary	53	36864	879	18(0.7%)	53	35766	883	15(0.5%)	53	35749	822	14(0.5%)

8.6.5 Significance of Multi-Classifier Fusion (MCF)

We applied our proposed scheme by using a single classifier for *VP-2* and *VP-3*. Table 8.14 shows the results we found from this analysis for *VP-2* with feature selection *Method-1* and SF. When we compare the results in Table 8.3 with the results in Table 8.14, we clearly see that the MCF performs better than each of the single classifiers.

Table 8.15 shows the results we found from this analysis for *VP-3* with feature selection *Method-2* and SF. When comparing the results in Table 8.6 with the results in Table 8.15, we also see that for *VP-3* the MFC performs better than each of the single classifiers.

8.6.6 Potential attack scenario

An attacker might try to take advantage of the fact that the threshold for considering *Digraph Key Actions* is 2000ms. The way he/she could do so is by waiting at least 2 seconds between consecutive key presses and performing no mouse action. This scenario may weaken the CA system, but it cannot bypass it. In the mentioned attack scenario, the system will now only consider *Single Key Actions* features for continuous authentication. The major disadvantage for the attacker in such a scenario is that he/she will have to be patient because the attack takes much longer time, *i.e.* maximum 30 typed characters per minute, which makes this attack less likely in the event that an attacker quickly

Table 8.16: Results obtained for *VP-2* and *VP-3* with STM.

Category	*VP-2*				*VP-3*			
	# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
+ / +	29		282		35		272	
+ / -	3		3518	7	6		1423	11
- / +	19	6593	349		11	4091	368	
- / -	2	25203	1083	2	1	23739	1113	1
Summary	53	28854	647	9(0.3%)	53	34020	615	12(0.4%)

Table 8.17: Results obtained for *VP-2* with harmful actions.

Category	# User	Combined		Keystroke		Mouse		# Imp. ND
		ANGA	ANIA	ANGA	ANIA	ANGA	ANIA	
+ / +	46		244		167		78	
+ / -	2		754		499		255	3
- / +	5	5624	381	3168	260	2455	121	
- / -								
Summary	53	23444	303					3(0.1%)

uses the fact that the genuine user has left his computer unattended for a short period of time.

In future research we will investigate how much the typing behaviour will change when typing single keys with large time intervals between them and determine in detail how well such an attack will work.

8.6.7 Experiments with Static Trust Model

We have also performed experiments with *Static Trust Model (STM)* (see Section 3.2 and Algorithm 3.1) for *VP-2* and *VP-3* where the parameters of STM are optimized using GA. Table 8.16 shows the results we obtained from this analysis. We can clearly see that the *Dynamic Trust Model* performs better than the *Static Trust Model* for both the verification processes.

8.6.8 Harmful Action

The solutions provided in the result analysis section (*i.e.* in Section 8.5) are based on the actions discussed in the Section 8.2. We can understand that a simple *Mouse Move* action does not harm the PC until it followed by a *Mouse Single Click* or a *Mouse Double Click* action. Similarly, a *Key Digraph* action derived from two consecutive keys, pressed within 2000ms, is considered in the analysis, but by itself is it not an activity that can cause any damage on the PC. Table 8.17 shows the obtained results from our analysis for *VP-2* with feature selection *Method-2* and PRF *i.e.* same analysis followed for Table 8.5. Table 8.18 shows the obtained results from our analysis for *VP-3* with feature selection *Method-2* and SF *i.e.* same analysis followed for Table 8.6 and SF, when excluding these two actions in the reporting. In these tables we report not only on the combination of keystroke and mouse dynamics, but also split the results into the number of actions for each of these modalities separately.

8.6.9 Challenges of this research

There are several challenges we faced during our research. Among these are the following ones most significant:

- Creating a context independent template from unstructured and absolutely non-predefined data. The multi classifier approach and our proposed trust model (see Equation 3.1 of Algorithm 3.3) have handled the situation to a high degree of satisfaction.

Table 8.18: Results obtained for *VP-3* with harmful actions.

Category	# User	Combined		Keystroke		Mouse		# Imp. ND
		ANGA	ANIA	ANGA	ANIA	ANGA	ANIA	
+ / +	47		167		111		57	
+ / -	3		944		640		305	3
- / +	3	2265	427	1369	285	896	143	
- / -								
Summary	53	23996	252					3(0.1%)

- Finding appropriate analysis techniques when users have missing actions in their template. In our research, we have used a fixed *Penalty* in the *VP-4* verification process for missing actions, which allows the CA system to detect an imposter faster due to fact that unexpected behaviour was observed. In case of the *VP-1*, *VP-2* and *VP-3* verification processes, the missing actions were predicted based on the user's global actions profile.
- The number of participants in our experiment is higher than in comparable research [1, 10, 63, 146], but still relatively low. Due to privacy concerns is it very hard to find volunteers to participate in a completely uncontrolled experiment where all keystroke and mouse activity is captured.

8.7 Summary

The summary of the major findings from this research is as follows:

- The dataset used for this research was collected in such a way as to replicate the real world scenario. The results we obtained from this research are promising in comparison to other state of the art research.
- We compared our proposed CA approach with other state of the art approaches (*i.e.* PA approaches) and found superiority of our approach, both in terms of user friendliness (*i.e.* higher ANGA values) and security (*i.e.* lower ANIA values).
- In this research, we tried not to use any imposter data for four different actions during the time of user's profile creation *i.e. VP-4* and found some promising results.
- We have explored *Penalty-Reward Fusion (PRF)* in this research and achieved some improvement on the results for *VP-2*.
- We developed two feature selection techniques that could be useful for other pattern recognition problems.
- We believe that the completely uncontrolled settings might be the main reason for failing to achieve the desired result, *i.e.* all 53 users in the *'+ / + '* category.

Chapter 9

Continuous Authentication on Mobile Devices

In this chapter, we investigated the performance of a continuous biometric authentication system for mobile devices under various different analysis techniques. We tested these on two publicly available swipe gestures database with 71 and 41 users respectively, but the techniques can also be applied to other mobile biometric modalities in a continuous setting.

This chapter is based on the papers published in: [100] MONDAL, S., AND BOURS, P. Swipe gesture based continuous authentication for mobile devices. In *Int. Conf. on Biometrics (ICB'15)* (2015), IEEE, pp. 458–465 and [99] MONDAL, S., AND BOURS, P. Does context matter for the performance of continuous authentication biometric systems? an empirical study on mobile devices. In *Int. Conf. of the Biometrics Special Interest Group (BIOSIG'15)* (2015), IEEE, pp. 1–5.

9.1 Introduction

In this chapter, we will introduce a CA biometric system, which checks the genuineness of the user during the full session. This system uses the behaviour of the user to determine the trust the system has in the genuineness of that user. In particular, it will focus on input of the user's swipe gestures. The behaviour of the current user will be compared with the stored information about the behaviour of the genuine user and as a result of that comparison the trust of the system in the genuineness of the user will increase or decrease and access to the device will be blocked if the level of trust in the genuineness of the user is too low. We also address a fundamental question whether context really matters for the performance of a biometric CA system. We look at how much the performance changes if the user's behaviour profile is created by using a particular task, performed in a specific application on the mobile device compared to test data coming from various applications with different tasks.

9.2 Datasets Description and Classification

We have used two separate publicly available swipe gesture based datasets for our analysis *i.e. Dataset-3* and *Dataset-4*. The complete description of these datasets are given in Section 5.3.

In the *Dataset-4*, the data was collected in 7 different tasks, *i.e.* 4 different Wikipedia Reading articles and 3 different Image Comparison Games. We will use this information for a context dependent experiment. We noticed in the data that tasks 1 to 4 were completed by all 41 users *i.e.* 3 different reading articles and one image comparison game. The task 5 (image comparison game) was completed by 40 users, but tasks 6 and 7 (reading article and image comparison) were completed by only 14 of the original 41 users.

9.2.1 Classification

Figure 9.1 shows the complete system architecture for this research. In this figure the training phase is marked by dotted arrows and the testing phase by solid arrows.

During the training phase, we found that two regression models performed best for *Dataset-3*. We applied ANN and CPANN in a MCF architecture in our analysis. The score vector we use is $(f^1, f^2) = (Score_{ann}, Score_{cpann})$ for this dataset to do further analysis. From these two classifier scores we calculate a score that will be used in the *Trust Model* (see Section 3.3) in the following

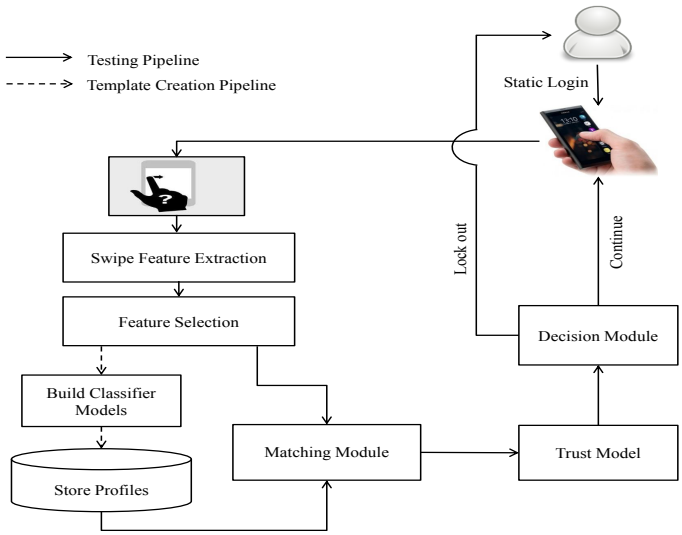


Figure 9.1: Block diagram of the proposed CA system for mobile devices.

way: For weighted fusion, $sc = w_1 \times f^1 + (1 - w_1) \times f^2$ where, w_1 is the weight for the weighted fusion technique.

We have also used two classifiers in an MFC architecture to achieve an acceptable system performance for *Dataset-4*. Due to the nature of the data we found that one prediction model and one regression model gave a better learning accuracy. We used SVM as a prediction model and CPANN as a regression model. We also tried ANN as a regression model but due to the lower learning accuracy we decided not to use it. The score vector we use for further analysis is $(f^1, f^2) = (Score_{svm}, Score_{cpann})$ for this dataset.

Before building the classifier models, we first apply the feature selection technique described in Section 8.3.2. Figure 9.2 shows the *Empirical Cumulative Distribution* plot of the selected features for *User-8* of *Dataset-3* after applying the feature selection technique described in Section 8.3.2. We also tested the feature selection technique proposed by Ververidis *et al*. [148] for both the datasets. We found that our proposed feature selection technique (see Section 8.3.2) works better for *Dataset-3* and the technique proposed by Ververidis *et al*. works better for *Dataset-4*. Therefore, we applied the feature selection technique according to this observation.

9.3 Result Analysis

In this section, we analyse the results that we obtained from our performance analysis. We divide our analysis into three major parts based on the verification process (see Section 4.2.2). We report the results from the zero-effort attack test in terms of ANIA and ANGA along with the total number of imposters not detected for fixed lockout threshold ($T_{lockout} = 90$). Also, we report the results with the user specific lockout threshold (T_{us}) where, the threshold for lockout will be $T_{us} = \max(50, \min(Trust_{genuine}))$. All the parameters of Algorithm 3.3 are optimized by linear search.

As mentioned before, in *Dataset-3* the total number of genuine users is 71 and hence, the total number of imposter users is 4970 (*i.e.* 71×70). In the *Dataset-4* the total number of genuine users is 41 for tasks 1 to 4 and hence, the total number of imposter users for these tasks is 1640 (*i.e.* 41×40). For task 5 these numbers are 40 and 1560 (*i.e.* 40×39) respectively, while for tasks 6 and 7

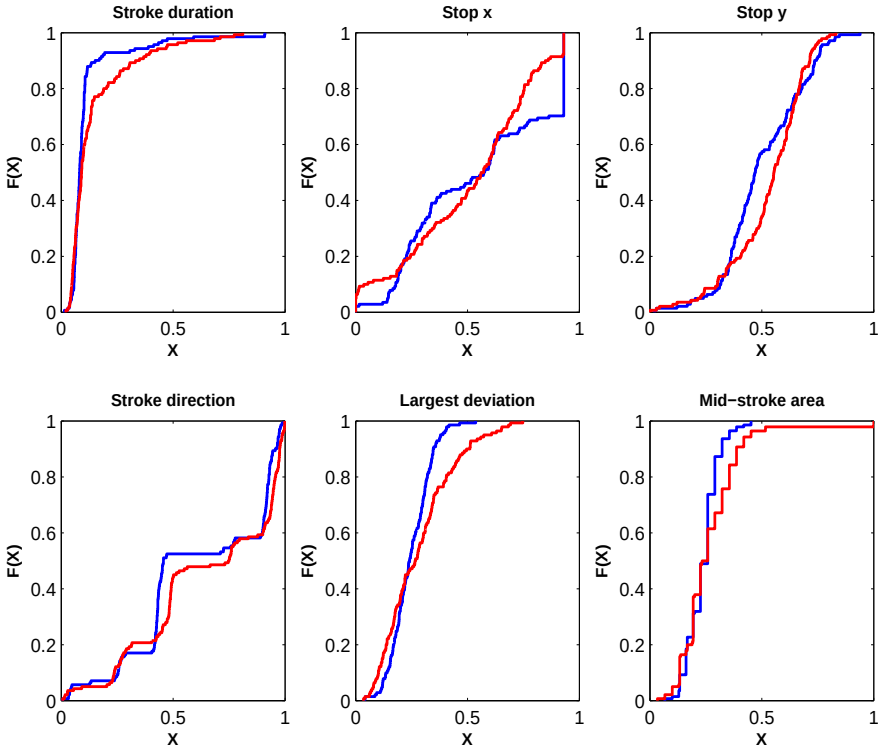


Figure 9.2: *Empirical Cumulative Distribution (ECD)* of selected features for *User-8* of *Dataset-3*. The blue ECD are generated from the genuine user data (*i.e. User-8*) and the red ECD are generated from the data of the imposters for this user.

these numbers are 14 and 182 (*i.e.* 14×13) respectively.

9.3.1 Results on Dataset-3

Table 9.1 shows the optimal result we got from this analysis for *VP-1*. The table is divided into two parts, based on the dataset used (*i.e.* `Set-1` and `Set-2`). For `Set-1` In total 68 users qualify for the best category for the user specific lockout threshold; whereas for the fixed threshold, this number drops to 63. We can observe from the table that if we go from a user specific lockout threshold (T_{us}) to fixed lockout threshold ($T_{lockout} = 90$) the results are getting worse. For the fixed lockout threshold, the number of best performing users are decreasing, and the numbers for '`+ / -`' are increasing. We also see that for `Set-2` all the 51 users satisfied the best case category for the user specific lockout threshold. Note again that `Set-2` only contains horizontal swipes, so other interactions are not used.

Table 9.2 displays the optimal result from the analysis for *VP-2*. We can clearly observe that all the 51 users satisfied the best case category for the user specific lockout threshold scheme for `Set-2`, but the ANIA increases from 5 for *VP-1* to 6 for *VP-2*. Finally, Table 9.3 shows the optimal result we got from this analysis for *VP-3*.

We can see from all these tables (*i.e.* Tables 9.1, 9.2, and 9.3) that the optimized results can be obtained from the user's specific lockout threshold (*i.e.* T_{us}). Therefore, we have performed our

Table 9.1: Results for *VP-1* with the analysis method on *Dataset-3*.

$T_{lockout}$	Category	Set-1				Set-2			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	68		4		51		5	
	+ / -	3		14	4				
	- / +								
	- / -								
	Summary	71		5	4(0.1%)	51		5	0(0%)
90	+ / +	63		14		47		19	
	+ / -	8		25	20	4		37	5
	- / +								
	- / -								
	Summary	71		16	20(0.4%)	51		21	5(0.2%)

Table 9.2: Results for *VP-2* with the analysis method on *Dataset-3*.

$T_{lockout}$	Category	Set-1				Set-2			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	66		4		51		6	
	+ / -	5		15	10				
	- / +								
	- / -								
	Summary	71		5	10(0.2%)	51		6	0(0%)
90	+ / +	56		15		48		19	
	+ / -	15		22	27	3		36	3
	- / +								
	- / -								
	Summary	71		17	27(0.5%)	51		20	3(0.1%)

Table 9.3: Results for *VP-3* with the analysis method on *Dataset-3*.

$T_{lockout}$	Category	Set-1				Set-2			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
T_{us}	+ / +	68		4		50		5	
	+ / -	3		10	3	1		27	1
	- / +								
	- / -								
	Summary	71		4	3(0.1%)	51		5	1(0.04%)
90	+ / +	57		14		49		17	
	+ / -	14		19	22	2		40	4
	- / +								
	- / -								
	Summary	71		16	22(0.4%)	51		18	4(0.2%)

next analysis only by using user's specific lockout thresholds.

9.3.2 Results on Dataset-4

We first considered the CA system performance irrespective of the context in this dataset. In this analysis, we have applied both the fusion techniques *i.e.* SF and PRF.

Table 9.4 shows the result we have obtained from this analysis with T_{us} as the lockout threshold. We clearly observe from the table that SF performs better than PRF for each of the verification processes. For this reason we only used SF for our further analysis.

Table 9.4: Results obtained for the context independent evaluation on *Dataset-4*.

	Category	SF				PRF			
		# User	ANGA	ANIA	# Imp. ND	# User	ANGA	ANIA	# Imp. ND
VP-1	+ / +	41		11		40		13	
	+ / -					1		435	14
	- / +								
	- / -								
	Summary	41		11	0(0%)	41		24	14(0.9%)
VP-2	+ / +	40		17		38		23	
	+ / -	1		88	1	3		275	39
	- / +								
	- / -								
	Summary	41		19	1(0.1%)	41		47	39(2.4%)
VP-3	+ / +	40		22		38		19	
	+ / -	1		286	1	3		233	4
	- / +								
	- / -								
	Summary	41		29	1(0.1%)	41		35	4(0.2%)

Table 9.5: Comparison with previous research for *Dataset-4*.

Reference	# Users	FNMR	FMR	Block size	ANGA	ANIA	*P-value*
[46]	41	3%	3%	12	400	12	
[128] - Horizontal	41	1.75%	1.75%	11	629	11	
[128] - Vertical	41	2.8%	2.8%	11	393	11	
Our (*VP-1* with SF)	41	0%	0%	NA	∞	11($\pm$9)	0.79
Our (*VP-2* with SF)	41	0%	0.06%	NA	∞	19($\pm$13)	0.98
Our (*VP-3* with SF)	41	0%	0.06%	NA	∞	27($\pm$15)	0.93

9.3.3 Comparison with previous research

We compared our research results with previous results based on the same dataset [46]. Table 9.5 shows the previous research results in terms of ANIA/ANGA by using the conversion technique described in the Section 4.3.1. We can see that our methods outperform the previous research for *VP-1*, while for *VP-2* and *VP-3* our ANIA values are higher. Note that in [46] and [128] the analysis was done in the same manner as we have done for *VP-1*. For all the verification processes we found high *P-values* values, which indicate that the results obtained in this analysis are statistically significant [79].

Tables 9.6 and 9.7 show the comparison of our CA approach with state of the art *Periodic Authentication (PA)* approaches with the same preprocessing and classification technique for *Dataset-3* and *Dataset-4* respectively with variable block size. From both tables we can see that our system performs better when comparing the ANGA values *i.e.* the CA system's ANGA value is much higher than the comparable PA system's ANGA value (marked as bold), when ANIA values are approximately the same.

9.3.4 Result Analysis With Context

The main objective of this analysis is not to find a better CA method, but to determine if including the context in the analysis has an impact on the performance. In this section we present the results we obtained from our analysis by considering this context for the *Dataset-4*. We measure the CA system performance by training the system using the data obtained from a particular task and then testing the system with the data from the various tasks performed by the users. Tables 9.8, 9.9, and 9.10 show the results we obtained from the *VP-1*, *VP-2*, and *VP-3* verification processes respectively by using the *Score Fusion (SF)* technique. In all cases we note that genuine users were never locked

Table 9.6: Comparison between PA and our CA for *Dataset-3*.

Approach	Block Size	*VP-1*				*VP-2*				*VP-3*			
		FNLR	FLR	ANIA	ANGA	FNLR	FLR	ANIA	ANGA	FNLR	FLR	ANIA	ANGA
PA	2	24.0	8.0	3	25	23.8	9.2	3	22	22.8	8.8	3	23
	3	20.8	6.1	4	49	21.5	7.1	4	42	**20.9**	**6.4**	**4**	**47**
	4	**18.0**	**6.3**	**5**	**64**	**20.2**	**5.2**	**5**	**77**	19.7	4.6	5	86
	5	17.8	3.6	6	138	17.8	5.1	6	98	17.6	4.1	6	121
	6	16.5	3.0	7	198	17.1	3.6	7	165	16.8	3.1	7	191
	7	16.2	2.6	8	267	16.3	3.5	8	201	15.7	2.8	8	250
	8	14.7	2.6	9	308	14.9	3.4	9	232	14.8	2.3	9	344
	9	13.6	1.8	10	488	14.6	2.1	11	434	14.6	1.6	11	570
	10	12.7	1.9	11	536	14.7	1.4	12	711	13.9	0.9	12	1102
Our CA				**5**	∞			**5**	∞			**4**	∞

Table 9.7: Comparison between PA and our CA for *Dataset-4*.

Approach	Block Size	*VP-1*				*VP-2*				*VP-3*			
		FNLR	FLR	ANIA	ANGA	FNLR	FLR	ANIA	ANGA	FNLR	FLR	ANIA	ANGA
PA	2	30.1	12.3	3	16	32.2	14.7	3	14	31.3	16.0	3	13
	3	28.8	11.0	4	27	30.9	13.6	4	22	31.9	13.1	4	23
	4	28.8	9.5	6	42	30.6	11.6	6	34	32.1	11.3	6	35
	5	26.5	9.7	7	51	29.4	11.7	7	43	32.0	9.4	7	53
	6	26.3	9.0	8	66	31.2	8.9	9	67	30.4	10.5	9	57
	7	27.6	7.2	10	98	30.8	8.4	10	83	30.9	8.8	10	79
	8	**26.4**	**6.9**	**11**	**117**	30.0	8.6	11	93	29.6	9.3	11	86
	9	27.4	5.1	12	176	29.6	8.4	13	107	28.6	9.9	13	91
	10	25.5	6.3	13	159	29.0	8.5	14	118	29.4	8.2	14	122
	11	25.1	5.7	15	193	27.7	8.8	15	125	30.5	6.3	16	175
	12	24.6	5.9	16	204	30.0	6.3	17	191	29.1	7.2	17	167
	13	23.6	6.1	17	213	**28.2**	**6.7**	**18**	**193**	29.6	6.0	18	218
	14	24.2	6.1	18	229	**28.4**	**6.9**	**20**	**202**	27.6	7.1	19	196
	15	23.5	6.1	20	245	28.2	6.9	21	217	26.9	7.5	21	199
	16	23.3	5.7	21	279	29.2	5.5	23	293	26.3	8.2	22	195
	17	21.0	7.5	22	227	28.4	5.1	24	333	26.9	7.0	23	244
	18	24.2	4.4	24	412	28.9	4.9	25	366	28.0	5.9	25	304
	19	22.4	5.3	24	360	28.1	5.2	26	362	27.6	5.7	26	335
	20	21.9	5.1	26	392	29.2	4.2	28	480	**26.2**	**7.1**	**27**	**284**
Our CA				**11**	∞			**19**	∞			**27**	∞

Table 9.8: Results obtained for context dependent evaluation of *VP-1* on *Dataset-4*.

Train \ Test	Task-1	Task-2	Task-3	Task-4	Task-5	Task-6	Task-7
Task-1	**3 (0.1%)**	3 (0.1%)	16 (4.1%)	30 (3.1%)	30 (2.8%)	11 (4.9%)	42 (1.1%)
Task-2	7 (0.6%)	**3 (0.1%)**	15 (3.6%)	28 (2.7%)	29 (2.4%)	12 (4.9%)	33 (0.5%)
Task-3	14 (2.8%)	19 (4.4%)	**4 (0.2%)**	30 (2.6%)	32 (2.8%)	13 (7.7%)	53 (3.8%)
Task-4	22 (4.5%)	20 (3.6%)	20 (4.4%)	**3 (0.1%)**	6 (0.1%)	15 (4.4%)	15 (0.5%)
Task-5	16 (2.3%)	27 (4.9%)	19 (4.4%)	7 (0.6%)	**4 (0.0%)**	17 (9.3%)	11 (0.5%)
Task-6	33 (19.2%)	30 (12.6%)	22 (9.3%)	54 (17.0%)	43 (13.7%)	**4 (0.0%)**	42 (1.6%)
Task-7	32 (19.2%)	47 (25.8%)	48 (33.0%)	13 (3.8%)	12 (2.7%)	15 (8.8%)	**4 (0.0%)**

out, hence we found that ANGA=∞, so these values are not reported. The tables only contain the found system ANIA values, as well as the *Imposter Non-Detected Rate (INDR)* between brackets. The values on the diagonal are displayed in bold to signify that training and testing is done using the same task, while for all off-diagonal values the training and testing is done with 2 different tasks.

Recall that tasks 1 to 4 had 41 participants, while task 5 had only 40 participants and the last 2 tasks had only 14 participants. When training with task i and testing with task j we only considered those participants that participated in both tasks. Also recall that tasks 1, 2, 3, and 6 are article reading tasks and tasks 4, 5, and 7 are image comparison games, which will help to understand the results and it's impact based on the type of the tasks.

On average we find that ANIA equals 3.6/7.3/7.7 for *VP-1*/*VP-2*/*VP-3* when training and testing with the same task, while it equals on average 23.9/25.1/21.8 when using different tasks for

Table 9.9: Results obtained for context dependent evaluation of *VP-2* on *Dataset-4*.

Train \ Test	Task-1	Task-2	Task-3	Task-4	Task-5	Task-6	Task-7
Task-1	**6 (0.2%)**	21 (4.0%)	17 (3.8%)	31 (3.3%)	29 (1.9%)	20 (12.1%)	51 (2.7%)
Task-2	11 (1.7%)	**5 (0.2%)**	18 (3.2%)	31 (3.5%)	28 (2.5%)	18 (11.5%)	46 (2.7%)
Task-3	26 (7.7%)	24 (6.7%)	**10 (1.6%)**	35 (4.7%)	35 (3.7%)	20 (13.2%)	48 (6.0%)
Task-4	27 (6.8%)	30 (4.4%)	33 (8.3%)	**10 (1.1%)**	15 (1.5%)	19 (10.4%)	14 (1.6%)
Task-5	22 (4.2%)	27 (4.6%)	24 (4.6%)	8 (0.7%)	**6 (0.3%)**	18 (6.6%)	18 (2.2%)
Task-6	16 (4.4%)	16 (1.6%)	21 (2.7%)	34 (0.0%)	38 (0.5%)	**7 (2.2%)**	39 (2.2%)
Task-7	22 (0.4%)	26 (1.2%)	30 (2.7%)	18 (0.0%)	17 (1.1%)	14 (4.4%)	**7 (0.5%)**

Table 9.10: Results obtained for context dependent evaluation of *VP-3* on *Dataset-4*.

Train \ Test	Task-1	Task-2	Task-3	Task-4	Task-5	Task-6	Task-7
Task-1	**9 (1.3%)**	18 (3.9%)	20 (5.2%)	28 (2.3%)	23 (2.1%)	19 (10.4%)	36 (4.4%)
Task-2	11 (1.6%)	**7 (0.9%)**	19 (5.0%)	30 (3.5%)	27 (1.9%)	15 (10.4%)	40 (2.2%)
Task-3	17 (4.5%)	20 (0.9%)	**7 (0.9%)**	24 (2.7%)	22 (1.3%)	14 (5.5%)	33 (2.2%)
Task-4	22 (4.2%)	21 (2.1%)	20 (5.1%)	**6 (0.5%)**	8 (0.7%)	15 (7.1%)	21 (1.1%)
Task-5	17 (2.4%)	24 (4.5%)	19 (4.1%)	9 (0.9%)	**8 (0.6%)**	12 (1.6%)	7 (0.0%)
Task-6	14 (4.9%)	13 (1.1%)	32 (8.8%)	39 (1.6%)	38 (0.5%)	**11 (4.4%)**	30 (1.1%)
Task-7	23 (8.2%)	33 (12.6%)	30 (7.1%)	16 (0.5%)	19 (1.6%)	15 (7.6%)	**6 (0.0%)**

training and testing. The last values are split according to the type of task (reading or image comparison) and we found that if the task is different, but the type of task is the same, then the ANIA equals 14.4/17.7/16.3, while if also the type of tasks differs, then the ANIA values are much higher: 31.0/30.7/25.9. Finally, if we consider a reading task for training then the system ANIA for any of the other tasks is 26.7/28.0/24.3 and for an image comparison tasks these values are 20.1/21.2/18.4.

9.4 Discussion

We observed some phenomena during our analysis, which are described below.

- For each of the three verification processes genuine users are never locked out from the system for both the datasets. This holds when using no context, when using context with the same task, and when using context with different tasks.

- We found that, for $T_{lockout} = 90$ the ANIA value is higher than the T_{us} for any given verification processes and datasets. Therefore, we can say that the system trust on the genuine users was always staying above 90%.

- In terms of ANIA we can conclude that the value decreases when the context is used, by approximately a factor 3. Similar findings are found in [71]. However, we do also note that the INDR goes up slightly when including context in our analysis, but the average values stay low when the same task is considered.

  Regarding the ANGA values, we note that in all cases we improve over known results, even in the more realistic cases where no (*i.e. VP-3*) or only partial (*i.e. VP-2*) imposter data is available to build the user model, our context based CA system outperforms the state of the art results. In particular for the *VP-1* verification process the improvement is impressive.

- Except four cases we have found that if we train and test with same tasks the performance was improved for all the verification processes. These four cases are: 1) Train with *Task-6* and test with *Task-4* for *VP-2*; 2) Train with *Task-7* and test with *Task-4* for *VP-2*; 3) Train with *Task-5* and test with *Task-7* for *VP-3*; 4) Train with *Task-6* and test with *Task-5* for *VP-3*.

- We found that the number of users in the '+ / +' category dropped when analysing cross type of tasks (*i.e.* Article Reading or Image Compression Game) for all the verification processes.
- Training with an image comparison task gives slightly better results when testing with an arbitrary other task.
- We can clearly see that testing with the same type of tasks gives better ANIA values than when testing is done with a different type of task, although those first results are still worse then when comparing with the same task.

There could be multiple reasons to obtain these findings. Some of them are pointed out below.

- Classifier *Over-Fitting* and *Under-Fitting* was one of the reason to get some unexpected results.
- We have used *Linear Search* to optimize all the algorithmic parameters and the lockout threshold. According to our understanding evolutionary techniques (*i.e. Genetic algorithm*, *Particle Swarm Optimization* or *Ant Colony Optimization etc.*) with a separate development dataset could improve the results. Due to the small number of strokes present in the dataset we are unable to do so.

9.5 Summary

In this chapter, we provided the detailed description of a CA system by using swipe gestures for mobile devices. All the experiments were conducted on two publicly available datasets, which makes the achieved results completely reproducible. For both datasets we achieved better results when compare to the state of the art approaches.

Part III

Continuous Identification

Chapter 10

Continuous Identification Concepts

In this chapter, we introduce the concept of *Adversary Identification* in combination with *Continuous Authentication*. From a security point of view it is important to not only use the traditional access control at the beginning of a session, but during a session continuously monitor of the current user is still the genuine user. In case an imposter is detected the system should lock to avoid loss or disclosure of personal or confidential information. In many cases it can be important to not only lock the system, but also establish the identity of the imposter. This concept has not been introduced in this manner before and it combines security and forensics in an innovative manner.

This chapter is based on the papers published in: [102] MONDAL, S., AND BOURS, P. Continuous user authentication and adversary identification: Combining security & forensics. Under Review in IEEE Transactions on Information Forensics & Security, 2016.

10.1 System Architecture

In this section, we discuss our proposed system architecture. The main motivation of this research is to unveil the identity of an imposter once the system has detected that an imposter is using the system. Figure 10.1 shows the block diagram representation of the complete system. Our proposed system is mainly divided into two major subsystems (see Figure 10.1 with dotted area), *i.e.* the *Continuous Authentication System (CAS)* and the *Continuous Identification System (CIS)*. The description of the CAS is given in the Part II. We describe the CIS subsystems in more detail in this chapter.

In Figure 10.1 we see that after *Static Login* (*i.e.* password, fingerprint, swipe pattern, *etc.*) the user is accepted as genuine and obtains the permission to use the device. Now during the use of the device each and every action performed by the user is used by the CAS and returns the current system $Trust$ in the genuineness of the user. The $Trust$ values are used in the decision module where it is compared with a predefined threshold ($T_{lockout}$) to determine whether the user can continue to use the device or, if the trust is too low, the device will be locked. After detecting that the present user is an imposter (*i.e.* $Trust < T_{lockout}$), all the performed activity done by the current user are used by the CIS to try to identify the imposter from a known/semi-known user database and return to the *Static Login* part of the system.

10.2 Continuous Identification System

Figure 10.2 shows the complete block diagram of the CIS. In this section we discuss the major components of CIS in more detail. The feature extraction and feature selection techniques are discussed in the Chapter 5 and Section 8.3 respectively. We applied *Pairwise User Coupling (PUC)* in CIS for comparison and the decision making process. PUC technique is able to reduce a multi-class classification problem into several two-class problems. This means that for each pair of users we determine, based on the available data, which of these two users most likely has generated this data. Therefore, contrary to the conventional training data preparation, pairwise training data preparation is different. We elaborate the description of this data preparation process in Section 10.2.1.

Three different schemes were developed for comparison and decision module (see Figure 10.2 marked with red line) that will be discussed in Sections 10.2.2, 10.2.3, and 10.2.4. For all schemes we show how many pairwise comparisons are made under the assumption that all pairwise comparisons require approximately the same amount of time and this will be an indication of the relative running times of the three schemes.

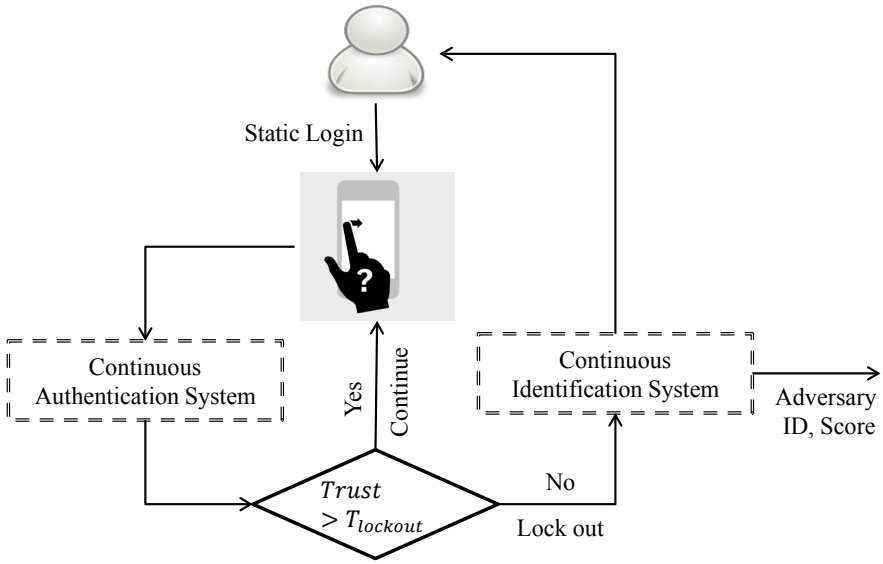


Figure 10.1: Block diagram representation of the architecture of our system.

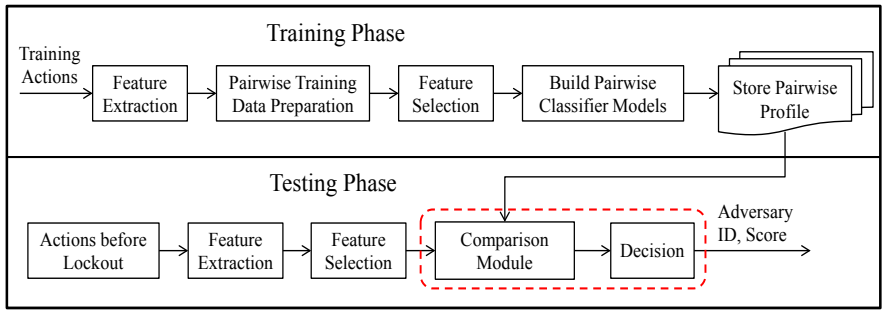


Figure 10.2: Block diagram representation of our Continuous Identification System.

10.2.1 Pairwise Training Data Preparation

Figure 10.3 shows an example of a multi-class (*i.e.* N class) training dataset for conventional training data preparation. In this example, FV_q^i represents the feature vector of user i from q^{th} sample, where $i = 1, 2, \ldots N$, $q = 1, 2, \ldots n$ and n is the total number of training samples present for user i. The last column represents the class label *i.e.* 1 to N. When we prepare our training dataset according to this process, we found a lower learning accuracy of the classifiers and we are unable to achieve the desired results. Therefore, we came up with a solution called PUC, where the multi-class classification problem will be divided into several two class classification problems.

Figure 10.4 shows an example of a multi-class (*i.e.* N class) training dataset for pairwise training data preparation. In this example, we can see that multiple training datasets are created (*i.e.* TP_j^i) for user i and j, where $i = 1, 2, \ldots N$ and $j \in J_i = \{[1, 2, \ldots N] - [i]\}$. The data samples from user i and j have 1 and 2 as a class label respectively. During testing, if the test sample originates from user i, then the score (sc) will be greater than 0.5, *i.e.* $sc > 0.5$ otherwise $sc < 0.5$. After that we

Feature Vector	Class Label
FV_1^1	1
...	...
FV_n^1	1
FV_1^2	2
...	...
FV_n^2	2
...	...
FV_1^{N-1}	$N-1$
...	...
FV_n^{N-1}	$N-1$
FV_1^N	N
...	...
FV_n^N	N

Figure 10.3: Conventional training data preparation.

train the classifier for every training pair TP_j^i and store the pairwise classifier models (*i.e.* CIS_j^i) to be used in comparison and decision module.

10.2.2 Scheme 1 (S1)

Algorithm 10.1, shows the algorithm for *Scheme 1 (S1)*. Figure 10.5 shows an example of a graphical representation of Algorithm 10.1 where the total number of users is 20 and the required rank is 1 *i.e.* $N = 20$ and $r = 1$. In this particular figure, the pairs are created in increasing order but in our actual analysis we have selected these pairs randomly in each round of the loop. We can see that after each iteration the number of users is reduced until the number of users satisfies the required rank based on the maximization of the score *i.e.* $S^i > S^j$ where $S^i = \frac{1}{m}\sum_{p=1}^{m} sc_p$ and $S^j = 1 - S^i$. If the number of users in a round is even $(2n)$, then in this round the number of users will be halved (n). In case of an odd number of users $(2n + 1)$, one user will continue without comparison, so at the end of the round $n + 1$ users are left over. The number of comparisons T_1 for this scheme, when starting with N users and stopping at rank r is $T_1(N, r) = N - r$.

10.2.3 Scheme 2 (S2)

Algorithm 10.2, shows the algorithm for *Scheme 2 (S2)*. Figure 10.6 shows an example of a graphical representation of Algorithm 10.2 where $k = 6$ and $r = 1$. During comparison, we first randomly choose k pairs from CIS^i set for the i^{th} subject and calculate the classification score for test set Γ for each of these k pairs. This will give total $k \times m$ score values for any given classifier. Let, λ_p denote the score values of of any given classifier where $p = 1, 2, \ldots m$, so we define $sc^q = \lambda$ denote the score vector obtain q^{th} pair where $q = 1, 2, \ldots k$. Then we obtain the resultant score for i^{th} user by $S^i = \frac{1}{m \times k}\sum_{q=1}^{k}\sum_{p=1}^{m} sc_p^q$. We repeat this procedure for all users (*i.e.* $i = 1, 2, \ldots N$) and select the user with the largest score S^i as the identified user. The number of comparisons T_2 for this scheme is independent of r, but depends on N and k. In particular we have $T_2(N, k) = N \times k$.

10.2.4 Scheme 3 (S3)

Algorithm 10.3, shows the algorithm for *Scheme 3 (S3)*. Let $\Delta = [\delta_1, \delta_2, \ldots \delta_c]$ be the set of the *Rank-c* users after applying S2 where $\Delta \subseteq \{1, 2, \ldots N\}$. Now we repeat S2 with the users in the set Δ with a fixed k *i.e.* $k' = c - 1$, meaning that we consider all imposter users in the reduced set

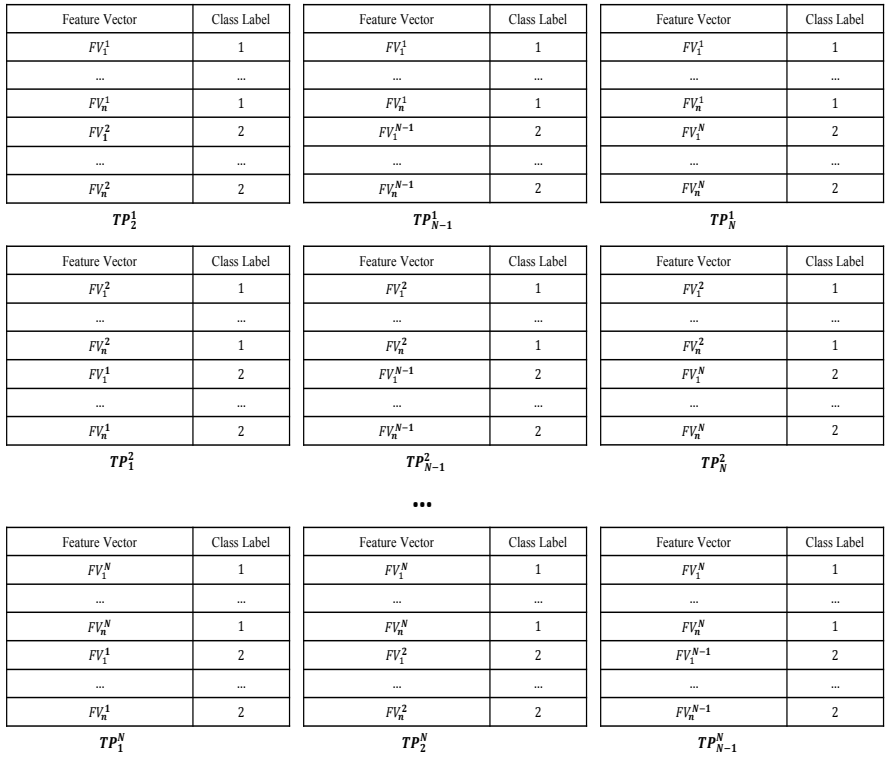


Figure 10.4: Pairwise training data preparation.

Algorithm 10.1: Algorithm for Scheme 1

Input: $CIS_j^i \leftarrow$ The pairwise classifier model where $i \in I = \{1, 2, 3 \dots N\}$, N is the number of users and $j \in J_i = I - \{i\}$; $\Gamma \leftarrow$ The set of test actions, where $|\Gamma| = m$ and m is the number of performed actions; $r \leftarrow$ The required rank.

Output: $User_{id,\ score}$

while $|I| > r$ **do**
  $k = \left\lfloor \frac{|I|}{2} \right\rfloor$; $T = I$; $SC = \emptyset$
  while $|I| \geq k$ **do**
    $i = random\{T\}$; $T = T - \{i\}$; $j = random\{T\}$; $T = T - \{j\}$
    Calculate score sc (*i.e.* $sc_p = P(\gamma_p|H_i)$ where γ_p is the feature vector after feature selection of the performed action p and H_i is the hypothesis for the user i) for all test actions Γ from the selected pair CIS_j^i with given classifier(s).
    So, $S^i = \frac{1}{m} \sum_{p=1}^{m} sc_p$; $S^j = 1 - S^i$
    if $S^i > S^j$ **then**
      $I = I - \{j\}$; $SC = SC \cup \{(i, S^i)\}$
    else
      $I = I - \{i\}$; $SC = SC \cup \{(j, S^j)\}$
$User_{id,\ score} = SC$

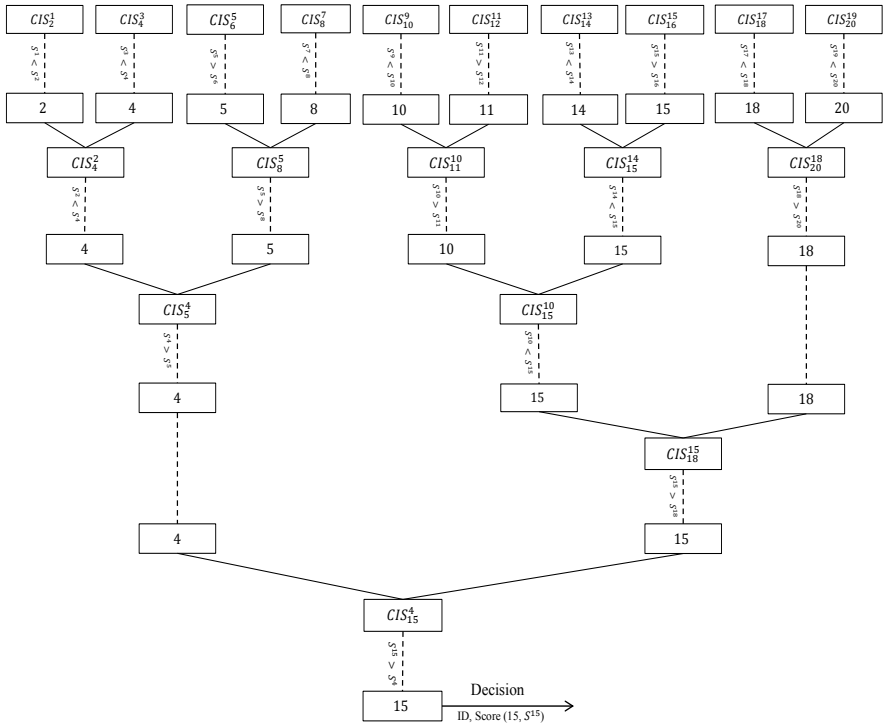


Figure 10.5: Example of a graphical representation of Algorithm 10.1 where $N = 20$ and $r = 1$.

Algorithm 10.2: Algorithm for Scheme 2

Input: $CIS_j^i \leftarrow$ The pairwise classifier model where $i \in I = \{1, 2, 3 \dots N\}$, N is the number of users and $j \in J_i = I - \{i\}$; $\Gamma \leftarrow$ The set of test actions, where $|\Gamma| = m$ and m is the number of performed actions; $k \leftarrow$ The number of comparison and $k < N$; $r \leftarrow$ The required rank.

Output: $User_{id,\ score}$

$SC = \emptyset$; $T = I$
while $|I| \neq 0$ **do**
  $i = \{I\}$; $I = I - \{i\}$; $J = T - \{i\}$; $S^i = 0$
  while $k > 0$ **do**
    $j = random\{J\}$; $J = J - \{j\}$
    Calculate score sc (*i.e.* $sc_p = P(\gamma_p|H_i)$ where γ_p is the feature vector after feature selection of the performed action p and H_i is the hypothesis for the user i) for all test actions Γ from the selected pair CIS_j^i with given classifier(s).
    So, $S^i = S^i + \frac{1}{m \times k} \sum_{p=1}^{m} sc_p$; $k = k - 1$
  $SC = SC \cup \{(i, S^i)\}$
$X = \{i|(i, S^i) \in SC\}$
$X_1 \subseteq X$ where $|X_1| = r$
$User_{id,\ score} = \{(i, S^i)| \min\{S^i|i \in X_1\} > \max\{S^j|j \in X - X_1\}\}$

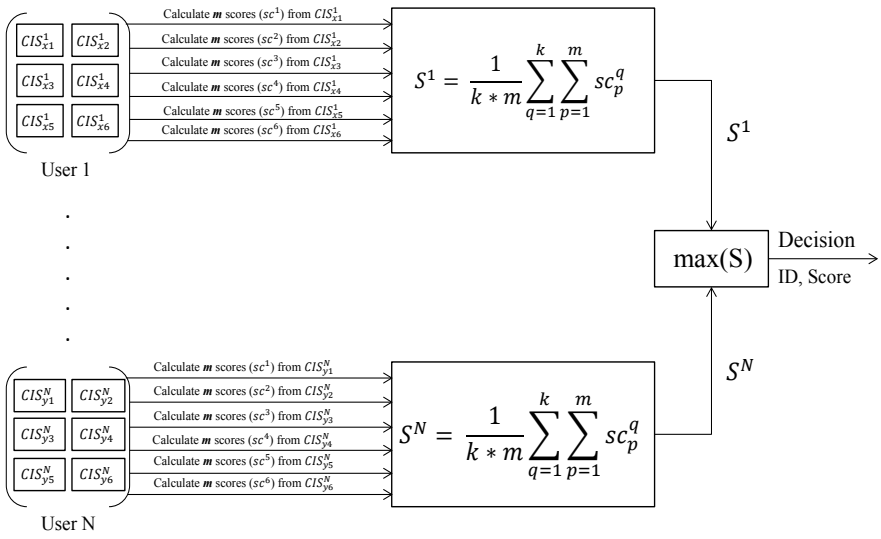


Figure 10.6: Example of a graphical representation of Algorithm 10.2 where $k = 6$ and $r = 1$.

Algorithm 10.3: Algorithm for Scheme 3

Input: $CIS^i_j \leftarrow$ The pairwise classifier model where $i \in I = \{1, 2, 3 \dots N\}$, N is the number of users and $j \in J_i = I - \{i\}$; $\Gamma \leftarrow$ The set of test actions, where $|\Gamma| = m$ and m is the number of performed actions; $k \leftarrow$ The number of comparison for Algorithm 10.2 and $k < N$; $c \leftarrow$ The rank correction; $r \leftarrow$ The required rank and $r < c$.

Output: $User_{id,\ score}$

$T \leftarrow$ The set of Rank - c users after applying Algorithm 10.2
$SC = \emptyset$; $I = T$
while $|I| \neq 0$ **do**
  $i = \{I\}$; $I = I - \{i\}$; $J = T - \{i\}$; $t = |J|$; $S^i = 0$
  while $t > 0$ **do**
    $j = \{J\}$;
    Calculate score sc (*i.e.* $sc_p = P(\gamma_p|H_i)$ where γ_p is the feature vector after feature selection of the performed action p and H_i is the hypothesis for the user i) for all test actions Γ from the selected pair CIS^i_j with given classifier(s).
    So, $S^i = S^i + \frac{1}{m \times t}\sum_{p=1}^{m} sc_p$; $t = t - 1$
  $SC = SC \cup \{(i, S^i)\}$
$X = \{i|(i, S^i) \in SC\}$
$X_1 \subseteq X$ where $|X_1| = r$
$User_{id,\ score} = \{(i, S^i)|\min\{S^i|i \in X_1\} > \max\{S^j|j \in X - X_1\}\}$

of c users. We would like to mention that S3 is not an independent scheme, it is a combination of S2 with an additional correction made after getting the *Rank-c* users from S2. This scheme can be considered as using S2 for a re-ranking process after the initial S2 scheme. For S3 the number of comparisons T_3 depends on N, k and c. To be precise we have $T_3(N, k, c) = T_2(N, k) + c \times (c - 1)$.

10.3 Experimental Protocol

In our study, we followed two experimental protocols, *i.e.* closed-set and open-set experiment. These protocols described in below.

10.3.1 Protocol 1

In this protocol, all the users are known to the system for both CAS and CIS. In case of CAS, the imposter part of the training data is taken from all $N-1$ imposters. This protocol is the same as described for *VP-1* (see Section 4.2.2.1 for more details).

In case of CIS, the set of users I for all the schemes (*i.e.* Algorithm 10.1, 10.2 and 10.3) will be $I = \{1, 2, 3 \dots N\}$. Therefore, this protocol can be seen as a closed system where the adversary is known to the system.

10.3.2 Protocol 2

In this protocol, 50% of the users are known to the system for both CAS and CIS. In case of CAS, the classifiers are trained with data from the genuine user as well as data of $\lfloor \frac{N-1}{2} \rfloor$ of the imposter users. This protocol is the same as the one described for *VP-2* (see Section 4.2.2.2).

In case of CIS, the set of users I for all the schemes will be $I = \{1, 2, 3 \dots \lfloor \frac{N-1}{2} \rfloor + 1\}$. Therefore, this protocol can be seen as a open-set system where 50% of the probable adversaries is known to the system and the other 50% are completely unknown to the system.

10.4 Performance Measure

The system performance measure techniques that are applied in this analysis will be discussed in this section. The CAS performance measure technique is common for both the experimental protocols *i.e. Protocol-1* and *Protocol-2*. The complete description of CAS performance measure technique was provided in Chapter 4.

Due to the different objective of these two protocols we have applied two different performance measure techniques for CIS. The description of these techniques are given below.

10.4.1 CIS Performance Measure for Protocol 1

This is a straight forward performance measure for *Protocol-1*, where every time that a user is locked out by the CAS system, the adversary ID is determined by the CIS system. In Figure 10.7 we displayed these adversary IDs in green (meaning a successfully identified adversary) and red (meaning identification of another person than the actual adversary). When the system is unsuccessful in identifying the correct adversary ID, we display the Rank-1 identity along with the rank of the correct imposter. In Figure 10.7 one such case is presented, where the system identifies the imposter as number 38, while the correct adversary ID of 8 is found at Rank-2 (R-2 marked in blue). During this experiment out of the 22 lockouts the CIS identified the correct adversary 21 times, therefore, the recognition accuracy for this example is $ACC_8^{22} = 95.45\%$. Note that here 22 is the genuine user and 8 is the imposter user.

10.4.2 CIS Performance Measure for Protocol 2

To measure the system performance for *Protocol-2*, we used a threshold (*i.e.* T_{open}) that will decide whether the adversary is within the group of known users or not. If the $User_{score} \geq T_{open}$ (see Algorithms 10.1, 10.2 or 10.3 for $User_{score}$) then we will say that the adversary is within the set of known adversaries, otherwise he/she said to be an unknown adversary. If we find that the adversary is known to the system, then the system will establish the identity of the adversary. In our study, we will define four different values that will provide the overall system performance for this protocol, where the summation of these four values will be 100%.

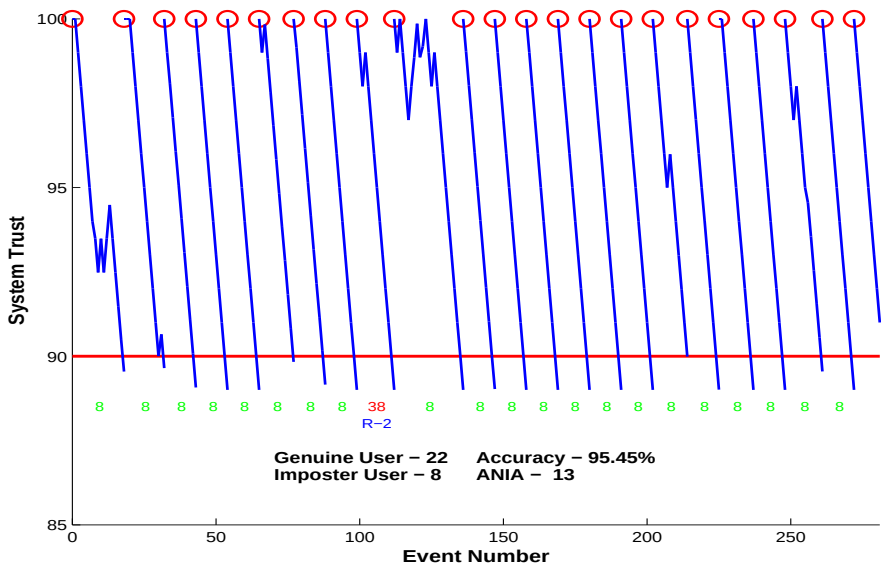


Figure 10.7: CIS performance measure for *Protocol-1* with genuine user 22 and imposter user 8.

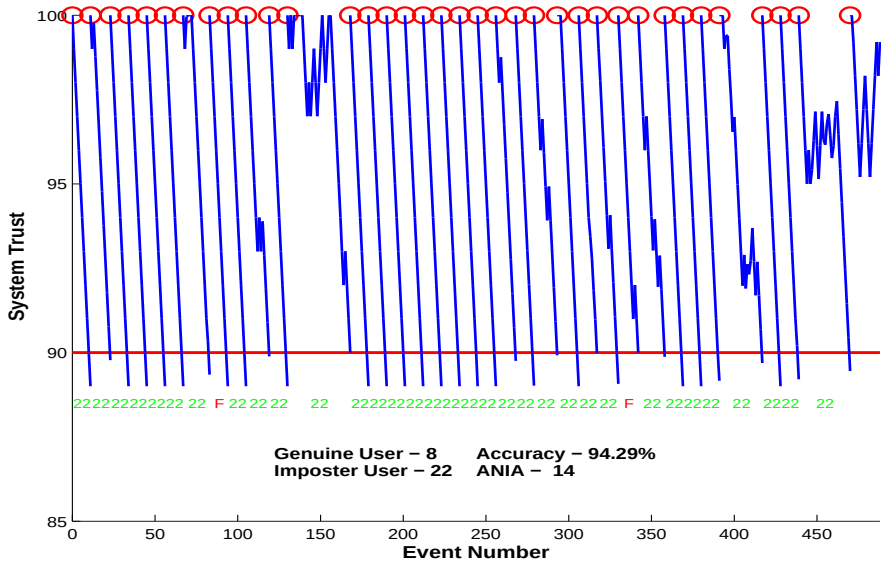


Figure 10.8: CIS performance measure for *Protocol-2* with genuine user 8 and imposter user 22.

- ***True ID (TID) :*** Where $User_{score} \geq T_{open}$ *i.e.* adversary is within the known user set and correctly identified.

- ***False ID (FID) :*** This is the sum of two different components.

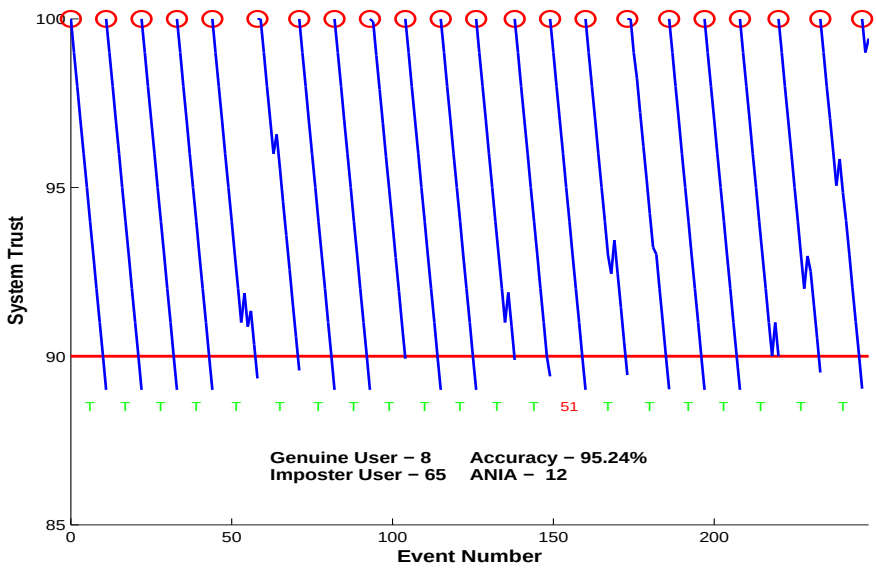


Figure 10.9: CIS performance measure for *Protocol-2* with genuine user 8 and imposter user 65.

  - $User_{score} \geq T_{open}$ *i.e.* adversary is within the known user set but, falsely identified.
  - In the second case, the adversary is not in the known user set, but the system says otherwise with a false adversary ID *i.e.* $User_{score} \geq T_{open}$.

- ***True Not In (TNotIn) :*** Where $User_{score} < T_{open}$ *i.e.* adversary was indeed not in the known user set.

- ***False Not In (FNotIn) :*** In this case system says the adversary is not in the known user set (*i.e.* $User_{score} < T_{open}$) but, actually the adversary is within the known user set.

In Figure 10.8 one case was present for *Protocol-2* with genuine user 8 and imposter user 22 where imposter user 22 presents in the known adversary user set. In this example, we see that two cases where the system says the adversary is not in the known user set (marked F in red color). Therefore, in this example $TID^8_{22} = 94.29\%$, $FNotIn^8_{22} = 5.71\%$, $TNotIn^8_{22} = 0\%$ and $FID^8_{22} = 0\%$.

In Figure 10.9 another case was present for *Protocol-2* with genuine user 8 and imposter user 65 where imposter user 65 not present in the known adversary user set. In this example, we see that one case where the system says the adversary is in the known user set with a false ID (marked 51 in red color). Therefore, in this example $TNotIn^8_{65} = 95.24\%$, $FID^8_{65} = 4.76\%$, $TID^8_{65} = 0\%$, $FNotIn^8_{65} = 0\%$. In these examples (*i.e.* Figures 10.7, 10.8, and 10.9), we have used *Dataset-3* (see Section 5.3.1 for more details about this dataset) and $T_{open} = 0.8$.

10.5 Summary

In this section we provided the following:

- We described three different identification schemes using pairwise user coupling. This approach was followed to mitigate the problem of behavioural biometric modalities (*i.e.* low inter-class variation and high intra-class variation). But note that this approach can be applied to any pattern identification problem.

- For the three described algorithms the number of pairwise comparisons increases from S1 to S2 and from S2 to S3, so $T_1 \leqslant T_2 \leqslant T_3$.
- We believe that the assumption made in S1, *i.e.* the correct subject always has the highest score for every pair of analysis. This assumption is not always valid for every dataset, especially for behavioural biometrics based datasets because of large intra-class variations. Therefore, it might produce lower identification accuracy.

Chapter 11

Continuous Identification using a Combination of Keystroke and Mouse Dynamics

In this chapter, we investigated the performance of a continuous identification for a PC under various analysis techniques discussed in Chapter 10. We have used our own datasets for this analysis *i.e. Dataset-2*. This dataset is a combination of keystroke and mouse usage behaviour data. The complete description of this dataset and the extracted features are given in the Section 5.2.

This chapter is based on the papers published in: [101] MONDAL, S., AND BOURS, P. Combining keystroke and mouse dynamics for continuous user authentication and identification. In *IEEE Int. Conf. on Identity, Security and Behavior Analysis (ISBA'16)* (2016), IEEE, pp. 1–8.

11.1 Background Knowledge

In this section, we discuss the background knowledge required to better understand this research. This includes the classifier used in this study and the profile creation process.

11.1.1 Classifier

Person identification by analysing the user's behaviour profile is challenging due to limited information, large intra-class variations and the sparse nature of the information. We have also observed that the statistical analysis (*i.e.* distance based classifiers) failed to achieve the desired results due to these challenges. Therefore a machine learning based approach was followed in this research, more precisely we have used the *Decision Tree (DT)* classifier in our research. A brief description of this classifier is given below.

DT is a tree structure based predictive learning model which maps features of an observation(s) about an item to the item's target value where leaves represent class labels and branches represent conjunctions of features that lead to those class labels [12]. In this study, we have used *Bagging (bootstrap aggregation)* DT which gives stability and accuracy for the classifier.

11.1.2 Profile Creation

In this section we describe the profile creation process for CAS and CIS (see Section 10.1 for the description of CAS and CIS). The complete CAS profile creation process was described in the Chapter 8 for both the experimental protocols used in this research (see Section 10.3 about the details of these protocols). We have used our own datasets for the analysis *i.e. Dataset-2*. There are 53 participants in this dataset, but, we have only used 25 participants in this research.

We use the DT classifier with the PUC approaches for CIS. For each combination of genuine user i and imposter user j we created a training set CIS^i_j. This classifier was trained with the training data m_i of user i and the training data m_j of user j. For example, we have $N = 25$ users from *Dataset-2*, so we have $N \times (N-1) = 600$ different CIS classifier models for any given action (see Section 5.2.5 and 5.2.6 about these actions). We have six different types of actions, therefore in total $6 \times 600 = 3600$ DT classifier models were generated. Before building the classifier models, we first apply the feature selection technique for both CAS and CIS as described in Section 8.3 for mouse move and drag-drop actions.

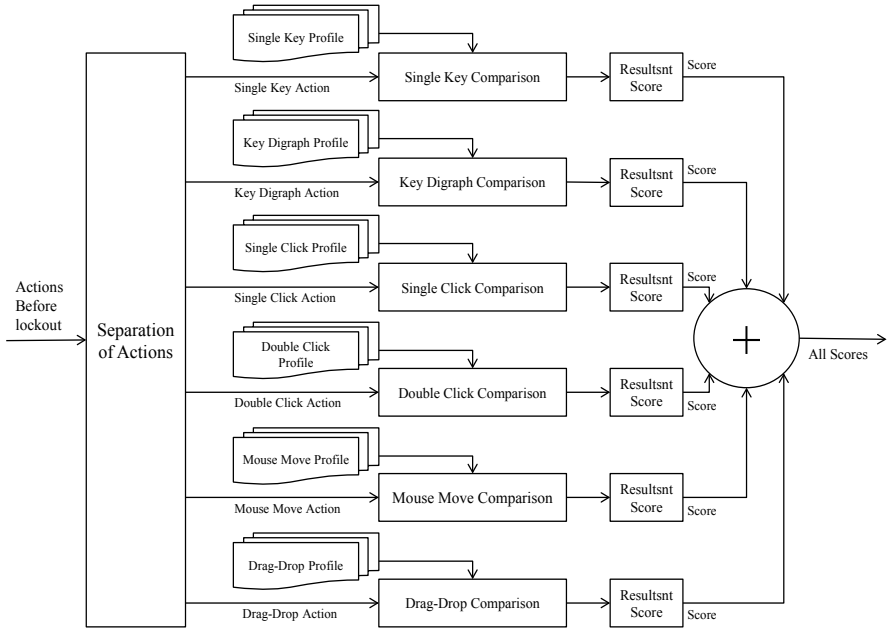


Figure 11.1: Expanded block diagram of the CIS *Comparison Module*.

We have also experimented with other classifiers *i.e.* SVM, ANN and CPANN in this research, but, due to a lower learning accuracy we did not use these classifiers in the analysis.

11.1.3 System Architecture

The CIS comparison and decision making process described in Section 10.2 is valid when we have only one type of actions performed by the users. Due to different different types of actions performed by the users, we applied different classifier models for different actions. The block diagram of the expanded CIS *Comparison Module* is shown in Figure 11.1.

In Figure 11.1 we can see that first we separate different actions performed by the users before lockout from CAS. For example, when a user was able to perform 100 actions before lockout by the CAS, then this could *e.g.* be split into 40 *Single Key Actions*, 15 *Key Digraph Actions*, 20 *Mouse Move Actions*, 20 *Single click Actions*, 5 *Double click Actions* and no *Drag-Drop Actions*. The specific actions go to the corresponding comparison modules and provide scores that are input to the next module.

11.2 Periodic Analysis of the Identification Schemes

In this section, we will show the performance of the proposed identification schemes discussed in the Section 10.2 under various circumstances. We have to mention that all the analysis presented in this section was done in a periodic manner. This was done to understand how our identification schemes perform under various parameters and circumstances in this dataset. The CA-CI performance results can be found in Section 11.3.

Figure 11.2 shows the identification accuracies obtained from S1 (see Algorithm 10.1) with different ranks (*i.e.* $r = 1, 2, \ldots 25$) for various number of actions (*i.e.* $m = 100, 200, \ldots 1000$).

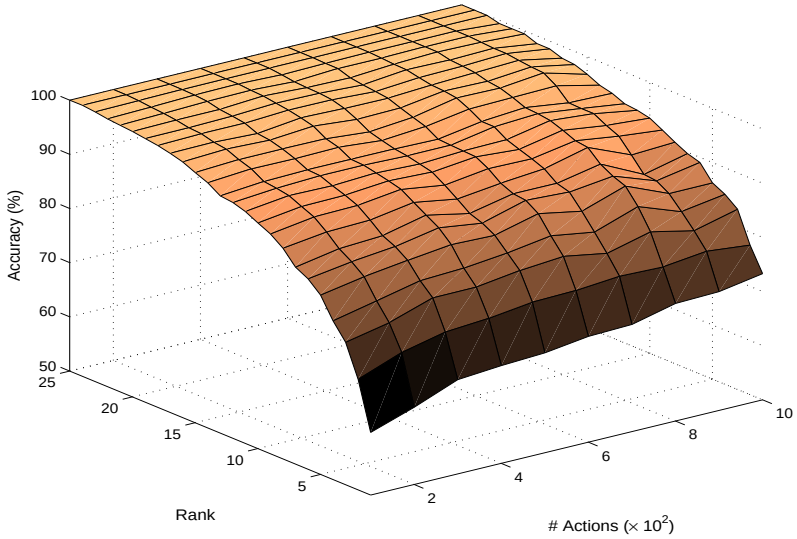


Figure 11.2: Results obtained from S1.

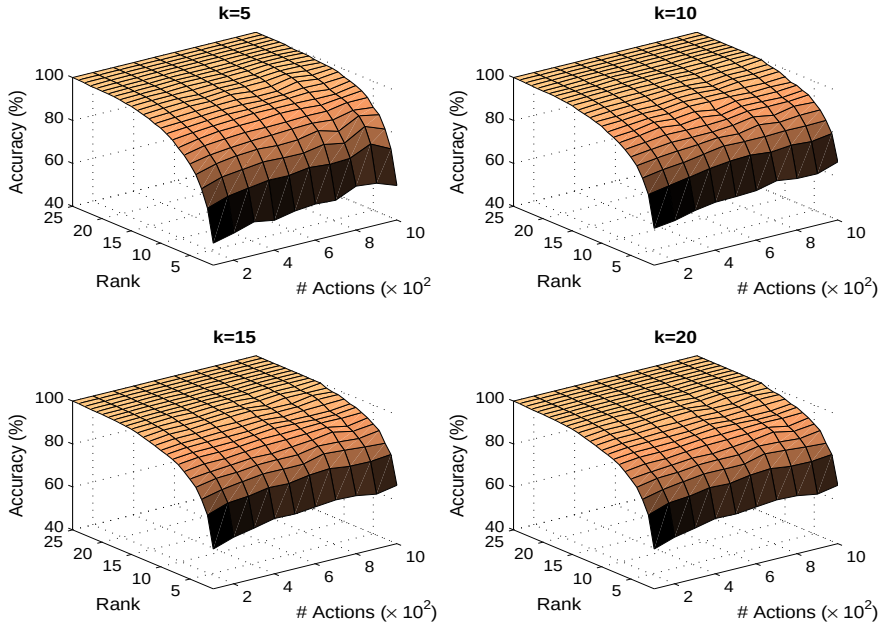


Figure 11.3: Results obtained from S2 for different k value.

We can observe from this figure that an increasing number of actions for identification will improve the *Rank-1* accuracy.

Figure 11.3 shows the identification accuracies obtained from S2 (see Algorithm 10.2) with

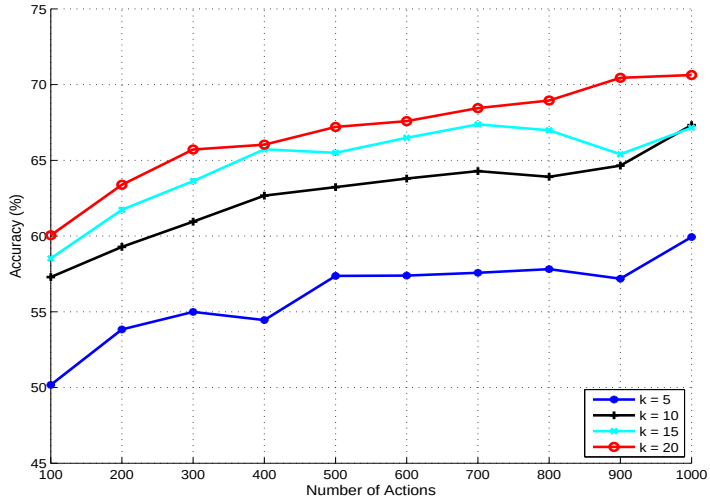


Figure 11.4: *Rank-1* accuracies obtained from S2 for different k value.

different ranks for various number of actions, with a different k value (*i.e.* $k = 5, 10, \ldots 20$). Figure 11.4, shows the *Rank-1* accuracies obtained from S2 for different k value. From this figure, we can see that a higher k value can improve the identification accuracy. We also found that there is a large difference in accuracy between *Rank-1* and *Rank-10* for any given k and m value from Figure 11.3, which motivated us to use the scheme as described in Section 10.2.4.

Figure 11.5 shows the *Rank-1* accuracies obtained from S3 for different k and c value. We can see the improvement of *Rank-1* identification accuracy by using S3 after comparing Figure 11.4 and 11.5. Figure 11.6 shows the comparison between different schemes with optimized parameter (*i.e.* k and c value) for various numbers of actions. We can see from this figure that the schemes S1 and S3 have a similar performance, and also that the performance of S2 is lower than the other schemes. We would also like to mention that the number of pairwise comparisons for the S1 scheme is lower than the other schemes (*i.e.* $T_1 < T_2 < T_3$). We do assume that all the pairwise comparison take approximately an equal amount of time.

11.2.1 Analysis on Different Actions

In this section we look at the performance of KD and MD actions separately. Figures 11.7, 11.8, and 11.9 show the *Rank-1* identification accuracies obtained from S1, S2 and S3 respectively for various numbers of actions. From these figures we can see that the identification accuracies of KD action are much better than the MD action for any giving number of actions and schemes.

Figure 11.10 shows the *Rank-1* identification accuracies obtained from all the schemes with optimized parameters for different KD and MD actions for various numbers of actions. From Figures 11.10(a) and 11.10(d) we can see that there are similar identification accuracies for different schemes for single key and mouse double click actions. For the other actions we can see large differences in identification accuracies for the S1 scheme, *i.e.* much higher accuracies when compared to the S2 and S3 schemes. There is a general sense that increasing the number of actions for identification will increase the identification accuracies, however, this phenomena is not fully observed in this analysis and for this dataset. We did not perform this analysis for the drag-drop actions because the number of actions present in the database is too low.

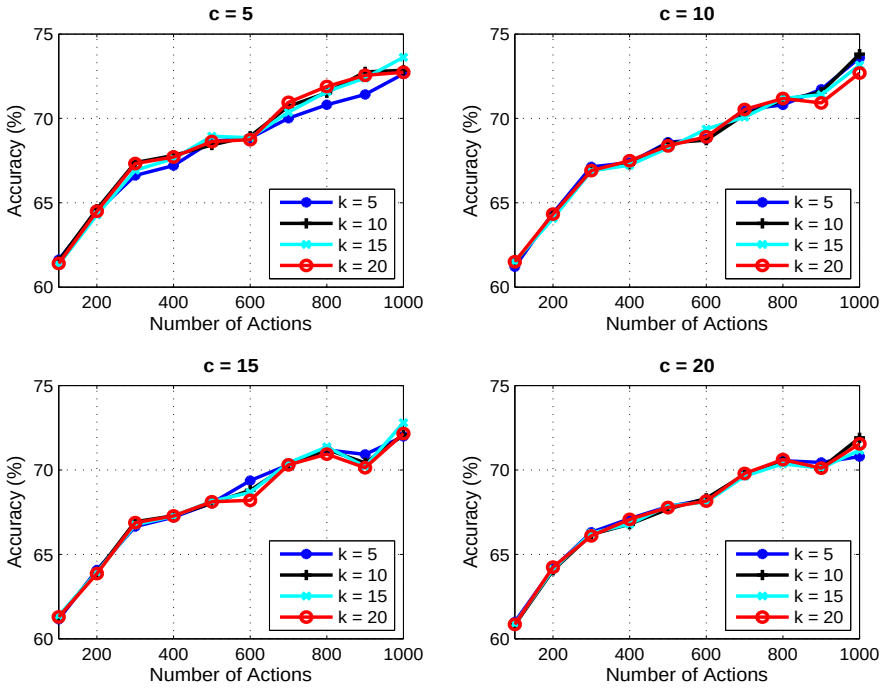


Figure 11.5: *Rank-1* accuracies obtained from S3 for different k and c value.

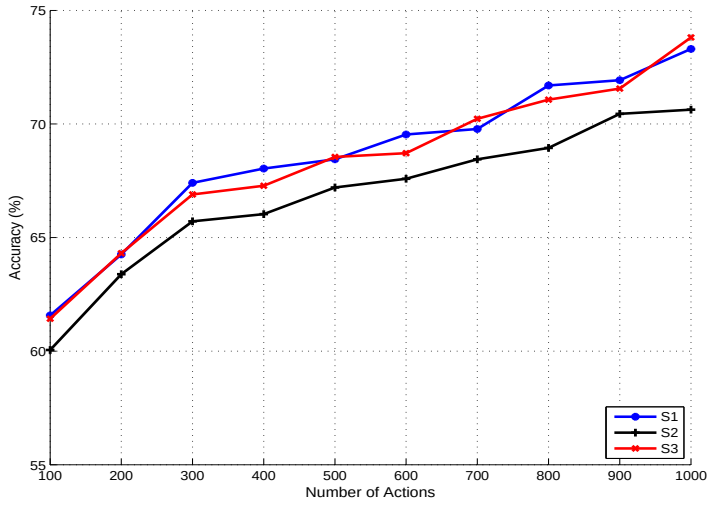


Figure 11.6: *Rank-1* accuracies obtained from all the schemes with optimized parameters.

11.3 Result Analysis

We report the results with the user specific (T_{us}) CAS lockout threshold (*i.e.* $T_{lockout}$, see Section 10.1 for more details about lockout threshold). We also report our results based on two different ex-

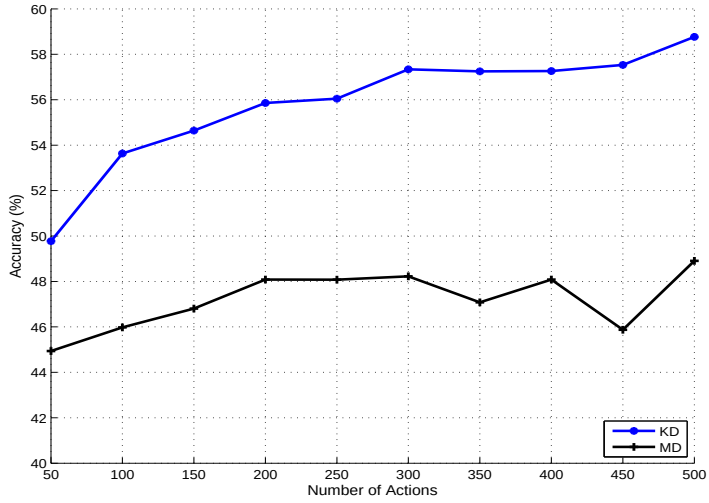


Figure 11.7: *Rank-1* accuracies obtained from the S1 schemes for KD and MD actions.

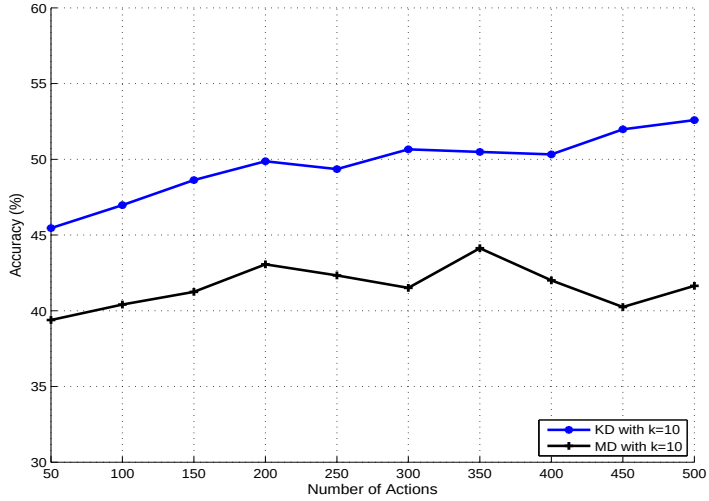


Figure 11.8: *Rank-1* accuracies obtained from the S2 schemes with optimized parameters for KD and MD actions.

perimental protocols followed in this research (see Section 10.3 about the details of these protocols). In our research, all the algorithmic parameters and T_{us} thresholds are optimized using GA.

In Tables 11.1 to 11.7 we see that most of the genuine users fall into the '+ / +' category while some genuine users fall into the '- / +' category for any given analysis technique. But there could be a situation where some genuine users may fall into one of the other categories *i.e.* the '+ / -' or '- / -' category. In that case it will be difficult to compare the advantages/disadvantages of the applied analysis methods. Therefore, we present a summary line of the results for any given analysis. The summary line for all the tables was calculated for S1, S2 and S3 by

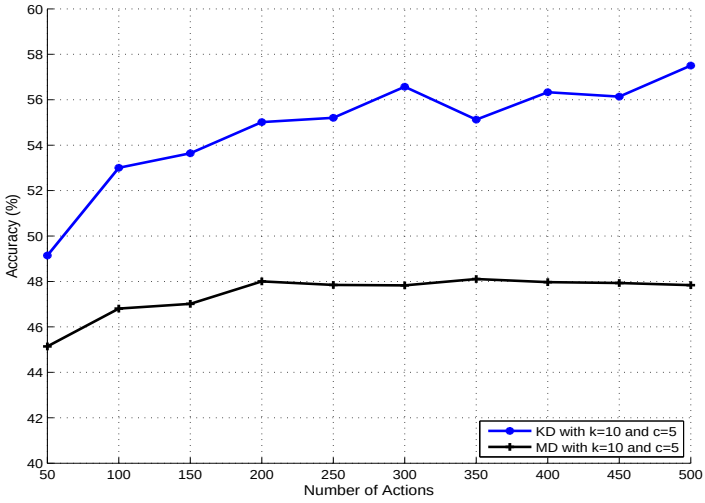


Figure 11.9: *Rank-1* accuracies obtained from the S3 schemes with optimized parameters for KD and MD actions.

$$V = \frac{[u_{++} \times t \times v_{++}] + [(u_{+-} \times t - nd_{+-}) \times v_{+-}] + [u_{-+} \times t \times v_{-+}] + [(u_{--} \times t - nd_{--}) \times v_{--}]}{(N \times t) - (nd_{+-} + nd_{--})}$$

where, u_{++}, u_{+-}, u_{-+} and u_{--} are the number of users that fall into the corresponding categories (*i.e.* '+ / +', '+ / -', '- / +' and '- / -') and v_{++}, v_{+-}, v_{-+} and v_{--} are the value we want to summarize (*i.e.* ANIA or CIS performances) that fall into the corresponding categories. Furthermore nd_{+-} and nd_{--} are the number of imposters not detected for the '+ / -' and '- / -' categories respectively and $t = N - 1$.

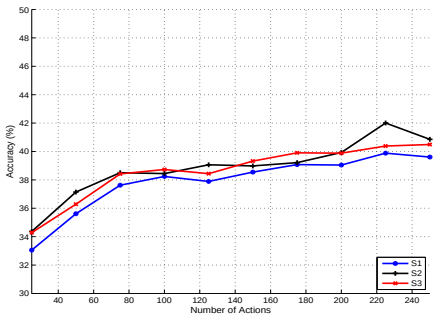
11.3.1 Results Obtained from Average Fusion

In this section we presented the results where we have given the same weight for any actions performed by the user for CIS. In Table 11.1 the system performance for *Protocol-1* that we obtained from CIS for different schemes are shown. In the *CIS Performance (%)* section, columns S1, S2 and S3 show the mean±SD results obtained for S1, S2, and S3 respectively according to the corresponding category. We can clearly see that there is a similar accuracies obtained from S1 and S3 techniques *i.e.* 61.3% and 61.2% respectively.

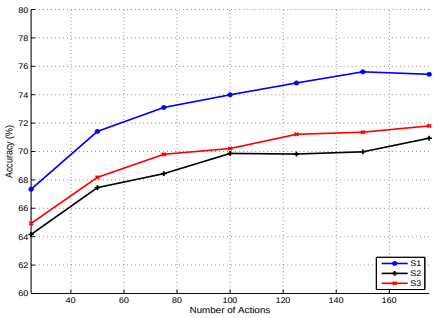
Tables 11.2 and 11.3 show the system performance for *Protocol-2* obtained from S1 and S3 respectively with user's specific T_{open} thresholds (see Section 10.4.2 about T_{open} threshold). The summation of *TID* and *TNotIn* (*i.e. Detection and Identification Rate (DIR)*) is lower than the accuracy of *Protocol-1* for both the schemes (*i.e.* for S1 and S3). We also see that the DIR is slightly higher for S3 when compare to S1 *i.e.* 58% for S1 and 59.2% for S3.

11.3.2 Results Obtained from Weighted Fusion

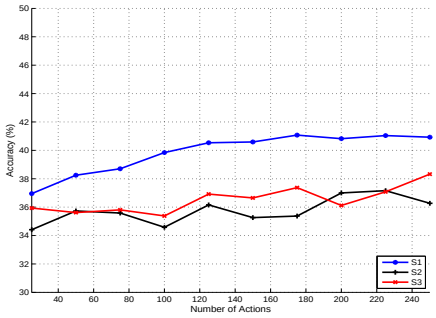
In this section we present the results where we have applied different weights for KD and MD actions performed by the user for CIS. Table 11.4 shows the system performance for *Protocol-1* obtained from this analysis. In this table we can see that there is huge decrement of the CIS performance when compared to the previous analysis (*i.e.* Table 11.1). The reason is that there could be a situation where no or very few KD actions and a comparatively large amount of MD actions is present for



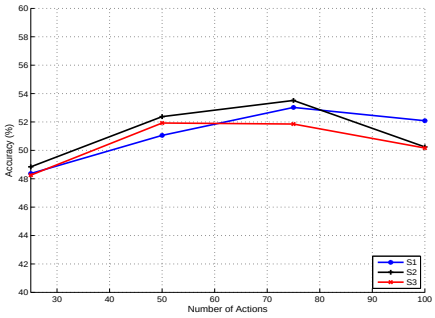
(a) Single Key Action



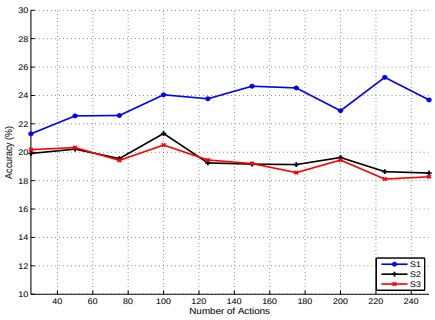
(b) Key Digraph Action



(c) Mouse Single Click Action



(d) Mouse Double Click Action



(e) Mouse Move Action

Figure 11.10: *Rank-1* accuracies obtained from all the schemes with optimized parameters for different KD and MD actions.

identification, or vice versa. Therefore we used a conditional weighted fusion method which is used only when the number of KD and MD actions is rather balanced. In case there is a large difference in the number of KD and MD actions will this method revert back to *Average Fusion*.

$$S = \begin{cases} wt \times \frac{\sum sc^{KD}}{\#KD\ Actions} + (1 - wt) \times \frac{\sum sc^{MD}}{\#MD\ Actions} & if \frac{|\#KD\ Actions - \#MD\ Actions|}{\#Total\ Actions} < T_r \\ Average\ Fusion & otherwise \end{cases}$$

Table 11.1: Result from KD and MD average fusion for *Protocol-1*.

Category	CAS Performance				CIS Performance (%)		
	# User	ANGA	ANIA	Imp. ND	S1	S2	S3
+ / +	22		487		61.2±6.1	59.5±6	61.1±6.2
+ / -							
- / +	3	2020	354		62.4±3	60.9±3.6	62.1±3.1
- / -							
Summary	25	22555	471	0(0%)	61.3	59.7	61.2

Table 11.2: Result from S1 scheme with KD and MD average fusion for *Protocol-2*.

Category	CAS Performance				CIS Performance (%)			
	# User	ANGA	ANIA	Imp. ND	TID	TNotIn	FID	FNotIn
+ / +	24		313		20.3±6.8	37.6±5	14.8±6.2	27.3±7.8
+ / -								
- / +	1	11349	812		24.1	35.2	16.2	24.5
- / -								
Summary	25	24795	333	0(0%)	20.5	37.5	14.9	27.2

Table 11.3: Result from S3 scheme with KD and MD average fusion for *Protocol-2*.

Category	CAS Performance				CIS Performance (%)			
	# User	ANGA	ANIA	Imp. ND	TID	TNotIn	FID	FNotIn
+ / +	24		313		18.2±3.4	41±3.8	10.7±4.7	30.1±4.1
+ / -								
- / +	1	11349	812		17.6	41.4	9.1	31.9
- / -								
Summary	25	24795	333	0(0%)	18.2	41	10.6	30.2

Table 11.4: Result from KD and MD weighted fusion without any condition for *Protocol-1*.

Category	CAS Performance				CIS Performance (%)		
	# User	ANGA	ANIA	Imp. ND	S1	S2	S3
+ / +	22		487		53.3±12.3	49±10.5	54.1±12
+ / -							
- / +	3	2020	354		58.1±5.4	52.9±4.7	59.1±4.9
- / -							
Summary	25	22555	471	0(0%)	53.9	49.5	54.7

This equation is used for the computation of S for the Algorithms 10.1, 10.2, and 10.3. In this equation, we can see that if the ratio between the absolute difference between the number of KD and MD actions and the total number of actions, is less than a given threshold (*i.e.* T_r), then we use the weighted fusion method, otherwise we fall back to the average fusion method (*i.e.* $wt = 0.5$).

Table 11.5 shows the system performance for *Protocol-1* with weighted fusion that we obtained from this analysis for different schemes. We can see that there is an improvement on the identification accuracies for the S1 and S3 schemes when comparing the results with average fusion method, but the accuracies has been decreased for S2 (see Tables 11.1 and 11.5). We also see the improvement on the identification accuracies for all the schemes when comparing the weighted fusion method with and without condition (see Tables 11.4 and 11.5).

Tables 11.6 and 11.7 show the system performance for *Protocol-2* obtained from the S1 and S3 schemes respectively. We obtained these results from our analysis techniques with user specific T_{open} thresholds. For both schemes we can see the improvement of DIR when compared to average

Table 11.5: Result from KD and MD weighted fusion with condition for *Protocol-1*.

Category	CAS Performance				CIS Performance (%)		
	# User	ANGA	ANIA	Imp. ND	S1	S2	S3
+ / +	22		487		62.1±6.4	56.9±6.2	61.3±6.2
+ / -							
- / +	3	2020	354		63.3±3	57.9±3.1	62.9±3
- / -							
Summary	25	22555	471	0(0%)	62.2	57	61.5

Table 11.6: Result from S1 scheme with KD and MD weighted fusion with condition for *Protocol-2*.

Category	CAS Performance				CIS Performance (%)			
	# User	ANGA	ANIA	Imp. ND	TID	TNotIn	FID	FNotIn
+ / +	24		313		20.2±6.4	38.7±4.2	13.7±5.2	27.5±7.2
+ / -								
- / +	1	11349	812		23.8	36.6	14.6	24.9
- / -								
Summary	25	24795	333	0(0%)	20.3	38.6	13.7	27.4

Table 11.7: Result from S3 scheme with KD and MD weighted fusion with condition for *Protocol-2*.

Category	CAS Performance				CIS Performance (%)			
	# User	ANGA	ANIA	Imp. ND	TID	TNotIn	FID	FNotIn
+ / +	24		313		18.1±3.6	41.3±3.2	10.6±4.5	30±4.4
+ / -								
- / +	1	11349	812		17.6	40.8	9.6	32.1
- / -								
Summary	25	24795	333	0(0%)	18.1	41.3	10.6	30.1

fusion method. There is a 0.9% increment of DIR for S1 scheme (see Tables 11.2 and 11.6) and 0.2% increment of DIR for S3 (see Tables 11.3 and 11.7). However, the improvements are within *Standard Deviation (SD)*.

We would also like to mention that the total number of times imposters are detected by the CAS are 16.8×10^4 for *Protocol-1* and 17×10^4 for *Protocol-2*. Each time the CIS was employed, which signifies the presented results in Tables 11.1 to 11.7. As an example, in Table 11.1 we can see that S3 has 1.6% higher accuracy when compare to S2. Meaning that S3 has given $\frac{16.8 \times 10^4 \times 1.6}{100} = 2688$ times more often the correct identity when compared to S2.

11.4 Summary

The major findings from this research are follows:

- The concept of *Continuous Identification* has been evaluated in this chapter for a PC. CIS in combination with CAS will not only protect a system against unauthorized access but will also try to establish the identity of the imposter. We obtained identification accuracy of 62.2% for the closed-set experiment (*i.e. Protocol-1*) and a DIR of 59.4% for the open-set experiment (*i.e. Protocol-2*).
- A combination of keystroke and mouse dynamics has been used in this analysis. Therefore, six different types of actions were taken into consideration which makes this analysis process highly time consuming.
- We achieved some improvement in the weighted fusion method when compared to the average fusion method for KD and MD actions. This approach can also be extended even for different

types of KD and MD actions (*i.e. Single Key Actions*, *Key Digraph Actions*, *Mouse Move Actions*, *Single click Actions*, *Double click Actions* and *Drag-Drop Actions*).

- This analysis provide us the proof of concept about *Continuous Identification*, which motivated us to apply this research on other datasets.

Chapter 12

Continuous Identification on Mobile Devices

In this chapter, we investigated the performance of a swipe gesture based continuous identification for mobile devices under various analysis techniques. We have used two separate publicly available swipe gesture based datasets for our analysis *i.e.* `Set-1` of *Dataset-3* and *Dataset-4*. The complete description of these datasets and the extracted features are given in the Section 5.3.

This chapter is based on the papers published in: [97] MONDAL, S., AND BOURS, P. Continuous authentication and identification for mobile devices: Combining security and forensics. In *7th IEEE Int. Workshop on Information Forensics and Security (WIFS'15)* (2015), IEEE, pp. 1–6 and [102] MONDAL, S., AND BOURS, P. Continuous user authentication and adversary identification: Combining security & forensics. Under Review in IEEE Transactions on Information Forensics & Security, 2016.

12.1 Profile Creation

In this section we describe the profile creation process for CAS and CIS (see Section 10.1 for the description of CAS and CIS). The complete CAS profile creation process was describe in the Chapter 9 for both the datasets.

We have explored three different classifiers for CIS. These classifiers are ANN, CPANN and SVM. We have also used MCF to obtain a better performance than for a single classifier [75]. For each combination of genuine user i and imposter user j we created a training set CIS_j^i. This classifier was trained with the training data M_i of user i and the training data M_j of user j, where, to avoid bias, the amount of training data of both users is taken equal. For example, we have $N = 41$ users for *Dataset-2*, so we have $N \times (N-1) = 1640$ different CIS classifier models for any given classifier (*i.e.* either ANN, CPANN or SVM) for this dataset.

Before building the classifier models, we first apply the feature selection technique for both CAS and CIS as describe in Section 8.3. Figures 12.1 and 12.2 show the Empirical Cumulative Distribution Function plot of the selected features for CAS^8 and CIS_1^8 respectively for *Dataset-3*. We can see that the selected features are different for CIS and CAS. We observed that different sets of features were selected for different combination of users (*i.e.* CIS_j^i) and also we have observed that the number of selected features varies for CAS and a pairwise combination of CIS.

We also tested the feature selection technique proposed by Ververidis *et al*. [148] for both datasets. We found that the feature selection technique describe in Section 8.3.2 worked better for *Dataset-3* and the technique proposed by Ververidis *et al*. worked better for *Dataset-4*. We decided to apply the feature selection technique according to this observation.

12.2 Periodic Analysis of the Identification Schemes

In this section, we will show the performance of the proposed identification schemes discussed in the Section 10.2 under various circumstances. All the analysis presented in this section was done on *Dataset-3* with a fixed length of data blocks (*i.e.* $m = 2, 4, \ldots 20$). Note that this analysis does not represent the CIS performance. This analysis was done to understand how our identification schemes perform under various parameters and circumstances. The CA-CI performance results can be found in Section 12.3.

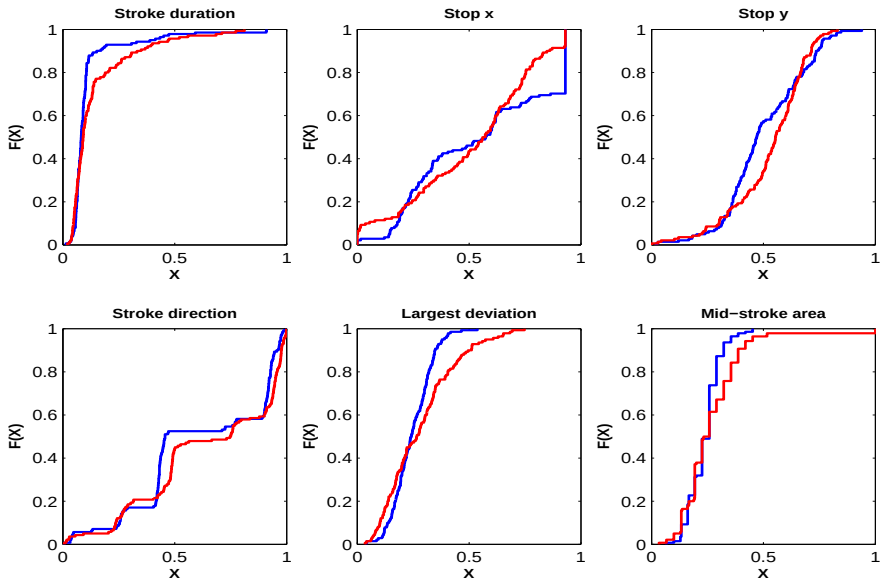


Figure 12.1: Cumulative Distribution of selected features for CAS^8 of *Dataset-3*. The blue CDF are generated from the data of user 8 and the red CDF are generated from the data of the imposters for this user.

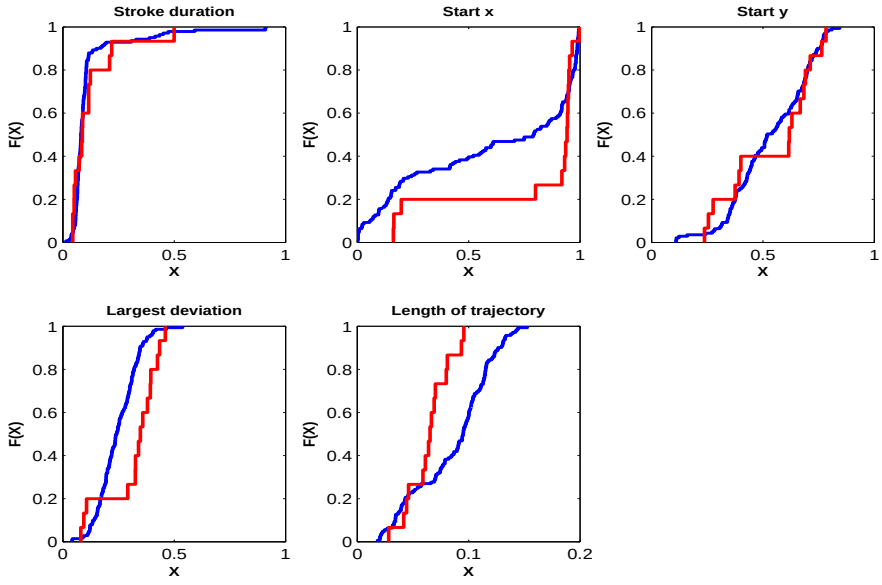


Figure 12.2: Cumulative Distribution of selected features for CIS_1^8 of *Dataset-3*. The blue CDF are generated from the data of user 8 and the red CDF are generated from the data of the user 1.

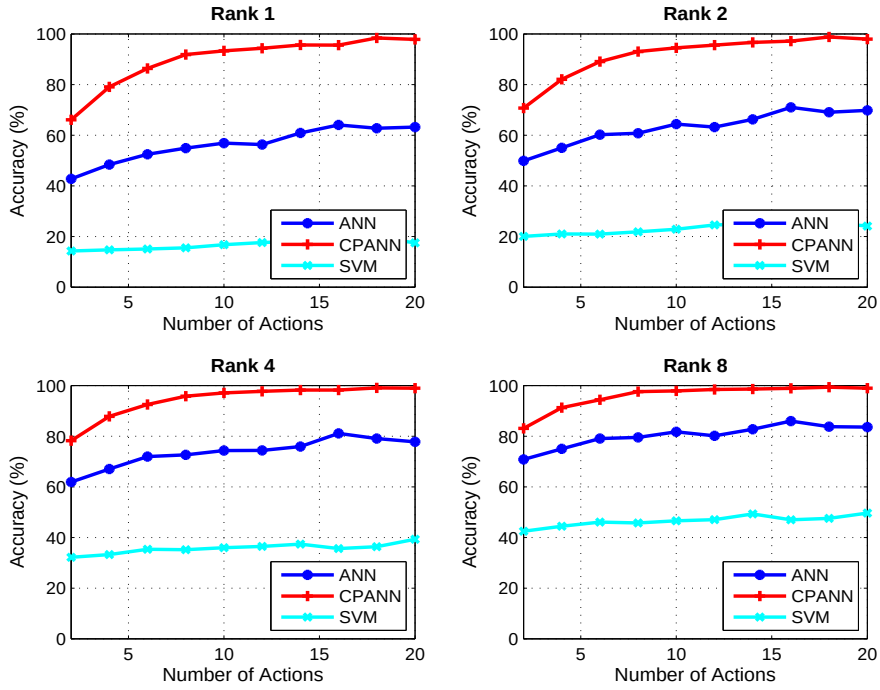


Figure 12.3: Results obtained from S1 with different classifier.

Let, λ_p, ρ_p and τ_p denote the score values for ANN, CPANN and SVM respectively calculated from CIS_j^i for test data Γ, where $p = 1, 2, \ldots m$. Furthermore, let sc denote the score vector, where $sc = \lambda$, $sc = \rho$ or $sc = \tau$ depend upon the classifier choice (*i.e.* either ANN, CPANN or SVM).

12.2.1 Analysis with Scheme 1

Figure 12.3 shows the identification accuracies obtained from S1 (see Algorithm 10.1) with different ranks (*i.e.* Rank-1, Rank-2, Rank-4 and Rank-8) for various numbers of actions $m = 2, 4, \ldots 20$. We can clearly see that the performance of CPANN is better than ANN and SVM. We also have noted that increasing the number of test actions does not always increase the identification accuracies.

We have also tested the S1 with MCF for various numbers of actions $m = 2, 4, \ldots 20$, for three different classifier combinations *i.e.* CPANN-ANN ($sc_p = W_{ci} \times \rho_p + (1 - W_{ci}) \times \lambda_p$), SVM-ANN ($sc_p = W_{ci} \times \tau_p + (1 - W_{ci}) \times \lambda_p$) and CPANN-SVM ($sc_p = W_{ci} \times \tau_p + (1 - W_{ci}) \times \rho_p$), where W_{ci} is the weight for weighted fusion MCF technique and $0 \leq W_{ci} \leq 1$. Figure 12.4 shows the obtained identification accuracies for this analysis with different ranks (*i.e.* Rank-1, Rank-2, Rank-4 and Rank-8) for various numbers of actions $m = 2, 4, \ldots 20$. We can observe that the performance of the MCF did not improve relative to the single CPANN classifier. Therefore, we decided to use only the CPANN classifier for further analysis.

12.2.2 Analysis with Scheme 2

Figure 12.5 shows the system identification accuracies obtained from S2 (see Algorithm 10.2) with different ranks (*i.e.* Rank-1, Rank-2, Rank-4 and Rank-8) for various numbers of actions $m = 2, 4, \ldots 20$, with different k value (*i.e.* $k = 5, 10, \ldots 25$). We found that there is a large difference

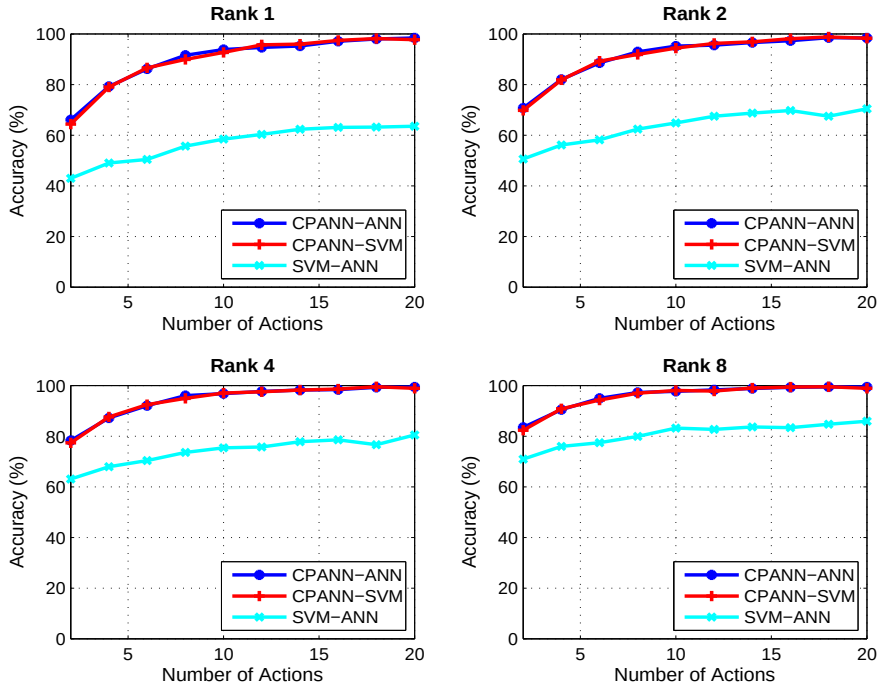


Figure 12.4: Results obtained from S1 with MCF.

in accuracy between *Rank-1* and *Rank-8* for any given k and m value, which motivated us to use the scheme as described in Section 10.2.4.

Contrary to the randomly chosen k number of pairs from CIS^i, we applied the scheme to a fixed set of k pairs, where the pairs are selected based on the maximum learning accuracy. Figure 12.6 shows the comparison of the identification accuracies randomly chosen pairs and fixed pairs for $k = 15$ with different number of actions $m = 2, 4, \ldots 20$. We can see that randomly chosen pairs perform better than fixed pairs for any given rank and any given m value. Due to the nature of behavioural biometrics higher learning accuracy of the classifier models that is based on the training set, does not always give the best results in the unknown test set. Therefore, we have decided to use randomly chosen pairs for further analysis.

12.2.3 Analysis with Scheme 3

Figure 12.7 shows the result we obtained after applying S3 (see Algorithm 10.3) for different number of actions (*i.e.* $m = 2, 4, \ldots 20$) with different k value (*i.e.* $k = 5, 10, \ldots 25$) and $c = 8$. We can see the improvement on the results for *Rank-1*, *Rank-2*, and *Rank-4* by comparing Figures 12.5 and 12.7. The *Rank-8* classification accuracy will be the same for both S2 and S3 (because S3 was derived from same *Rank-8* users of S2). We can see that increasing the value of k will increase the recognition accuracy, but it will saturate after a certain value of k. Therefore, we used $k = 15$ in the remainder of the analysis.

Figure 12.8 shows the results obtained after applying S3 for different numbers of actions (*i.e.* $m = 2, 4, \ldots 20$) with different c value (*i.e.* $c = 8, 10, \ldots 16$) and $k = 15$. We can see that increasing the value of c will not increase the recognition accuracy (*i.e. Rank-1* accuracy). Therefore, we have decided to use $c = 8$ for further analysis.

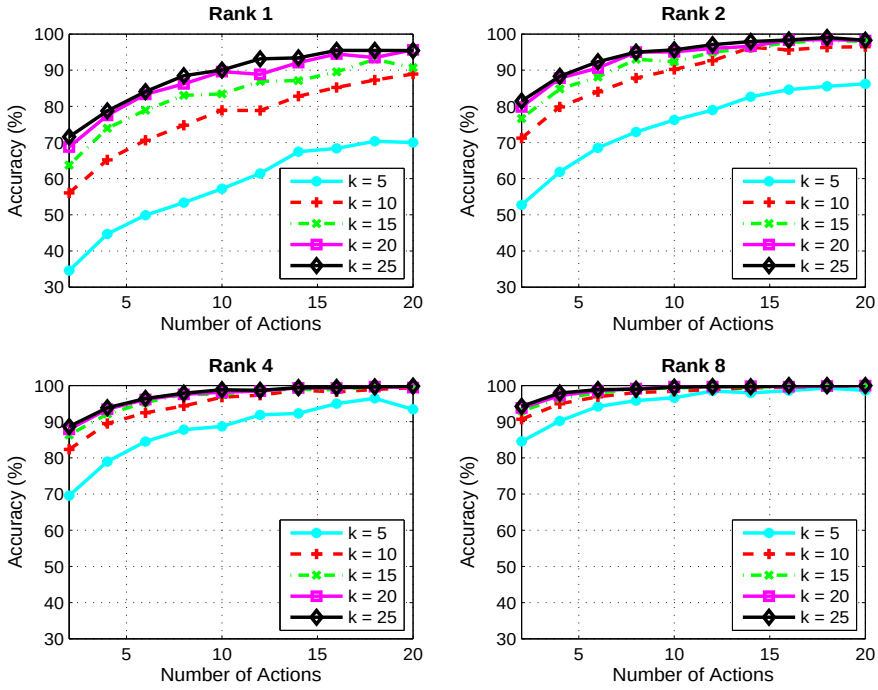


Figure 12.5: Results obtained from S2 for different k value.

12.2.4 Discussion

In this section, we will discuss some observations made during the analysis.

- Note that all the analysis presented in this section is performed on *Dataset-3* and we found that the CPANN classifier outperforms the other single classifiers as well as MCF. When we applied these schemes on *Dataset-4* we found that the MCF of CPANN-SVM performed best. We applied the techniques in our final analysis presented in Section 12.3 according to these findings.

- From the above analysis, we can see that identification performance of S3 is slightly better than S1 at the cost of more pairwise comparisons needed to perform the analysis (*i.e.* $T_3 \geqslant T_1$). Also, the performance of S1 is better than S2, even though the number of pairwise comparisons is less (*i.e.* $T_1 \leqslant T_2$).

12.3 Result Analysis

We report the results with the user specific lockout threshold (T_{us}), where the threshold for lockout will be $T_{us} = \max(50, \min(Trust_{genuine}))$ and with a fixed lockout threshold, where $T_{lockout} = 90$ (see Section 10.1 for more details about lockout threshold). We also report our results based on two different experimental protocols followed in this research (see Section 10.3 about the details of these protocols). In our research, all the algorithmic parameters and T_{us} thresholds are optimized using linear search.

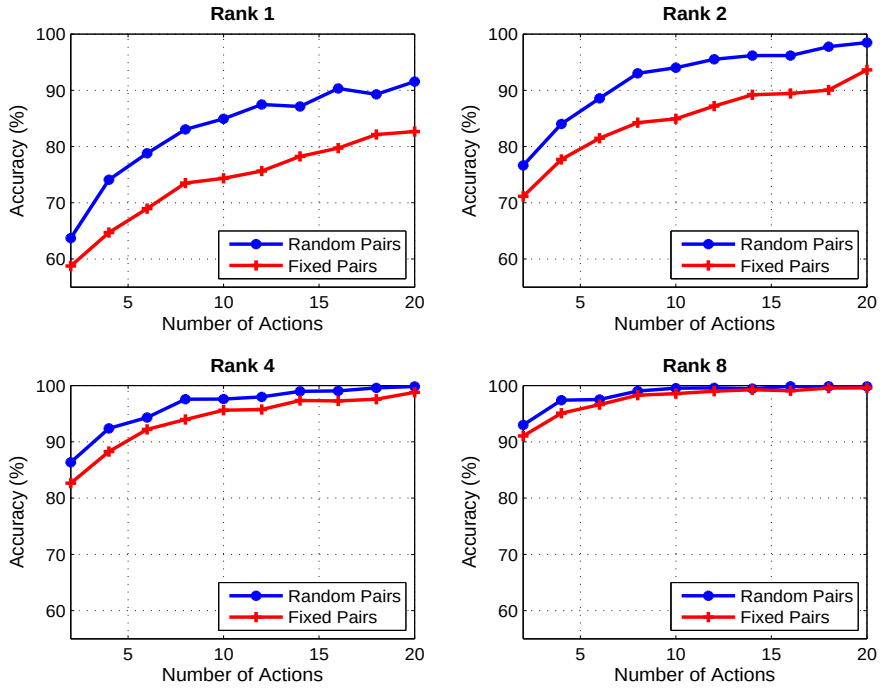


Figure 12.6: Results obtained from S2 with $k = 15$ for random and fixed pairs.

12.3.1 Results for Protocol 1

In Table 12.1 the system performance for *Database-3* and *Protocol-1* that we obtained from CIS for different analysis techniques are shown. In the *CIS Performance (%)* section, columns S1, S2 and S3 show the mean±SD results obtained from *S1*, *S2* and *S3* respectively according to the corresponding category. We can clearly see that the S3 technique performs better than the other techniques. For the '+ / +' category, the number of imposter actions is lower than for the '+ / -' category. This implies that the CIS has fewer data available for identifying the imposter, which obviously has a negative impact on the CIS performance.

Table 12.2 shows the system performance for *Database-4* and *Protocol-1* that we obtained from CIS for different analysis techniques. Similar to previous table we can also observe that the S3 technique performed better than the other techniques even though the difference with S1 is not as significant. We have decided to use S3 for the *Protocol-2* analysis that will be presented in the next section. We can also see that ANIA value is a bit higher than in the previous table for any given lockout threshold, but on the other hand all the users are within the '+ / +' category for T_{us}.

12.3.2 Results for Protocol 2

Tables 12.3 and 12.4 present the system performance for *Protocol-2* for *Dataset-3* and *Dataset-4* respectively. We obtained these results from our analysis techniques with user specific T_{open} thresholds. In Table 12.3 we can see that 66 users qualified in the '+ / +' category with an ANIA of 4 actions and the rest of the users qualified as '+ / -' category with ANIA of 15 actions for T_{us}. In case of CIS performance the summation of *TID* and *TNotIn* (*i.e. Detection and Identification Rate (DIR)*) is 74.8% which is lower than the accuracy of *Protocol-1*. Similar to previous observations,

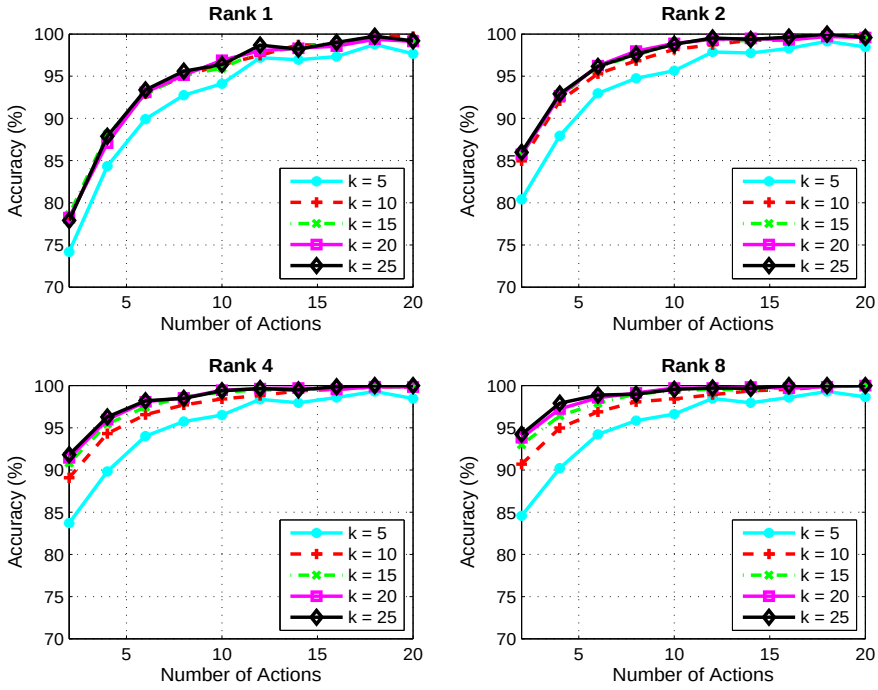


Figure 12.7: Results obtained from S3 with $c = 8$ for different k value.

Table 12.1: Result obtained from our analysis methods for *Database-3* and *Protocol-1*.

$T_{lockout}$	Category	CAS Performance				CIS Performance (%)		
		# User	ANGA	ANIA	Imp. ND	S1	S2	S3
T_{us}	+ / +	68		4 ± 2		72.9 ± 6.8	69 ± 5.2	**82.3 ± 5.4**
	+ / -	3		14 ± 3	4	88.3 ± 3.8	81.8 ± 1.7	**93 ± 2.4**
	- / +							
	- / -							
	Summary	71		5	4(0.1%)	73.5	69.5	**82.7**
90	+ / +	63		14 ± 3		94.5 ± 1	87.3 ± 1.4	**97.8 ± 0.6**
	+ / -	8		25 ± 7	20	96.4 ± 1.2	90.8 ± 2.4	**98.5 ± 0.6**
	- / +							
	- / -							
	Summary	71		16	20(0.4%)	94.7	87.7	**97.9**

the CIS performance can improve if we use Tr_{90} as a lockout threshold because of the higher ANIA value. Similar observations can also be made in the Table 12.4.

In Figure 12.9 the system performance with *Protocol-2* that we obtained from our analysis techniques are shown for different T_{open} threshold (*i.e.* $T_{open} = 0.5, 0.6, \ldots 0.9$). Figure 12.9(a) shows the results obtained from *Dataset-3* and Figure 12.9(b) shows the results obtained from *Dataset-4*. In both figures, the first stack for each T_{open} value is generated from the summary results of Tr_{90} and the second stack was generated from the summary results of Tr_{us}.

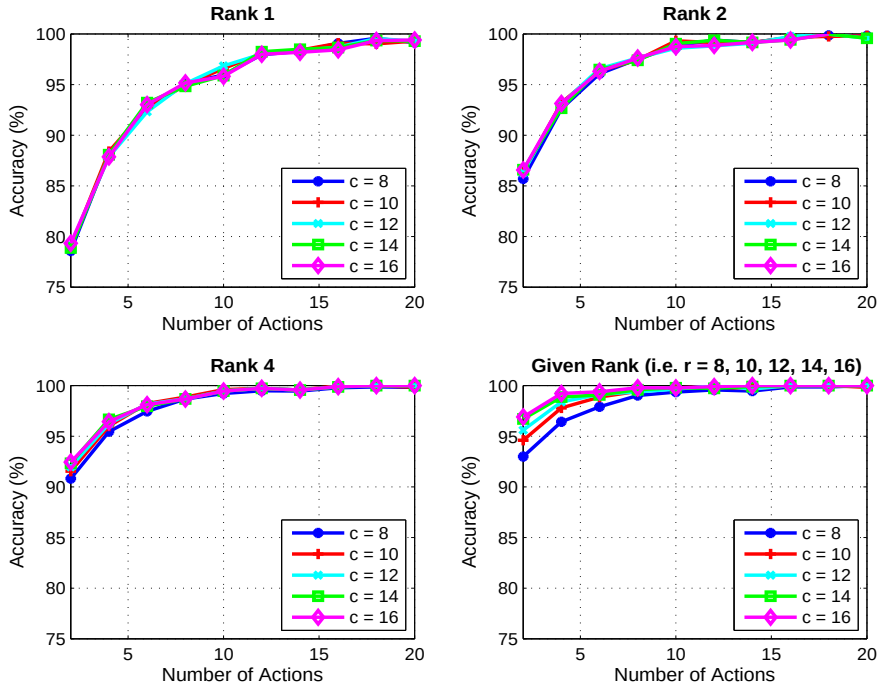


Figure 12.8: Results obtained from S3 with $k = 15$ for different c value.

Table 12.2: Result obtained from our analysis methods for *Database-4* and *Protocol-1*.

$T_{lockout}$	Category	CAS Performance				CIS Performance (%)		
		# User	ANGA	ANIA	Imp. ND	S1	S2	S3
T_{us}	+ / +	41		11 ± 9		79.7 ± 5.3	69.6 ± 5.1	**82.2 ± 4.9**
	+ / -							
	- / +							
	- / -							
	Summary	41		11	0(0%)	79.7	69.6	**82.2**
90	+ / +	39		27 ± 16		84.8 ± 1.8	72.9 ± 2.3	**86.4 ± 1.7**
	+ / -	2		246 ± 141	4	95.2 ± 4.8	88.7 ± 9.3	**95.5 ± 4.5**
	- / +							
	- / -							
	Summary	41		37	4(0.2%)	85.3	73.6	**86.8**

12.3.3 Comparison with Previous Research

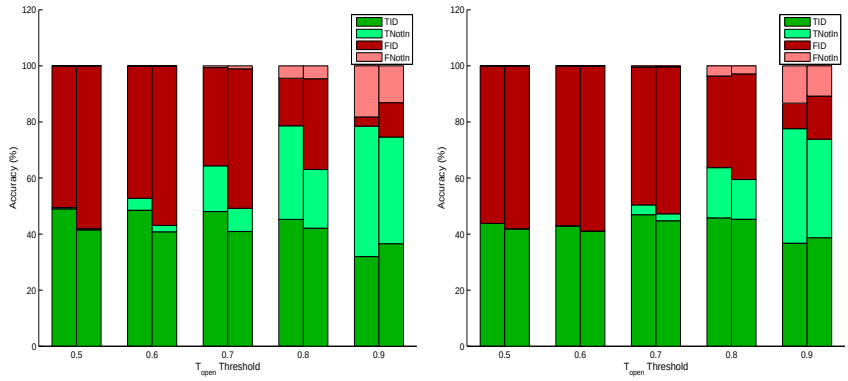
Due to the novelty of this research we did not find any research which is directly related to our research. Therefore, we are unable to compare our continuous identification results with previous results, but, we can compare our proposed identification technique with the previous research done on the same dataset *i.e. Dataset-3*. Figure 12.10 shows the identification accuracy comparison with previous research done by Antal *et al.* [7] with their best technique for different numbers of actions. In research [7] best result was found by using the SVM classifier without pairwise user coupling. We would like to mention that in [7] the amount of data used for classifier training and testing the system is not clearly mentioned. We see that the proposed method outperforms the existing research. More

Table 12.3: Result obtained from our S3 analysis method for *Database-3* and *Protocol-2*.

$T_{lockout}$	Category	CAS Performance				CIS Performance (%)			
		# User	ANGA	ANIA	Imp. ND	FID	FNotIn	TID	TNotIn
T_{us}	+ / +	66		4 ± 2		12.2 ± 4.2	13.2 ± 2.3	36.5 ± 2.2	38.1 ± 4
	+ / -	5		15 ± 8	10	13 ± 2.6	9.8 ± 4	40.8 ± 4.2	36.4 ± 2.6
	- / +								
	- / -								
	Summary	71		5	10(0.2%)	12.3	13	36.8	38
90	+ / +	56		15 ± 4		8 ± 6.3	12.7 ± 6.6	37.1 ± 6.3	42.2 ± 6.1
	+ / -	15		22 ± 8	27	11.7 ± 6	7.5 ± 6.4	43.2 ± 6.7	37.6 ± 6.2
	- / +								
	- / -								
	Summary	71		17	27(0.5%)	8.8	11.6	38.4	41.2

Table 12.4: Result obtained from our S3 analysis method for *Database-4* and *Protocol-2*.

$T_{lockout}$	Category	CAS Performance				CIS Performance (%)			
		# User	ANGA	ANIA	Imp. ND	FID	FNotIn	TID	TNotIn
T_{us}	+ / +	40		17 ± 13		15.5 ± 5.8	10.9 ± 2.6	38.6 ± 2.5	35 ± 5.5
	+ / -	1		88	1	8.8	6.7	44.1	40.5
	- / +								
	- / -								
	Summary	41		19	1(0.1%)	15.3	10.8	38.7	35.1
90	+ / +	38		32 ± 20		9 ± 1.7	13.7 ± 1.8	36 ± 1.8	41.2 ± 1.7
	+ / -	3		138 ± 80	9	9.3 ± 2.6	7.6 ± 3.3	46.5 ± 4.6	36.6 ± 1.9
	- / +								
	- / -								
	Summary	41		39	9(0.6%)	9	13.3	36.7	40.9



(a) Results for *Dataset-3* (b) Results for *Dataset-4*

Figure 12.9: System performance for different T_{open} threshold (*i.e.* $T_{open} = 0.5, 0.6, \ldots 0.9$). In both the figures first stack was generated from the summary results of Tr_{90} and second stack was generated from the summary results of T_{us}.

specifically, there is a large difference in accuracy when using a small number of actions, which benefited our research when the imposter users is locked out after a few numbers of actions.

12.3.4 Discussion

In this section, we will discuss some of the major findings from this study.

- Contrary to the state of the art CA research on mobile devices (*i.e.* PA), we have used an actual

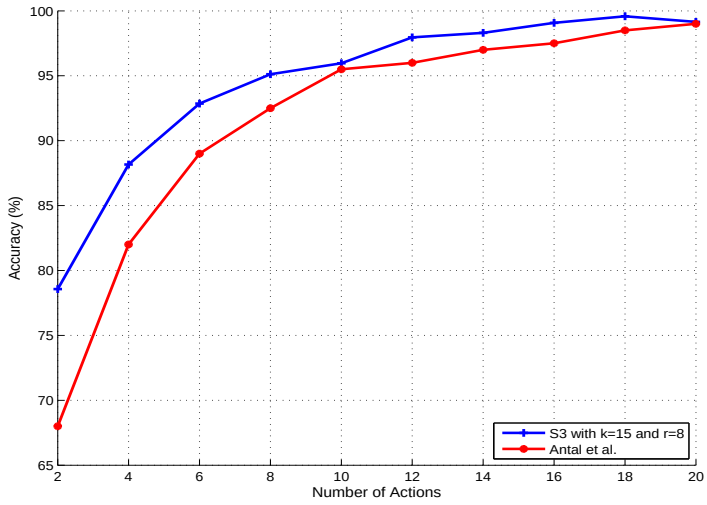


Figure 12.10: Identification accuracy comparison with previous research on the *Dataset-3* .

CA system in our research. The advantage of the applied CAS is that, whenever the system is confident about the illegitimacy of the current user, it does not wait to complete the block of a fixed number of actions before locking out the user. From a classification point of view, this poses a huge challenge, because the classifier only learns in terms of actions basis not a chunk basis. We overcome this problem by applying the trust model.

- One of the challenges that we had in this research is that the number of the data samples that are used for the CIS is variable. The number of data samples equals the number of actions that the current user could perform before he/she was locked out by the CAS module. We mitigate this problem by our proposed identification schemes.

- Due to large intra-class variation, small inter-class variation and limited information of behavioural biometrics (*i.e.* swipe gesture), performing authentication (*i.e.* 1:1 comparison) is a challenging task. On top of that, we are performing identification (*i.e.* 1:N comparison), which increases this challenge even further. We overcome these challenges with high degree of confidence by our proposed identification techniques. These techniques are general enough that they can be applied to other identification problems.

- We have observed that none of the genuine users are wrongly locked out from the system for any give lockout threshold (*i.e.* Tr_{us} or Tr_{90}), dataset and protocol. This means that our system will provide a high degree of user friendliness. In Table 12.5 the total number of time imposters are detected by the CAS, *i.e.* the number of times CIS was run after lockout by CAS. These numbers show the significance of the presented results in Tables 12.1, 12.2, 12.3, and 12.4.

12.4 Summary

The summary of the major findings from this chapter is as follows:

- The concept of continuous identification has been evaluated in this chapter for mobile devices. CI in combination with CA will not only protect a system against unauthorized access but will

Table 12.5: Total number of times imposters detected by CAS, where $N = 71$ for *Dataset-3* and $N = 41$ *Dataset-4*.

$T_{lockout}$	Protocol	*Dataset-3*	*Dataset-4*
T_{us}	*Protocol-1*	$4778 \times N$	$4177 \times N$
	Protocol-2	$4755 \times N$	$4048 \times N$
Tr_{90}	*Protocol-1*	$1039 \times N$	$1112 \times N$
	Protocol-2	$1004 \times N$	$992 \times N$

also with high probability identify the imposter. This might then be an additional barrier for an imposter when considering illegal access on an other person's system.

- We have shown that for an optimal CA system (*i.e.* for T_{us}) the imposter is identified with a more than 82% accuracy for both datasets. In a slightly less optimal CA system (that would allow an imposter some more activity before lockout *i.e.* for Tr_{90}), the correct identification rate will be much higher *i.e.* 98% for *Dataset-3* and 87% for *Dataset-4*.
- We have evaluated the system in a closed-set *e.g.* a work environment (*i.e.* all the imposters are known to the system), but also in an open-set (*i.e.* 50% of the imposters are unknown to the system), where some of the potential adversaries are known, but some are unknown. For the open-set we found that the probability of a correct decision by the CIS (either correctly identify the imposter or correctly determining it was none of the known persons *i.e.* DIR) was also almost 80% for both the datasets.

Part IV

Conclusion

Chapter 13

Summary of Findings

Our research is divided into two major parts *i.e. Continuous Authentication (CA)* (see Part II) and *Continuous Identification (CI)* (see Part III). The summary of the major findings of this research will be discussed below.

13.1 Continuous Authentication

The objective of a CA system is person re-authentication after initial session login (*i.e. Static Login* with a password or any biometric modalities) to protect the device from session hijacking. The CA problem statement was introduced to the research community in 1995, but according to our observation, most of the state of the art CA research was performed in a periodic manner (*i.e.* periodic authentication). This means that there is fixed interval (fixed number of actions or fixed length of time) after which the system will check the authenticity of the user. Therefore, it provides room for the imposter user to perform at least that fixed number of actions or control the device at least that fixed time period, because the system is not performing the re-authentication task within this period.

In our research, we redefine this problem where every action performed by the user will be checked immediately and can lead to a decision where the current user can continue to work on the device based on the system confidence in the genuineness of the present user or will be locked out. A detailed description of the proposed CA approach can be found in Part II. The summary of the findings is given below.

- We have shown that in a CA system every single action by the current user will influence the decision on the genuineness of the current user and that a decision to lockout the user or not can be made at each action. We have developed a robust trust model algorithm that can be used for a CA system, irrespective of the biometric modality (see Chapter 3). We found that the *Dynamic Trust Model (DTM)* performs best on every dataset used in this research.

- For a CA system, it is important to know that an imposter is detected by the system but, it is more important to know how much activity that imposter can perform before getting detected by the system. Therefore, we have described *Average Number of Imposter Actions (ANIA)* and *Average Number of Genuine Actions (ANGA)* as a performance reporting metric for a CA system comparable to FMR and FNMR for an SA or PA system (see Chapter 4). We also show that how FMR and FNMR can be converted to ANIA and ANGA for inter-system performance comparison of CA systems.

- We have validated our CA approach by using four different datasets where three of them are publicly available and one is collected by ourself. All of these datasets are behavioural biometrics based datasets. Two of these datasets are collected from PCs where the biometric modalities are keystroke and mouse dynamics and the other two datasets are collected from mobile devices where the biometric modality is swipe gestures. The detailed description of these datasets and extracted features from the raw data can be found in Chapter 5

- We have also validated our approach by using three different verification protocols (see Section 4.2.2) for all the datasets and shown that our approach outperforms state of the art approaches. The detailed results can be found in Chapters 6 to 9. Table 13.1 shows a summary of best results obtained for different modalities.

Table 13.1: Best results obtained for continuous authentication for different modalities.

Chapter	Modality	Dataset	Summary Result			
			# User	ANGA	ANIA	Imp. ND
Chapter 6	MD	Dataset-1	49		70	0%
Chapter 7	KD	Dataset-2	53	16057	499	1.30%
Chapter 8	KD & MD	Dataset-2	53	23996	252	0.10%
Chapter 9	Swipe Gestures	Dataset-3 (Set -1)	71		4	0.10%
		Dataset-3 (Set -2)	51		5	0%
		Dataset-4	41		11	0%

- We have developed the *Score Boost (SB)* algorithm (see Section 6.2.1) to reduce the overlap between imposters and genuine user scores. We also developed a classifier *Weighted Fusion* algorithm (see Section 6.2.2), where this algorithm finds the weight for classifier fusion based on the learning accuracies of the used classifiers. Furthermore, we developed a *Feature Selection* algorithm (see Section 8.3.2) based on the maximization of the separation between two *Multivariate Cumulative Distributions*. In this algorithm, we applied the evolutionary algorithm, more precisely *Genetic Algorithm (GA)*, as a feature searching technique. These algorithms can be applied to other pattern recognition problems.

13.2 Continuous Identification

The concept of CI has been introduced for the first time in the research community. The combination of continuous authentication and identification will provide a robust system that not only protects the device from an adversary, but also aims to establish the identity of the adversary that could be used as a forensic evidence, which is the main motivation behind CI. The CI related chapters can be found in Part III. The summary of the findings is given below.

- The approach followed for CI in combination with CA was discussed in Chapter 10. We have evaluated the system with three different datasets in a closed environment like a work environment (*i.e.* closed-set where all the imposters are known to the system), but also in a open environment (*i.e.* open-set where 50% of the imposters are unknown to the system), where some of the potential adversaries are known, but some are unknown.

- Due to large intra-class variation, small inter-class variation and limited information on behavioural biometrics, performing authentication (*i.e.* two class problem) is a challenging task, and on top of that, we are performing identification (*i.e.* N class problem) which increases this challenge even further. To overcome these challenges we developed three different identification schemes by using a pairwise user coupling (see Section 10.2). But, these approaches can be applied to any pattern identification problem.

- We first evaluated the CI system on our own dataset (*i.e.* combined keystroke and mouse dynamic data) and found a satisfactory result which provided proof of concept. We obtained an identification accuracy of 62.2% for the closed-set experiment and a DIR of 59.4% for the open-set experiment. The detail analysis and achieved results can be found in Chapter 11.

- We have also performed CI experiments on two swipe gesture based datasets and showed that in a closed environment with a strict CA system, the identification accuracy is already at a high level (*i.e.* accuracy of 82%), but a slightly less strict CA system will get an even better identification rate (*i.e.* accuracy of 98%). In an open environment, we found that the accuracy of the CIS (either correctly identifying the imposter or correctly determining it was none of the known persons *i.e.* DIR) was almost 80% for both swipe gesture based datasets (see Chapter 12).

13.3 Dataset

We found that all the state of the art experiments on CA that combine keystroke and mouse dynamics were done under controlled lab environments, where participants had to complete a predefined task or use a set of predefined applications. Based on our understanding, these types of data collection processes do not represent the participant's natural behaviour, because participants are highly focused on completing the given task. These experiments are the starting point to understand the science behind CA but not enough to extend it to a real world situation. Therefore, we build a dataset that represents the real world continuous user's PC interaction data through keystroke and mouse. Our dataset consists of both keystroke and mouse information, hardware usage information and includes user application information (*i.e. Software Interaction (SI)*). The complete description of this dataset can be found in Section 5.2.

Due to a high degree of privacy concern, we are unable to share this dataset publicly. But, we can share our data collection software with other researchers to build their own dataset for their experiments. The details of this data collection software can be found in Appendix A.

Chapter 14

Future Work

We divided the future work into two parts. In Section 14.1 we discuss two major issues that each by itself could constitute a full PhD research. In Section 14.2 we will describe some smaller issues that are not addressed in this thesis.

14.1 Major Issues

14.1.1 Template Update

Behavioural biometrics is generated from human motor skill activities. Therefore, it is considered that the used biometric modalities (*e.g.* keystroke dynamics, mouse dynamics and swipe gestures) are not permanent. These will change over time due to the increased experience of using these modalities or simple ageing of the user. This means that if someone starts using a smart phone today and his/her swipe gestures will be slightly different after say two months. It will also change after a sudden change of gadgets (*i.e.* change of keyboard, mouse or phone). However, it will normalize when users get used to these new gadgets. Therefore, a strong template update mechanism is required to build a robust continuous authentication and identification system [54]. To perform this research, there is also a strong need to have a controlled user group performing an experiment over a longer period of time, *e.g.* at least one year. Below is the list of ideas that could be the starting point to address this issue:

- One approach could be a periodic update of the template for the genuine users, *e.g.* update user's template after 2-3 months of usage. This is a straight forward approach and it can be implemented very easily. In this process, we only need to make sure that the genuine user is really using his/her device when the system is recording the raw data for the template update. Therefore, we can make sure that users go through the static login session just before this raw data capture process starts and users can be warned not to leave their device unattended or unlocked during this time.
- Another approach could be an automatic update of the template for the genuine users when the system notices the need to update the genuine user's template. If a genuine user gets locked out from the system frequently, then there is a strong need to update that user's template. One way could be to update the template information when system trust stays above some high level for a predefined large number of actions. We believe that there could be many possibilities to explore and more research is needed in this domain.

From a pattern recognition point of view updating a template for distance based classifiers is not a complex process (add new information and remove old information from the template). But, there is a common challenge from a machine learning perspective for both the above approaches. We used different classifiers for user pattern classification, therefore updating the template means to retrain all the classifiers. So, an interesting research topic is whether there is a possibility to update the user's template without completely retraining the entire classifiers.

14.1.2 Active Attack Scenario

In our research, we followed a *"zero-effort imposter attack scenario"*, but it is important to know the system performance in an *'active attack scenario'*, where the attacker tries to actively mimic the

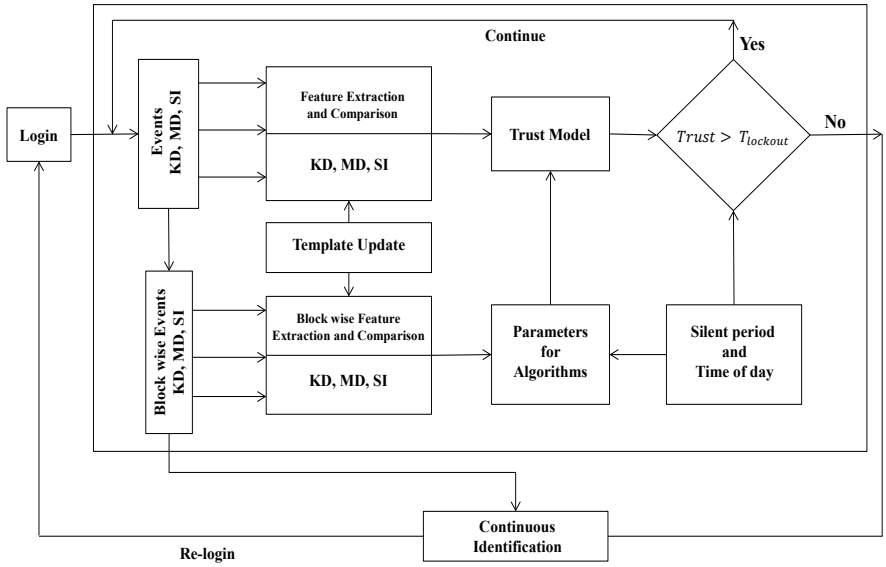


Figure 14.1: Complete system architecture.

victim's behaviour. According to our understanding, mimicking a victim's behaviour continuously is a difficult task. Therefore, designing an experiment to measure the possibility for active mimicking will be complex and will require much effort and time of the participants. Below is the list of ideas that could be the starting point to address this issue:

- One approach could be a direct approach, where we can create an attacking group and train them to behave like a given victim. In this process, we can train the attackers by showing them the victim's behaviour profile or by showing a video recording of the victim's PC/Mobile usage behaviour.

- Another approach could be the indirect approach, where we can use the *Central Pattern Generator (CPG)* to model the human hand movement and generate raw data of a victim's behaviour for an experiment [29]. This approach is not similar to the indirect software/hardware attack where researchers have injected software into the system that is able to bypass the CA/CI system. Sometimes attackers use a hardware device between a PC and a keyboard/mouse that generates the victim's pattern for any given task performed by the attacker [123]. CPG can also be used to generate a victim's like behaviour for any given task performed by the attacker. Therefore, the challenge will be the distinction between a human attack and a machine attack.

14.2 Minor Issues

To make the CA and CI system more robust, we proposed the system architecture according to Figure 14.1, where template update has been taken into consideration. Some of the following interesting issues need to be researched to improve our current system performance:

- In KD, we have followed the user's global keystroke rhythms for missing digraphs and monographs. We feel that alternate approximation technique for the missing digraphs and monographs could improve the system performance. In [74] researchers have used linear correlation

between pairs of keys to address this issue. We could use this approach in our research as a starting point.

- We observed that the outlier removal process played an important role in the user's profile creation process. Therefore, it is important to determine a proper outlier removal technique, in order to improve the system performance [61].

- We have noticed that the user's mood and time of day influence the dynamics [38]. We have also seen that users behave differently for different applications *e.g.* the user's behaviour will change from playing games to typing documents. These could be useful to adjust the $T_{lockout}$ threshold to improve the system performance. As an example, when a user is plays games we can lower the $T_{lockout}$ threshold or we can adjust the parameters A, B, C and D of the Algorithm 3.3.

- We have compared our approach with state of the art periodic approaches. The periodic approach can be seen as a special case of a *Sliding Window* based approach [121], where the window size and the step size are same. It would be interesting to see the results for *Sliding Window* based approach when the window size and step size are different.

- In our research, we assumed that at any time a session could be hijacked. In a practical scenario, the session hijacking can happen when the system is unattended for a small but significant amount of time, *i.e.* when there is no activity for an amount of time that would allow the user to leave his work place and for an attacker to get unnoticed access to the system.. Therefore, it is safe to assume that if there is continuous system activity (*i.e.* without a significant amount of pause), that then the data is coming from the same user (*i.e.* either genuine user or imposter user). So, to make our system more robust we can change the algorithmic parameters and the lockout threshold according to the user's activity. As an example, we can increase $T_{lockout}$ threshold or we can adjust the parameters A, B, C, and D of the Algorithm 3.3 after a significant pause of the user's activity and relax them during a period of continuous activity.

- As mentioned in Section 13.3, our dataset has user's application usage information (*i.e.* SI), these could also be useful as a clustering technique to improve the system performance. A preliminary results related to application based clustering approach to improve the system accuracy for CI can be found in [109] when using KD.

- We did not perform any analysis on the computational cost or system overhead for our CA system. Because computations are performed on single actions will each calculation of the change of the trust take limited time. We have selected behavioural biometrics because of the lower computational complexity compared to for example face recognition. More elaborate analysis is required before producing a deployable system.

- We believe that the sample size for the template creation has a very high impact on behavioural biometric research. We did not perform any analysis regarding this and we take this as a future work.

- There is an analogy between actions on a PC and on a touch screen. The touch screen tapping can be compared with keystroke actions while the swipes can be compared with mouse movements. We have performed research on mouse actions and keystroke actions separately and combined. The research on mobile devices using only swipe actions can be compared to the research using only mouse actions. As can be seen from the thesis will the performance change when including both keystroke and mouse actions and we believe that a similar observation can be made when including tapping with swipe actions. Our proposed techniques are general enough that they can be applied to any continuous authentication system, irrespective of the biometric modality. However, this needs to be confirmed with more experiments and analysis, and can be taken as a future work.

- Stylometry features [23, 24] and language identification technique [120] could be applied to improve the system performance. In literature stylometry features have been computed over a block of data, therefore we have to apply this technique in our research in a different way. If a user did not lock out from the CA system after a significant amount of keystroke actions, we can compute stylometry features within these keystroke actions (see *Block wise Feature Extraction and Comparison* module in Figure 14.1). Based on the classification results of these stylometry features we can re-adjust the algorithmic parameters to make the CA system more robust. Similarly, language identification technique and state of the art statistical features can be applied to improve the system performance.

- The objective of CI is to use it as forensic evidence. Through our experiment we produce the proof of the CI concept and the possibility to explore this in future. In our research, we directly use classification scores to identify the potential adversary. Therefore, it has a limitation to produce this evidence in court. The *likelihood-ratio* computation (*i.e.* $\frac{P(SC|H_p)}{P(SC|H_d)}$, where SC is the score computed by the CIS, H_p is the prosecution hypothesis and H_d is the hypothesis of the defence) could be explored to convert the identification scores as a forensic evidence in CI [6, 125].

Part V

Appendix

Appendix A

BeLT - Behaviour Logging Tool

We present the design and implementation of a Windows operating system based logging tool, which can capture keystroke, mouse, software interaction and hardware usage simultaneously and continuously. Log data can be stored locally or transmitted in a secure manner to a server. Filter drivers are used to log with high precision. Privacy of the users and confidentiality of sensitive data have been taken into account throughout the development of the tool. Our behaviour logging software is mainly designed for behavioural biometrics research, but its scope could also be beneficial to proactive forensics and intrusion detection.

This chapter is based on the papers published in: [106] MONDAL, S., BOURS, P., JOHANSEN, L., STENVI, R., AND ØVERBØ, M. *Importance of a Versatile Logging Tool for Behavioural Biometrics and Continuous Authentication Research.* IGI Global, 2015, ch. Handbook of Research on Homeland Security Threats and Countermeasures.

A.1 Introduction

Facing an increasing number of computer users and cyber-crime enabled by weak authentication mechanisms, a CA system that can monitor a claimed user's identity throughout a session could be a strong addition. It is challenging enough to design a CA system, which is unobtrusive, user friendly (where the legitimate user is never or very infrequently locked out by the system) and at the same time secure enough to detect any illegitimate user as soon as possible. There are many possible ways to implement a CA system, but behavioural biometrics are promising enough to achieve cost effectiveness (because no special hardware is required) and unobtrusiveness [151]. To create such a system, it is necessary to analyse a large amount of information about multiple users regarding how they interact with their computers. This information includes keystrokes, mouse usages, software interaction and hardware events. It is also necessary to focus on the input of the user via mouse and keyboard simultaneously to defend against an attacker avoiding detection by restricting to one input device because the system only checks the other input device [1, 10, 63, 146]. Software Interaction and Hardware usage information could be used to improve the system performance.

Existing literature does not very well cover approaches that combine both keyboard and mouse logging with arbitrary application interaction. The granularity of measurements is sometimes too coarse for analysis with respect to behavioural biometrics. Surveying the literature, we found also that there is a lack of discussion about the capture software and the capture environment. Few of the articles provided information on the technology behind the capture software. Most of the datasets and tools used for capture are not publicly available. Hence, it is impossible to replicate their results and methodology. We address these issues and present a tool that combines different methods of interaction of a user with a computer, and we disclose implementation details so that the technologies used to capture keyboard, mouse, and application interaction can be employed in alternative implementations.

Most of the logging tools available at present can capture only mouse and keystroke information [50, 53, 77]. According to our knowledge only *AppMonitor* [5] can store software interactions, limited to two specific applications. Therefore, there is strong demand within the behavioural biometrics based CA research community to design a logging tool that can capture the relevant users behaviour information and share the captured data with the research community for analysis. Based on our survey of related work, our tool *BeLT* is the first tool which can capture extensive amounts of information, *i.e.* keyboard, mouse, arbitrary application interaction as well as certain hardware

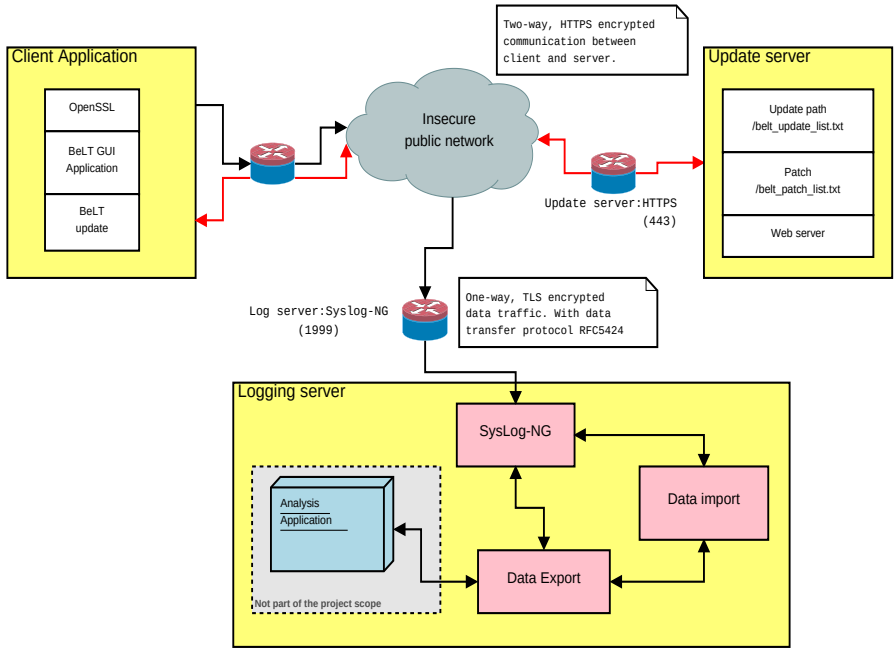


Figure A.1: Deployment diagram of BeLT system.

events, simultaneously and unobtrusively. Also, it gives users the choice to store the information either on the local computer or have it sent to a secure server.

A.2 Architecture of this tool

In this section we discuss the methodology and the architecture of BeLT. Figure A.1 shows the complete deployment architecture, comprising the BeLT client application, the logging server and the update server. Logging server and update server can be the same host, and for each server there can be many hosts running the BeLT client application.

A.2.1 BeLT client application

Figure A.2 shows a logical view of the client application. The yellow boxes indicate components and arrows indicate information flow. The *Data Capturing* component runs in a separate thread and sends the data to the *Data Processing* component. Then the Data Processing component sends the locally stored data to the *Transmission* component. The Transmission component is responsible for sending all the logs to a central server. If users want to peek into the captured data in real-time, the Data Processing component can send the processed data to the *Graphical User Interface* component. The *Update Service* updates the software if applicable.

A.2.1.1 Data capturing

This component captures all events related to keystroke, mouse, software interaction and hardware information. Key and mouse events are recorded using hooks (see Section A.3.1), software interaction is picked up using *User Interface Automation (UIA)* (see Section A.3.2). Hardware events include information about user I/O devices (keyboard type, language, screen resolution), CPU/RAM

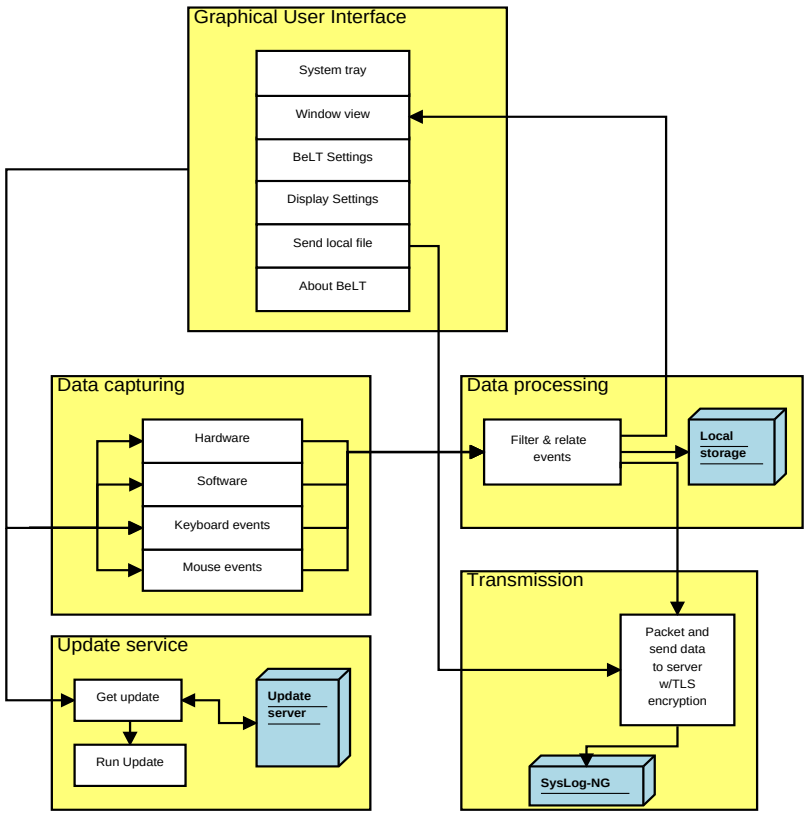


Figure A.2: Logical view of the BeLT client application.

consumption, and comprise events like change in screen resolution and insertion/removal of USB-connected devices. The data format for captured events is documented in Section A.4.

Hooking and UIA are techniques that are available in the Win32 API of the Microsoft Windows platform and do not exist as such in other operating systems like Linux, OS X, or Android. However, similar approaches could employ *events* of the */dev/input* device in Linux-based operating systems, and making use of the X11 protocol for window-based user interfaces. Modification of the operating system kernel would also be a feasible approach to integrate logging of user input, but is not as easily deployable as a solution that does not require kernel changes.

A.2.1.2 Data processing

All registered events are immediately sent to the data processing component. Its tasks are to relate events, filter out unnecessary data (*e.g.* duplicate events triggered by some applications), and to format the data according to the format of the *Syslog* protocol.

All mouse and keyboard events are seen as input from the user, and software events are seen as a consequence of that input. For later analysis, it can be useful to know which events are related to other events. Inside the data processing component we determine how events relate to each other. For a full discussion on how this is done, see Section A.4. When the data processing component has finished processing an event, the event is added to a list in memory. When that list reaches a configurable threshold (in our current implementation this is set to 500 events), the list is flushed to

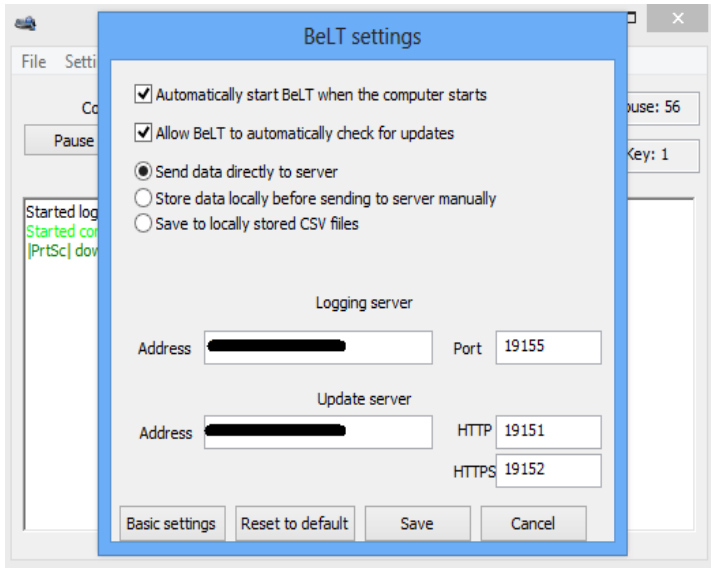


Figure A.3: GUI settings options.

the transmission component in a separate thread. Queuing events in memory before sending them over the network ensures that events are processed fast enough. If they were not processed fast enough, the hook used for processing might be silently removed by the operating system to improve user interface performance.

A.2.1.3 Transmission

This component has the responsibility to set up an encrypted session between the client and the server and to send all the data to the server. If the BeLT client application is unable to connect to the server, or unable to send an event, the event is kept in memory. When a configurable threshold is reached (in our current implementation this is set to 10,000 events), the list of events is flushed to local storage. The next time, the BeLT client application is launched, the locally stored data is attempted to be transmitted to the server.

A.2.1.4 Graphical user interface

The BeLT client application contains a simple *Graphical User Interface (GUI)*. It is mainly used for testing or by researchers collecting small amounts of data. In a large-scale deployment, the GUI is less relevant, because most users are not interested in controlling execution of BeLT.

Configurable options available to the user via the GUI are displayed in Figure A.3. Most relevant is the option to store data locally before manually sending it to the central logging server. That way, users can review data and remove sensitive parts to guard their privacy.

A.2.1.5 Update service

The update service ensures that the user always runs the most recent version of the BeLT client application. Maintenance updates and security fixes can hence be distributed easily to all users. The update service communicates with the *update server* that is further described in Section A.2.3. The service checks whether the update server provides a newer version than is currently installed, and then downloads the new version and installs it. A new version is downloaded over *https* as a MSP

or MSI file (Microsoft Patch/Microsoft Installer). After the file has been downloaded, an update process is launched, BeLT is terminated, the patch installed, and BeLT is started again.

A.2.2 Logging server

Our logging server is a Ubuntu server running MySQL as the DBMS and Syslog-NG to receive log data. Functionality of the logging server is described in Section A.3.3.

A.2.3 Update server

The update service of the BeLT client application needs to communicate with the *update server* that maintains information about which versions have been released and what the current version is. The update server maintains two separate lists: the first list contains the version number for the latest release, and the second list contains all patches. Access to these two lists is provided by the Apache 2 web server running on the update server. The update server can be the same machine as the logging server or can be a separate machine. All patches have to be applied incrementally by the update service, based on the current version of BeLT on the local machine and the target version supplied by the update server.

A.3 Underlying technologies

In this section we discuss some of the technologies that we apply in our work.

A.3.1 Hooks

Hooks are a mechanism in the Microsoft Windows API to monitor and intercept window messages between applications and the operating system[1]. It is possible for threads to call *SetWindowsHookEx()* to install a subroutine to be notified about window messages sent to applications and to modify the content of these messages. The mechanism does not require privileges to be used, but might be restricted for applications on a lower integrity level (which in practice is seldom used, the exception being *Microsoft Internet Explorer*).

BeLT uses *WH_KEYBOARD_LL* and *WH_MOUSE_LL* hooks to monitor messages on keyboard and mouse events respectively. Owing to the granularity of the timestamps for window messages, we are able to capture the user's behaviours with 16ms granularity. Further have we implemented Filter Drivers to reduce the timestamps granularity into 1ms, see Section A.3.6.

A.3.2 User Interface Automation

User Interface Automation (UIA) is a Microsoft Windows API to retrieve information about the current state of *User Interface (UI)* elements of running software, and about state changes of UI elements. This technology was introduced with Windows XP SP3 and has been available in all following versions of Windows[2]. The intention is to make it easier to develop assistive software for users with special needs. It is also a convenient way of observing user interaction with applications, using the same technology for non-assistive software.

Table A.1 shows the complete list of software events that we capture with BeLT, together with a description of when those events occur.

A.3.2.1 Constraints

All UIA calls need to be made on a separate non-UI thread to avoid negative impact on performance; using a UI thread to create and capture UIA events in the same thread might lead to the application to

[1] `http://msdn.microsoft.com/en-us/library/windows/desktop/ms632589.aspx`
[2] `http://msdn.microsoft.com/en-us/library/windows/desktop/ee684009`

Table A.1: List of UIA/MSAA events captured by BeLT.

Name	Description
Window Open (WO)	Whenever a program opens a new window, this event occurs.
Visual Change (VC)	This event occurs when the application window is resized or minimized or restored.
Menu Opened (MO)	When the user browses menu items, this event occurs.
Menu Mode Started (MMS)	This event occurs when the user clicks on a menu for the first time.
Focus Change (FC)	Whenever the focus shifts to another GUI element, this event occurs.
Element Invoked (EI)	This event occurs when the user presses any button.
Text Changed (TC)	This event occurs whenever the text changes in any user editable element.
Object Change State (OCS)	Whenever the element's state changes, this event occurs.

stop responding to the UI requests. We use caching of UIA events, to efficiently retrieve information on a large number of events, including *e.g.* process ID, or element name.

We can only capture application interaction from applications that support UIA, either explicitly or because the GUI framework used in these applications supports UIA by default. We have successfully verified that, among others, Microsoft Office, Microsoft Visual Studio, Matlab, web browsers, PDF readers, and Skype support UIA.

A.3.2.2 Privacy protection

We need to avoid logging any sensitive information by our software to maintain privacy of users and security of the users' data. In most cases, it is hard to see what sensitive information is, but passwords are one type of information we are able to detect and, hence, do not store. We detect whether an *Edit* control possesses the *ES_PASSWORD* style indicating this UI element contains a password or other sensitive information not to be meant to be shared with others and made BeLT not to record this information.

A.3.3 Remote logging

Our tool BeLT can store data on the local machine where it is captured, and can send it immediately or deferred to a server using an encrypted channel. We use the established Syslog transfer protocol, RFC 5424[3].

Figure A.4 shows the logical view of the BeLT server application. We can divide the server application into three main parts:

- *Syslog-NG* – We use the open source version of Syslog-NG as our server component that receives logs, using the Syslog protocol.
- *Data Import* – To obtain our specific data format, we import the XML-formatted data received via Syslog-NG into an indexed database so that a large amount of data from a large number of users can be handled efficiently.
- *Data Export* – For research and analysis, we can export the database content to formats amenable to processing with other tools, including CSV.

Transport Layer Security (TLS)[4], is a way to provide confidentiality and integrity of data transmitted over an insecure channel. The protocol can be divided into two layers: one is the handshake protocol and the other is the record protocol. The handshake protocol is based on three properties:

[3] http://tools.ietf.org/html/rfc5424
[4] http://tools.ietf.org/html/rfc4346

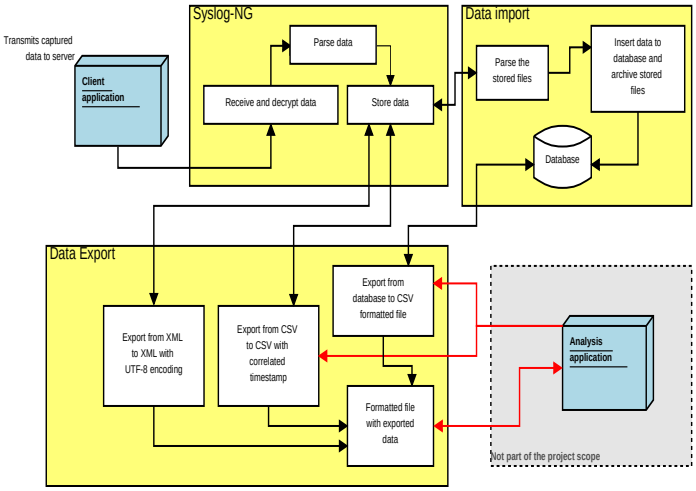


Figure A.4: Logical view of the BeLT server application.

1. Each peer can be authenticated.
2. Secure negotiation of a shared secret.
3. The negotiation cannot be altered without detection.

The record protocol serves as encapsulation of higher level protocols and has the following properties:

1. The connection is private, where the symmetric keys are generated uniquely each time.
2. The connection is reliable, the message must include an integrity check.

A.3.4 Pseudonymity of the user

When we capture behavioural data, we also capture information that can identify the person behind the behaviour. To anonymise the real identity of the person behind the data, BeLT assigns each user a unique identity (ID) to be used in the log. This ID is generated by concatenating serial numbers for the operating system, the motherboard, the primary disk drive, and the unique ID for the computer system as reported by WMI (if available, can be all zeros). The concatenated serial numbers are then hashed with the widely used and fast MD5 hash function to generate the 128 bit ID, encoded to *base64*. In case of a hash name collision, identical IDs would cause information from multiple users to be stored as belonging to a single user. Data could be appended to an already existing file or corrupt an existing file. This would render the research data useless for analysis. Since the number of hash function operations currently believed to be necessary to find collisions for MD5 is 2^{64}, we consider the choice of MD5 for generation of fixed length unique IDs to be appropriate [85]. After the ID is generated, it is stored in the user's profile to keep it persistent. The profile is stored in the user-dependent *AppData* folder of the local computer.

A.3.5 Mouse movement data compression algorithm

Storing every single mouse move event takes up space on the local machine and later on the server. Earlier work by [77] measured 22 KB/minute with "extensive mouse movements". This amounts to CA 10 MB per workday just in mouse movement logging.

Algorithm A.1: Algorithm for mouse data compression

Data: $A = (x_{i-1}, y_{i-1})$, 2^{nd} last recorded coordinate; $B = (x_i, y_i)$, last recorded coordinate; $C = (x_c, y_c)$, current mouse pointer coordinate; $\Delta d \leftarrow$ threshold for distance; $\Delta\theta \leftarrow$ threshold for angle

Result: $\rho = (x, y)$, mouse pointer coordinate that needs to be recorded

begin
- $\bar{u} = \overrightarrow{AB} = \langle x_u, y_u \rangle = \langle x_i - x_{i-1}, y_i - y_{i-1} \rangle$
- $\bar{v} = \overrightarrow{BC} = \langle x_v, y_v \rangle = \langle x_c - x_i, y_c - y_i \rangle$
- $ds = \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2}$
- $ang = \frac{180}{\pi} \cos^{-1} \frac{\bar{u} \cdot \bar{v}}{|u||v|} = \frac{180}{\pi} \cos^{-1} \frac{(x_u \times x_v) + (y_u \times y_v)}{\sqrt{(x_u)^2 + (y_u)^2} \times \sqrt{(x_v)^2 + (y_v)^2}}$
- **if** $ds \geq \Delta d$ **then**
  - $\rho = (x_c, y_c)$
- **else**
  - **if** $ang \geq \Delta\theta$ **then**
    - $\rho = (x_c, y_c)$
  - **else**
    - *continue without storing* C

We wanted to limit this number while still managing to recreate the original path with good accuracy. We developed Algorithm A.1 for mouse move data compression. The Algorithm A.1 is based on two parameters, the difference in distance between two points and the change in angle between two points. To calculate the distance we use Euclidean distance and calculate the angle with *arc tangent*.

After rigorous testing we found that having a distance change of 10 and a degree change of 5 provided decent results. The compression rate for these settings was 70% while the quality of the reduced dataset was still high. Hence, we use these as default parameters in the BeLT tool. Figure A.5 shows the mouse movement trajectory both for the original path (in black) and for the compressed data points (in red) generated using our algorithm with a distance change of 10 and a degree change of 5. Figure A.6 shows the same mouse movement trajectory compared against the compressed data points using a distance change of 20 and a degree change of 20. The compression rate in this case was almost 90%, but the quality of the reduced dataset was judged too low.

A.3.6 Filter Drivers

We implemented a keyboard filter driver and a mouse filter driver to capture keystroke and mouse event information in kernel mode before they are processed by the user mode APIs of the Windows operation system. The filter drivers use *KeQuerySystemTime()* (Windows Driver Kit 7.1.0 and Visual Studio 2012 on Windows 7) and *KeQuerySystemTimePrecise()* (Windows Driver Kit 8.1 and Visual Studio 2013 on Windows 8.1) to retrieve timestamp data.

To test granularity of measurements, we subjected the keyboard filter driver twice to 10,000 simulated keystrokes. The keystrokes were from the range 'a' to 'z' and sent without delay between two keystrokes. After 10,000 simulated keystrokes, a pause of 5 seconds was inserted. Simulation was implemented with a Teensy 3.0 board that was connected as a keyboard device to the USB port of the computer.

We observed that timestamps for recorded keystrokes had 1-2ms granularity on Windows 7 x64 and 15-16ms granularity on Windows 8.1 x64 using the same driver source code.

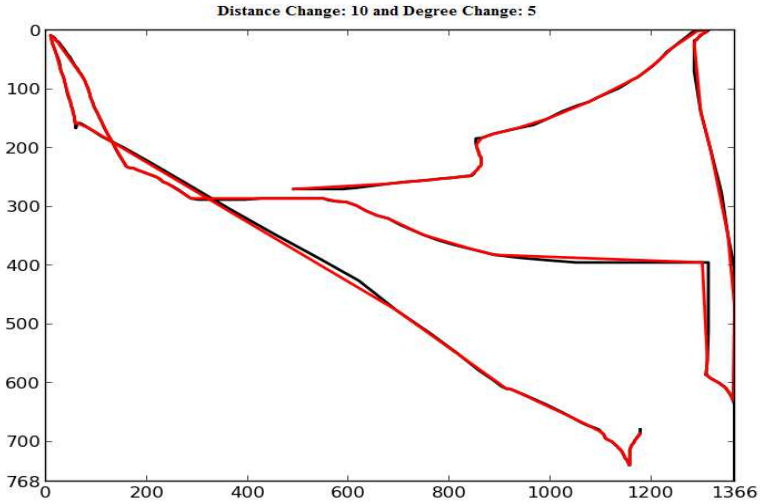


Figure A.5: Compression of 30% of the original data points.

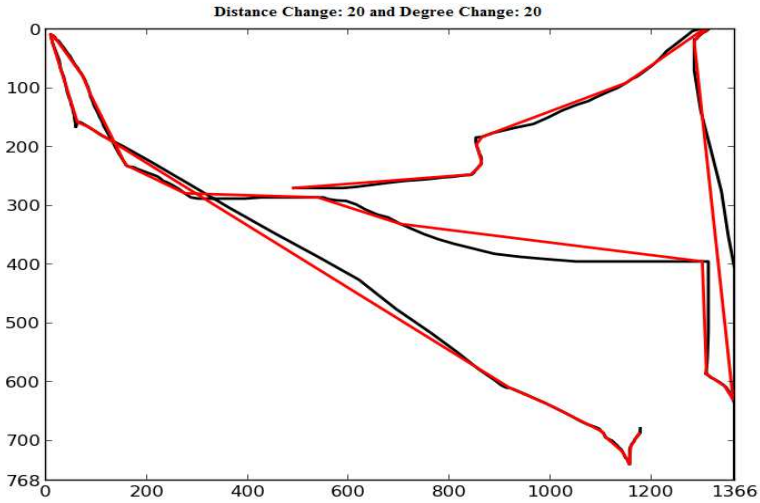


Figure A.6: Compression of 11% of the original data points.

A.4 Data Format

We store the data we capture in a unified format for all events, so that they can be easily transformed for export in formats used by post-processing and analysis software.

Table A.2: Data format for keystroke events.

Sequence	Event Type	Action	Value	Time	Relation	Flag	Additional fields
n	'K'	'D' 'U'	String	ms	Seq.	Int	n/a Count

Attributes shared by all events are:

- **Sequence** – a session-unique Integer value assigned by order of occurrence
- **Event type** – a *char* value indicating the type of event, *i.e.* 'K'/keyboard, 'M'/mouse, 'S'/software and 'H'/hardware;
- **Action** – a sequence of *char* values further detailing the event, depending on the *event type*;
- **Value** – a String value representing an instance of a detailed event type, depending on *event type* and *action*;
- **Time** – a timestamp when the event occurred, with millisecond precision;
- **Relation** – a *Integer* value representing a related *Sequence*.

In addition, some event types have special attributes:

- **Flag** – used for keystroke, mouse, and software events
- **Value-2** – used for hardware events
- **Count** – used for keystroke events
- **Rectangle** – used for mouse events
- **Element description**, **element ID**, **area**, **extra information**, **additional flag** – used for software events

A.4.1 Keystroke events

The data format for keystroke events is shown in Table A.2. The *event type* is always 'K'. Keystroke events have only two types of *actions*, key press ('D'/down) and key release ('U'/up). The UTF-8-encoded *Value* field states which key was pressed or released; it is either the typed character or a string with a descriptive key name, enclosed by "||". A ISO8601-compliant *timestamp* of when the event occurred is recorded in milliseconds [62]. The *relation* attribute contains a corresponding seq. number of a previous event and is further documented in Section A.4.5. *Flag* is an Integer indicating which alternate/system key was active. Bit 0 (LSB) is set when the *Alt* key was down, bit 1 is set for *Ctrl*, bit 2 for *Shift*, bit 3 for *Win*, bit 4 for *Caps Lock*, bit 5 for *Num Lock*, bit 6 for *Scroll Lock*. *e.g.* , if *Alt+Shift* was pressed, *Flag* would be set to 5. In case of a key up event, an additional field *Count* states how often a key was repeated. Permanent use of *Num Lock* may indicate that the user operates a numeric keypad.

A.4.2 Mouse events

The data format for mouse events is shown in Table A.3. The *event type* is always 'M'. Mouse events can have four types of *actions*, mouse move ('M'/move), mouse wheel use ('W'/wheel), mouse button press ('D'/down), and mouse button release ('U'/up). The *Value* field contains the x_y mouse pointer coordinates concatenated by an '_' underscore character. In case of mouse wheel use, *Value* is the corresponding delta value indicating how much the wheel was scrolled; positive values are upward scrolls, negative are downward scrolls. A ISO8601-compliant *timestamp* of when the event occurred is recorded in milliseconds [62]. The *relation* attribute contains a corresponding seq.

Table A.3: Data format for mouse events.

Sequence	Event Type	Action	Value	Time	Relation	Flag	Additional fields
n	'M'	'M'	x_y	ms	Seq.	n/a	n/a
		'U'	x_y			Int.	Rectangle
		'D'	x_y				Rectangle
		'W'	Delta			n/a	n/a

Table A.4: Data format for software events.

Sequence	Event Type	Action	Value	Time	Relation	Flag	Additional fields
n	'S'	"FC"	s/w name	ms	Seq.	El. type	Desc., ID, area
		"EI"					Desc., ID, area
		"MO"					Desc., ID
		"MMS"					
		"WO"					
		"TC"					Desc., ID, extra inf.
		"VC"					Desc., ID, add. flag
		"OCS"				State	Desc., ID, Area

number of a previous event and is further documented in Section A.4.5. *Flag* is an Integer indicating which mouse button was pressed/released: $1 = left$, $2 = middle$, $3 = right$. If several buttons are pressed, several events are generated. Mouse moves and and mouse wheel events do not use the *Flag* value. In case of a mouse button press/release, an additional field *Rectangle* stores the active area, *i.e.* the client area of the UI element, where the mouse click happened.

A.4.3 Software events

The data format for software events is shown in Table A.4. The *event type* is always 'S'. Software events can have eight types of *actions* as described in Table A.1. The *Value* field contains the executable file name of the process displaying the user interface. A ISO8601-compliant *timestamp* of when the event occurred is recorded in milliseconds [62]. The *relation* attribute contains a corresponding event ID of a previous event and is further documented in Section A.4.5. *Flag* is an Integer indicating the type of element according to the control type identifiers listed in[5] or the state of the UI element ($0 = unpressed$, $1 = pressed$) in case of the *OCS* action. Flag is stored as an offset to the *UIA_ButtonControlTypeId* base constant, *i.e.* the one with the lowest value. Additional fields comprise a UI *element description* and an *element ID*, *i.e.* the *UIA_NamePropertyId* (description) and the *UIA_AutomationIdPropertyId* (ID) as documented in[6]. An *additional flag* is recorded for Visual Change ("VC") events, with values covering $1 = Restored$, $2 = Maximized$, $3 = Minimized$. *Area* logs the rectangular area on the desktop that the element occupies, formatted as left, top, right, bottom. *Extra information* contains the resulting text of the UI element when a Text Change ("TC") event occurs.

A.4.4 Hardware events

The data format for hardware events is shown in Table A.5. The *event type* is always 'H'. Hardware events can have five types of *actions*: *KEY* (keyboard type and language), *SCR_Info* (info on screen resolution per physical screen whenever one is detected during the session; change of screen resolution also triggers a *SCR_Info* action), *SCR* (change of screen used for user input in case of focus change event when more than one screen is used), *RES* (CPU+RAM resource use recorded every 10 minutes), *DEV* (insertion / removal of USB-connected devices; HID-class devices could alter input behaviour). The *Value* field depends on the *Action*. For *KEY*, the value is the language and

[5] http://msdn.microsoft.com/en-us/library/windows/desktop/ee671198.aspx
[6] http://msdn.microsoft.com/en-us/library/windows/desktop/ee684017.aspx

Table A.5: Data format for hardware events.

Sequence	Event Type	Action	Value	Time	Relation	Flag	Additional fields
		”KEY”	Language				Keyboard type
		”SCR_Info”	Resolution				ID
n	’H’	”SCR”	ID	ms	n/a	n/a	n/a
		”RES”	CPU				RAM
		”DEV”	Insert/Remove				n/a

Table A.6: Events and their relationships

Event Type	Action	Relation to previous event
Keystroke	Key pressed	Software event
	Key released	Key pressed
Mouse	Mouse moved	0 (for beginning of sequence of mouse moves)
		Latest mouse button down event (if button still down, *e.g.* drag& drop)
		Previous mouse move sequence
	Mouse button pressed	Software event
	Mouse button released	Mouse button pressed
	Mouse wheel used	Software event
Software	all	Latest keystroke/mouse *released* event

sub-language. For *SCR_Info* and *SCR*, value contains the screen resolution as top, left, right, bottom. For *RES* events, value records the percentage of CPU use. For *DEV*, value is 1 for device insertion and 2 for device removal. A ISO8601-compliant *timestamp* of when the event occurred is recorded in milliseconds. The *relation* and *Flag* attributes are not used. Additional fields comprise *keyboard type* for *KEY* events, *screen ID* for *SCR_Info* events, and *RAM* for *RES* events, with the value stating the percentage of RAM used.

Hardware events are not a direct capture of user behaviour; they give a more complete picture of the user environment for later analysis.

A.4.5 Relation between events

Many events are related to other events. This concerns, *e.g.* pairs of key presses/releases and mouse button presses / releases as well as software events triggered by keyboard / mouse input and sequences of mouse movements and software events. A full overview of events and their relationships is given in Table A.6.

Keystroke/Mouse events that are related to previous software events, *i.e.* key presses, mouse button presses and mouse wheel use, store the seq. number of the latest event pertaining to the active window. Key releases and mouse button releases store the seq. number of the preceding corresponding key presses and mouse button presses for the same key/button. Mouse move events are not related to previous events, *i.e.* store an seq. number of 0, when they are the beginning of a sequence of mouse moves. They store the seq. number of the latest mouse button down event if the button is still down at the time of the mouse move. If several buttons are pressed, relation is stored to the latest button down event. Relation to a button down event is stored to indicate that the user is likely to drag a logical object with the mouse. Mouse move events store the seq. number of the previous mouse move event in all other cases. Software events always store the seq. number of the latest keystroke/mouse event as its likely cause.

A.5 Summary

Our designed logging tool *BeLT* has the following functional and non-functional properties, which gives the advantages over existing logging tools [5, 50, 53, 77]:

1. Continuous collection of keystroke, mouse, software interaction and hardware events;
2. Security of the users' sensitive behaviour by exclusion of typed passwords;
3. Unobtrusive and stable data capture;
4. Efficient compression of mouse movement data;
5. Pseudonymity of the users to maintain their privacy;
6. Transmission of recorded behaviour to a server through a secure channel or storing information on the local system depending on the user's choice;
7. Implemented filter drivers to log the keystroke and mouse data with high precision;
8. Export options including raw data, CSV, SQL query/dump, XML;
9. Industry-grade application quality as required by Microsoft[7];

Our BeLT logging tool can be used in various types of experiments. In this research, we have performed an experiment on continuous authentication to utilize the highest capabilities of this tool.

[7] http://msdn.microsoft.com/en-us/library/windows/desktop/hh749939.aspx

Appendix B

Complexity Measurement of a Password for Keystroke Dynamics: Preliminary Study[1]

Abstract

This paper discusses the complexity measurement of a password in relation to the performance of a keystroke dynamics system. The performance of any biometric system depends on the stability of the biometric data provided by the user. We first present a new way to calculate the complexity related to the typing of a password. This complexity metric is then validated with the keystroke dynamics data collected in an experiment, as well as the user's experience during the experiment. Next, we show that the performance of the keystroke dynamics biometric system will depend on the complexity of the password and in particular that the performance of the system decreases with an increasing complexity. This leads then to the conclusion that random passwords might, although harder to guess by an attacker, might not be the most suitable choice in case of keystroke dynamics.

B.1 Introduction

The most used method for authenticating a person is still using passwords. This method is easy to implement and users are very much used to it. Passwords systems do have well known weaknesses because people choose simple passwords that are easy to guess. Many breaches of password databases have shown this.

It is known that the quality of chosen passwords is insufficient when left to the user. It is clear that users have a tendency to select passwords that are too simple and easy to guess. Most likely this will never improve, but we can use biometric keystroke dynamics to improve the quality of the authentication mechanism, even for simple passwords.

Websites that require passwords will now more and more indicate the strength of the password chosen by the user so that at the very least the user is aware of the quality of his password. We all know the strength of a password depends on the length, the unpredictability, and the complexity of a password. In this case, complexity refers to using a combination of letters, numbers, capitals, and special characters. The complexity of a password will increase when using a combination of these instead of restricting to just a single type of characters. Sometimes complexity of a password is defined as the entropy of the password.

In this paper, we look slightly different at the complexity of passwords in relation to the performance of Keystroke Dynamics (KD). We will in fact only restrict to using passwords that only contain letters when defining our complexity. We will look into combining KD [11, 70, 108] with password systems, *i.e.* not only should the user type the correct password, he or she should also type it in the correct manner. In this case, complicated words containing capitals and special characters might hinder the performance of the system. A password like 'px7(W1x,L*' will be difficult to type, even for the genuine user. In [73], an experiment was conducted with a random password and the best performance reported in the paper is 9.6% Equal Error Rate (EER). With a simple passwords like 'password' will the user be much more consistent in his or her way of typing this password. In

[1]This chapter is based on the paper published in: [105] MONDAL, S., BOURS, P., AND IDRUS, S. Z. S. Complexity measurement of a password for keystroke dynamics: Preliminary study. In *6th Int. Conf. on Security of Information and Networks (SIN'13)* (2013), ACM, pp. 301–305.

[66], the authors considered identification of users based on free typed text. They concluded that short English words (for example 'the', 'and', 'a' or 'in') are not very well suited for identification of people. This might result from the fact that these words are very short and that most people are proficient in typing those words.

Our contribution made in this paper as follows,

- New way to measure the complexity of a password for KD.
- Validate the complexity metric with data of 110 users.
- Performance evaluation with relation to complexity of the password for authentication.

In the remainder of this paper we will present a new complexity metric in Section B.2. We will validate this metric in Section B.3 and show that the performance of a biometric KD systems depends on the complexity of a password in Section B.4. Finally we will draw conclusions and present future work in Section B.5.

B.2 Password complexity

In the past, the complexity of a word has been defined based purely on the layout of the keyboard. In particular the physical distance between the keys related to two consecutive characters in a password was used [55]. This complexity metric seems to be related to the time it takes to travel from one key on the keyboard to another and the further two keys are apart, the more complex the key combination. For example, the distance between 'A' and 'P' is large and the distance between 'P' and 'O' is small, hence 'AP' is more complex than 'PO'. The complexity of a full word is the sum of the complexities of the bi-grams in a word.

The above complexity metric seems to assume that the user is using a single finger to type the password and that the uncertainty (and hence complexity) of moving to the next letter increases with the distance. Generally, people will use 2 hands to type on a keyboard, and the number of fingers used is more often near 10 then near 2.

Our new complexity metric depends on the following:

1. The layout of the keyboard;
2. The frequency of bi-grams occurring in English;
3. The number of consecutive letters to be typed with each hand;
4. The length of the word.

We will elaborate on each of these in the following subsections. Please note that we will restrict here to the typing of words that only consist of lower case letters, so we will not include capitals, numbers or special characters.

B.2.1 Keyboard Layout

In Figure B.1, an ordinary QWERTY keyboard is displayed. In our complexity measure we divided the keyboard into 7 areas, as shown with red lines in Figure B.1.

$$C_1 = \sum_{i=1..n-1} kb(k_i, k_{i+1}), \tag{B.1}$$

The complexity based on the layout of the keyboard (C_1 as in Equation B.1) represents the complexity of using the fingers to type two consecutive keys. If both keys are typed with different hands, then the complexity is low. The complexity increases when using the same hand, and even more if the same region of the keyboard is used.

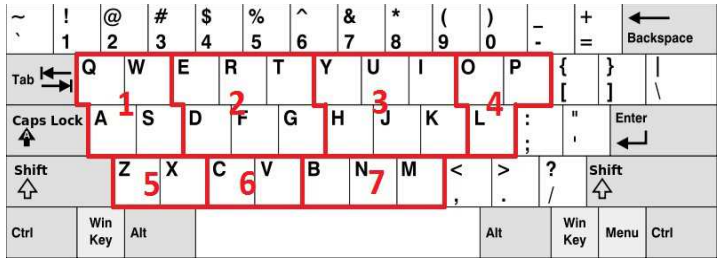


Figure B.1: QWERTY Keyboard Layout (Adapted from [150]).

The value of $kb(k_1, k_2)$ defines the complexity of typing the bi-gram k_1k_2, based on the layout of the keyboard. The total amount of complexity based on the layout of the keyboard is defined in Equation B.1.

The complexity of a particular bi-gram k_1k_2 is defined as a function of the areas on the keyboard, as marked in Figure B.1. The complexity is based on moving the fingers within each of the areas. For k_1 and k_2 in different areas the complexity $kb(k_i, k_{i+1}) = 0$. Also for typing the same key twice, *i.e.* actually not moving the finger, the value of $kb(k_1, k_1) = 0$. For the complexity for moving a finger inside one of the areas, the following rules apply:

1. If the keys are on the same row, then the complexity $kb(k_i, k_{i+1})$ equals either,
   a) 0 if the movement is away from the middle of the hand, or
   b) 0.2 if the movement is towards the middle of the hand.
2. If the keys are on different rows, then the complexity $kb(k_i, k_{i+1})$ equals either,
   a) 0 if the movement is straight up or down, or
   b) 0.5 if the movement is sideways and away from the middle of the hand, or
   c) 0.8 if the movement is sideways and towards the middle of the hand.

B.2.2 Bigram Frequency

People will get more fluent in typing particular key combinations when they have more practice. In this paper we assume that people use English on a daily basis while using the keyboard. Due to more frequent use of certain key combinations, the user will get more fluent in typing them, hence these key combinations will appear to be less complex to the user. Combinations like 'th' or 'in' occur more frequent in the English language than combinations like 'qi' (as in 'qiviut'). From this we derive that the complexity increases if the frequency decreases. Various frequency tables of bi-grams in the English language exist, all with minor differences to each other. We have decided to use the tables from [68]. We normalized the frequencies to range between 0 and 1. We then used the following formula to calculate the influence of the bi-gram frequency on the complexity of a word:

$$C_2 = \sum_{i=1..n-1} 1 - freq_{norm}(k_i, k_{i+1}), \tag{B.2}$$

where n is the length of the word, k_i is the i^{th} letter and $freq_{norm}(k_i, k_{i+1})$ represents the normalized frequency of the bi-gram k_ik_{i+1}.

B.2.3 Consecutive Letters with Each Hand

Typing will become easier if we can switch between hands often. When typing for example 'an', then when the left hand types 'a', the right hand can already "prepare" to next type the 'n'. On the other hand, when typing 'ta', then the left hand must perform both actions. In our metric we do assume that typing a bi-gram with one hand might not really pose a problem, but if more than 2 letters need to be typed by the same hand that this will increase the complexity of typing. For example, the word "state" needs to be fully typed with the left hand and might be considered more complex than for example the word "paper". If three or more consecutive letters have to be typed by the same hand, then the complexity increases. In fact, for each consecutive $r > 2$ letters with the same hand the additional complexity will be $(r - 2)$.

In general if there are l runs of at least 3 consecutive letters with either left or right hand and the lengths of these runs are $r_1, r_2, \ldots, r_l$, then the following formula represents the additional complexity due to these consecutive letters:

$$C_3 = \sum_{i=1..l} (r_i - 2), \tag{B.3}$$

B.2.4 Length of the Word

It is clear that the length of the word influences the complexity. It is also clear that a short word like "pet" is less complex than a long word like "interacting". At this point we do need to stress that we are considering pass-phrases, where various words in the pass-phrase are separated by spaces. If a pass-phrase consists of k words of lengths n_i for $i = 1..k$, then the part of the complexity related to the length of the word is simply defined as the average length of the words:

$$C_4 = \frac{1}{k} \cdot \sum_{i=1,\ldots k} n_i \tag{B.4}$$

B.2.5 Total complexity

The total complexity of a password/pass-phrase is not defined as the sum of the average word length (C_4) and the sum of the complexities C_1, C_2, and C_3 per bi-gram, or:

$$C = \frac{C_1 + C_2 + C_3}{\#bigrams} + C_4 \tag{B.5}$$

where the number of bi-grams can be calculated as $\#bigrams = \sum_{i=1,\ldots k} (n_i - 1)$, where the n_i are as defined in Section B.2.4.

B.3 Complexity metric validation

In this section we are going to discuss our data collection process and the validation methods for the complexity metric measure by the equation B.5.

B.3.1 Data Collection

We have checked the validity of our complexity metric based on an experiment performed at research institutes in France and Norway. In both institutes, participants were asked to type 5 different passwords, each 10 times. The number of participants was 70 in France and 40 in Norway. Notably, the participants in France used an AZERTY keyboard, while the participants in Norway used the QWERTY keyboard.

Table B.1: List of chosen passwords with entropy and incorrect typing per character.

nr	password	Entropy	incorrect/character
1	leonardo dicaprio	80.8	23.0
2	the rolling stones	85.6	14.8
3	michael schumacher	85.6	21.1
4	red hot chilli peppers	104.6	18.5
5	united states of america	114.1	17.5

Table B.2: Complexity of passwords.

pw	QWERTY		AZERTY	
nr	Eq. B.5	[55]	Eq. B.5	[55]
1	8.9	62.9	8.9	64.4
2	6.1	32.1	6.1	32.1
3	9.4	47.2	9.4	53.0
4	5.5	41.1	5.9	41.1
5	6.0	53.0	6.3	59.4

The number of incorrect typing was also recorded during the experiment. The passwords are given in Table B.1 and these passwords have been chosen because of the fact that all people know these names, so remembering them is relatively easy.

In Table B.1 also the entropy of the passwords is given. The entropy is directly proportional to the length of the password, which might make it less suitable to measure the complexity of a password. If L denotes the length of the password and N denotes the number of symbols that can be used in a password, then the entropy is equal to $L \cdot \log_2(N)$. In our case we used $N = 27$ because we used the 26 lower case letters and the space.

The number of incorrect typing per character of each of the passwords is given in column 4 of Table B.1. As longer passwords have more places where a user can make a mistake, we have divided the number of incorrect typing by the number of characters in the password to get the third column. It shows that passwords 1 and 3 again do have a higher error rate than the other three passwords while typing.

After the data collection the participants were also asked which password they felt was the most difficult to type. Approximately 95% of the participants answered that password 3 was the most difficult, while about 5% felt that password 1 was the most difficult. Hardly any participant felt that either of the other passwords was most hard to type.

B.3.2 Discussion

In Table B.2 the complexity of the passwords are presented. In this table the second and fourth column represents the calculated complexity according to new complexity metric from equation B.5. The complexity for the QWERTY keyboard in column 2 and for the AZERTY keyboard in column 4 are actually almost the same. Note that the absolute values in this table are not as relevant as the ranking of the passwords according to the complexity metric. We can see that passwords 1 and 3 are more complex than the other 3 according to our new complexity metric. After taking the experiment, the users indicated indeed primarily that passwords 1 and 3 were the most complex, which then complies with the data in the table. Columns 3 and 5 represent the complexity according to the measurement in [55] for both the QWERTY and the AZERTY keyboard. We can see here that this complexity measure has a different ranking. In both cases, password 1 and 5 are considered the most complicated due to the highest values.

We did consider that people might be more fluent when typing with their dominant hand, *i.e.* find that typing with their non-dominant hand is more slightly difficult. For this reason we adjusted

the $kb(.,.)$ values in Equation B.1, so that some complexity was added when going from one area to another. Also in that case, typing multiple letters consecutively with the dominant hand (as part of Equation B.3) did not add to the complexity anymore.

B.4 Performance assessment

In this section we will have a closer look at the data collected in the experiment and see if the complexity of the 5 passwords from Table B.1 influences the performance of the KD system. As mentioned earlier, each participant typed each of the passwords 10 times. We create a template for each user by calculating the mean and standard deviations for the latencies related to a password. A password of length n will give $n-1$ latencies and each user has typed the password 10 times from which the $n-1$ pairs (μ_i, σ_i) are calculated.

As only 10 instances is not really sufficient to split the data to create a high quality template and have sufficient data left for testing, we did adjust our analysis slightly. In Section B.4.1, we calculated the performance of the system by comparing templates to templates, where each template is based on all 10 samples of a user. For completeness sake we did also analyze the data by using 5 of the 10 instances to create a template and the remaining for testing, but given these numbers, the conclusions on the performance of the system for the various passwords may not be extended to a system where sufficient sample is available. The performance analysis with state of the art technique can be found in Section B.4.2.

B.4.1 Comparison based on templates

The way we did the performance analysis in this research is by comparing the templates of two different users. We do assume that the latencies approximately have a normal Gaussian distribution with the mean value μ and standard deviation σ as in the template. Given the known 68-95-99.7 rule, we know that $68\%/95\%/99.7\%$ of the measurements of the genuine user are within $1/2/3$ standard deviation σ from the mean μ. Furthermore we use the following simple distance metric between a template $T = ((\mu_1, \sigma_1), \ldots, (\mu_{n-1}, \sigma_{n-1}))$ and a test input $t = (t_1, \ldots t_{n-1})$. We calculate $D = dist(T, t) = \sum_{i=1,\ldots n-1} \Delta_i$, where

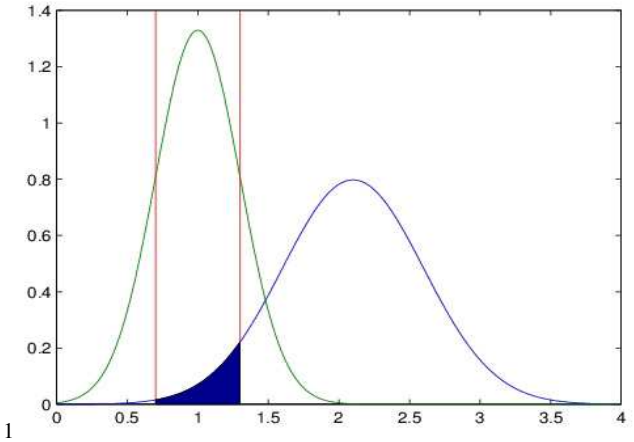
$$\Delta_i = \begin{cases} 0 & \text{if} \quad |t_i - \mu_i| \leq k \cdot \sigma_i \\ 1 & \text{if} \quad |t_i - \mu_i| > k \cdot \sigma_i \end{cases} \tag{B.6}$$

where $k = 1, 2, 3$. In the description below we will assume that $k = 1$, but the results will be summarized for all 3 values of k. In Figure B.2, we see the distribution a single latency of the genuine user in green and the distribution of that same latency of the imposter user in blue. The red lines indicate the ranges of values that will result in $\Delta_i = 0$. The size of blue area in the figure does now represent the probability that for that given latency the imposter value is accepted (*i.e.* $\Delta_i = 0$). Obviously this probability needs to be calculated for all of the $n-1$ latencies in the password.

In our analysis, we will compare the templates with each other instead of actually comparing a template to a test input. Given 2 normal distributions of the i^{th} latency, one with mean $\mu_i^{(1)}$ and standard deviation $\sigma_i^{(1)}$ and one with mean $\mu_i^{(2)}$ and standard deviation $\sigma_i^{(2)}$, it is easy to calculate the probability that a measurement of the second normal distribution falls within the range of $(\mu_i^{(1)} - \sigma_i^{(1)}, \ldots \mu_i^{(1)} + \sigma_i^{(1)})$. If this probability is denoted by p_i, then the expected contribution to the distance metric in Equation B.6 is actually $1 - p_i$. From this, it follows that the expected distance between the two templates equals

$$D = \sum_{i=1,\ldots n-1} 1 - p_i = (n-1) - \sum_{i=1,\ldots n-1} p_i, \tag{B.7}$$

where each of the p_i values depends on the latency distribution in the genuine and imposter template. From the above formula, we can derive that lower probability values lead to a higher distance value.



1

Figure B.2: Overlay of 2 normal distributions.

Table B.3: False Match Rate (FMR) for different *k* values (in %).

nr	*k*=1	*k*=2	*k*=3
1	2.03	0.26	0.11
2	1.14	0.03	0.02
3	3.30	0.71	0.34
4	0.63	0.06	0.02
5	0.36	0	0

If we compare the template of a genuine person to his own template, then we know that the probability approximately equals 68%, hence the expected distance will be equal to $(n-1)*(1-0.68)$.

In our analysis, we now compared each of the 110 genuine templates to all the 109 imposter templates in the above described manner. We counted how often the expected distance was below 5.12 for password 1. The results for all passwords and for $k = 1, 2, 3$ is given in Table B.3. From these results, we can clearly see that the highest percentages correspond to passwords that are indicated as being the most complicated in the second column of Table B.2.

It would be interesting to see if there were "cultural" differences, meaning that if we analyzed only the data from the French or only from the Norwegian participants, if the results would be different. We repeated the above analyses for $k = 1$ only and split the data into two parts, according to the location of the participants. The results are given in Table B.4. Generally the numbers are similar, but there are two major conclusions that we can draw from the numbers in Table B.4. The first is that for the French participants, password 3 has indeed the worst performance, but the expected low performance of password 1, based on the complexity of the word on the AZERTY keyboard is not visible. The second thing that sticks out is the major difference in performance between the French and the Norwegian participants of the second password: "the rolling stones". This is especially amazing as the characters in this password are on the exact same location on the QWERTY and the AZERTY keyboard. No reasonable explanation has been found yet for this difference.

Table B.4: FMR based on different *cultural* background (in %).

nr	all	French	Norwegian
1	2.03	1.59	2.82
2	1.14	1.80	0.19
3	3.30	3.13	3.85
4	0.63	0.70	0.96
5	0.36	0.39	0.77

Table B.5: FNMR in % for FMR=20%.

nr	all	French	Norwegian
1	31	33	30
2	27	29	25
3	36	35	38
4	25	30	20
5	25	29	18

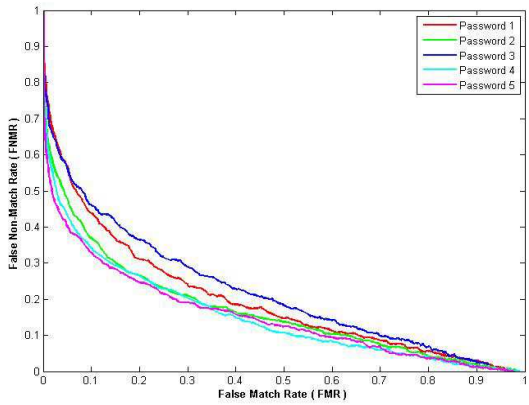


Figure B.3: DET curve for five passwords.

B.4.2 Performance measure by state of the art technique

In this section we are going to show the authentication performance using common biometric performance analysis methods. In our dataset we have 10 data sample per user per password. We randomly choose 5 samples of the genuine person to create the template. Testing is done with the remaining 5 samples for the genuine user. For testing with imposter data 100 random data samples are selected from all of the available data samples of the imposter users. In our analysis we used the Euclidean distance as our distance metric. We have separately calculated the system performance for data collected in France and Norway (see Table B.5).

In the analysis first we were interested to see the effect of the complexity on the False Non-Match Rate (FNMR) for a fixed False Match Rate (FMR). We adjusted the threshold such that the FMR was fixed at 20% and recorded the corresponding FNMR. Based on our observation we can again clearly see that the FNMR increases if the complexity of the password increases (see Table B.5). In Figure B.3 we show the DET curves of the performance for the various passwords. In the construction

of the DET curves the data of all 110 participants is taken into account. We can clearly see from these DET curves that the performance of the system highly depends on the particular choice of the password.

B.4.3 Discussion

In this section we will discuss the results that we found by our analysis methods. In Table B.1 the entropy of our passwords is given. The entropy is normally used as a measure for the strength of a password, but if KD is included as an extra security measure, then this measurement of strength is no longer appropriate.

In the analysis of Section B.4 we have shown that there is a relationship between the complexity of a password as given in Section B.2 and the FMR/FNMR found in the analysis of the collected data. This leads to the idea that for password systems that use KD as an extra security measure, it could be wise to use "simple" passwords (for example dictionary passwords) but still with a reasonable length. The length will ensure a reasonable entropy, while the complexity will still be relatively low, hence the performance of the system will be best. More research is needed to verify the correctness or in-correctness of this idea. More research needs to be done to improve the total performance of the system, but that is beyond the scope of this article.

Note that the large performance difference between French and Norwegian participants for password 4 and 5 (see Table B.5) coincides with the complexity difference between these two passwords (see Table B.2).

B.5 Conclusions and Future Work

In this paper, we came up with a new complexity metric for passwords that can be used to make predictions about the performance of a biometric KD system. We have shown that easier (*i.e.* less complex) passwords give a better performance than more complex passwords. This then implies that, when KD is used, users can relax slightly in the actual password that they choose.

There are a number of improvements that can be made on the metric as it is now, for example including frequency tables that are not just based on the English language and extending the metric to include numbers and special characters. In this preliminary study we have mainly concentrated on the number of users to validate our metric. A natural next step will then be to include more passwords with the full range of characters on a keyboard, including capitals, special characters and numbers.

Appendix C

Person Identification by Keystroke Dynamics using Pairwise User Coupling

In this chapter, we describe a user identification system when typing on a keyboard. We use various Machine Learning techniques as classifiers in combination with a three different *Pairwise User Coupling (PUC)* technique and show the performance of each separate technique as well as the performance when combining two or more together. In particular, we show that PUC in a bottom-up tree structure scheme gives the best performance, both in terms of accuracy and time complexity. However, these schemes could equally well be applied to other pattern identification problems. We have also investigated the optimized feature set for person identification by using Keystroke Dynamics. Finally, we also investigated the performance of the identification system when a user, unlike his normal behaviour, types with only one hand, and we show that performance then is not optimal, as was to be expected.

This chapter is based on the paper published in: [103] MONDAL, S., AND BOURS, P. Person identification by keystroke dynamics using pairwise user coupling. Under Review in IEEE Transactions on Dependable and Secure Computing, 2016.

C.1 Introduction

Keystroke Dynamics (KD) is a well established behavioural biometric modality due to the unobtrusive nature of biometric data collection, low computational complexity and no special hardware required for data collection [11, 151]. KD is a well explored research domain in authentication, where the research problem is a two class problem *i.e. legitimate* or *imposter* user [14, 19, 70], but there is little research on the potential of KD for user identification *i.e.* on the N class problem [116, 144].

Due to the increasing vulnerabilities in cyberspace, security only is not enough to prevent a breach, but cyber forensics or cyber intelligence is also required to prevent future attacks or to identify the potential attacker. The unobtrusive and covert nature of biometric data collection of KD has a high potential for use in cyber forensics or cyber intelligence.

In our research, we will explore the potential of KD for person identification. We will focus on classification techniques with different KD features for identification. We will also explore the effect of one-handed typing on the accuracy of identification.

The main contributions of this chapter are as follows:

- We propose three different identification schemes for pairwise user coupling. These schemes could be useful for person identification when the biometric features are weak;
- Extensive analysis was done with a on-line exam based keystroke datasets;
- We performed analysis for both open-set and close-set settings;
- We compared the performance accuracy for one and two handed typing behaviour.

C.2 Related Research

In KD, users are identified or authenticated based on the way they type on a keyboard. When a password is typed, then not only the correctness of the password itself is checked, but also if the typing rhythm when entering the password is correct. This process is sometimes called password hardening. A KD based authentication or identification system is low cost and easy to implement, because most of systems are software based. In such a system, the keystroke timing information has to be captured and features for authentication or identification are extracted [11, 8].

KD is a well establish biometrics for static authentication or continuous authentication (*i.e.* continuously monitor the legitimacy of the user [19, 136]). The first article, as far as we know, referring to KD is by Umphress *et al*. [147] from 1985, but the majority of research in this area is from 2000 or later. Due to limited information, sparse data, high intra-class variation and low inter-class variation is person identification using KD a difficult task.

In [116] has an artificial neural network technique been used for real time user identification using KD. This approach was validated experimentally based on data of 6 users, each typing a 15 character phrase 20 times. The achieved identification accuracy was 97.8%. In [144] a Euclidean distance based nearest neighbour classifier has been used for personal identification and an accuracy of 99.3% for 36 users was achieved. The same technique was applied on the dataset that we have used our research to create a baseline, but the technique was not optimized for this dataset (see Table C.1).

C.3 Data Description

A unique keystroke dataset was used in our experiment which was collected from three online exams with five essay questions for each exam. The participants are undergraduate students from PACE University. To add more complexity to the dataset were the students instructed to type normally with both hands for the first exam, and for the second and third exam, they were instructed to use only the left hand and right hand for typing respectively. This dataset was also used in the "One-handed Keystroke Biometric Identification Competition" which was a part of the 8^{th} IAPR International Conference on Biometrics (ICB'15) [91].

The dataset consists of typing data of 64 students who provided at least 500 keystrokes on each exam. A subset of data from both-hands typing (*i.e.* data from first exam) was used as a training set and the rest of the data was used for testing. The training dataset only consisted of 500 normally typed keystrokes samples from each of the 64 subjects. All testing is done on blocks of up to 500 keystrokes. Due to the low number of keystrokes (*i.e.* < 1000) of some subjects does the test dataset included only data of 61 subjects. The number of blocks of 500 keystrokes per subject ranged from 1 to 38, and in total we have 471 such blocks. Split over the 3 exams we find:

- 203 blocks from the first exam (*i.e.* both-hands typing);
- 131 blocks from the second exam (*i.e.* left hand typing);
- 137 blocks from the third exam (*i.e.* right hand typing).

We would like to mention that this dataset was collected in an uncontrolled environment. From various previous studies we learned that collecting experimental data under controlled settings, with a specific task on a specific computer, has major disadvantages. In such a case will the user be focused more on completing the task, and their keystroke dynamics will not represent their normal typing behaviour [37, 3].

To the best our knowledge, is this the first keystroke dataset where one-handed typing of users has been considered for free-text analysis. This is a realistic scenario where the reference template was built by the keystroke data typed by both hand and the test data is partially coming from the single hand typing. In an every day situation it might happen that the user uses one hand for another task, like making a phone call or writing down notes. In this scenario, the identification performance

of keystroke dynamics base on one-handed typing needs to be studied. The complete description of the dataset can be found in [91].

Every keystroke k is encoded as $k = (A, T^p, T^r)$, where T^p, T^r are the timestamps in millisecond for key press and key release and A is the value of the pressed key. From the raw data are the feature vectors encoded as $FV_i = (A_i, A_{i+1}, d_i, l_i^{rp}, l_i^{rr}, l_i^{pp})$, where A_i and A_{i+1} are the i^{th} and $(i+1)^{th}$ keys, and d_i is the duration of the i^{th} pressed key (*i.e.* $d_i = T_i^r - T_i^p$). Furthermore do l_i^{rp}, l_i^{rr}, and l_i^{pp} represent latencies between the i^{th} and $(i+1)^{th}$ keys, in particular $l_i^{rp} = T_{i+1}^p - T_i^r$, $l_i^{rr} = T_{i+1}^r - T_i^r$, and $l_i^{pp} = T_{i+1}^p - T_i^p$.

C.4 Result Analysis

A conventional multi-class classification technique with any given classifiers (*i.e.* SVM, ANN, CPANN or DT) failed to achieve a good recognition rate. To overcome this problem did we develop the *Pairwise User Coupling (PUC)* technique, using the above mentioned classifiers. Three different identification schemes (*i.e.* S1, S2, and S3) were used for the comparison and decision module (see Section 10.2 for the description of these schemes).

In this section will we discuss the identification results obtained from the different analyses. The analyses focus on the proposed algorithms, keystroke features for identification, multi-classifier fusion and analysis on user's typing hand.

C.4.1 Analysis of the Proposed Schemes

We will discuss in this section the performance of our proposed PUC schemes under various circumstances.

C.4.1.1 Analysis of Scheme 1 (S1)

Figure C.1 shows the system identification accuracies obtained from S1 (see Section 10.2.2) for different block sizes m ($m = 50, 100, 150, 200, 500$) and different ranks r ($r = 1, 2, 4, 8$).

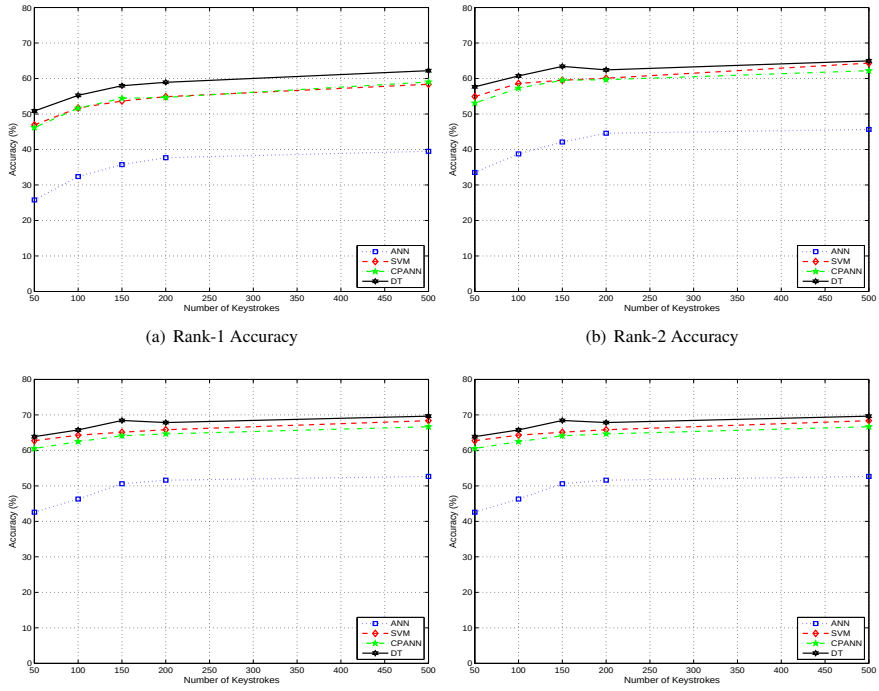
C.4.1.2 Analysis of Scheme 2 (S2)

Figure C.2 shows the system identification accuracies obtained from S2 (see Section 10.2.3) where $k = 25$ and for different block sizes m, where $m = 50, 100, 150, 200, 500$. We found that there is a large difference in accuracy between *Rank-1* and *Rank-8* for given k and m values, which motivated us to use the S3 scheme.

C.4.1.3 Analysis of Scheme 3 (S3)

Figure C.3 shows the system identification accuracies obtained from S3 (see Section 10.2.4), where $k = 25$ and $c = 8$. We can see the improvement on the results for *Rank-1*, *Rank-2*, and *Rank-4* when compared to S2. The *Rank-8* classification accuracy will be same for both S2 and S3, because S3 was derived from same *Rank-8* users of S2. We have also tested this scheme with different k values for the initial S2 part (*i.e.* $k = 5, 15, 25, 35$). Figure C.4 shows the result obtained from this analysis for *Rank-1* identification.

Apart from choosing k random pairs, we have also tested with fixed sets of k pairs, where the selected pairs have the highest learning accuracy. We observed that the *Rank-1* accuracy was a little bit better for the random choice for each of the 4 classifiers (these improvements ranged from 0% to 2.1%). We can also fix these k pairs by applying an optimization algorithm (*i.e.* Genetic Algorithm or Particle Swarm Intelligence) with a development dataset, but unavailability of such a set prevented us from performing this analysis.



(a) Rank-1 Accuracy

(b) Rank-2 Accuracy

(c) Rank-4 Accuracy

(d) Rank-8 Accuracy

Figure C.1: Results obtained from S1 for different ranks *i.e.* different r values.

C.4.1.4 Discussion

We will discuss some observations made, based on the results of the analysis in this section. We would like to mention that all the analysis is done with the feature vector $f_i = (A_i, d_i)$ (*i.e.* only key press time of a particular key). We can see from the results that the ANN performance is low compared to the performance of the other classifiers, for all the schemes for any given scenario.

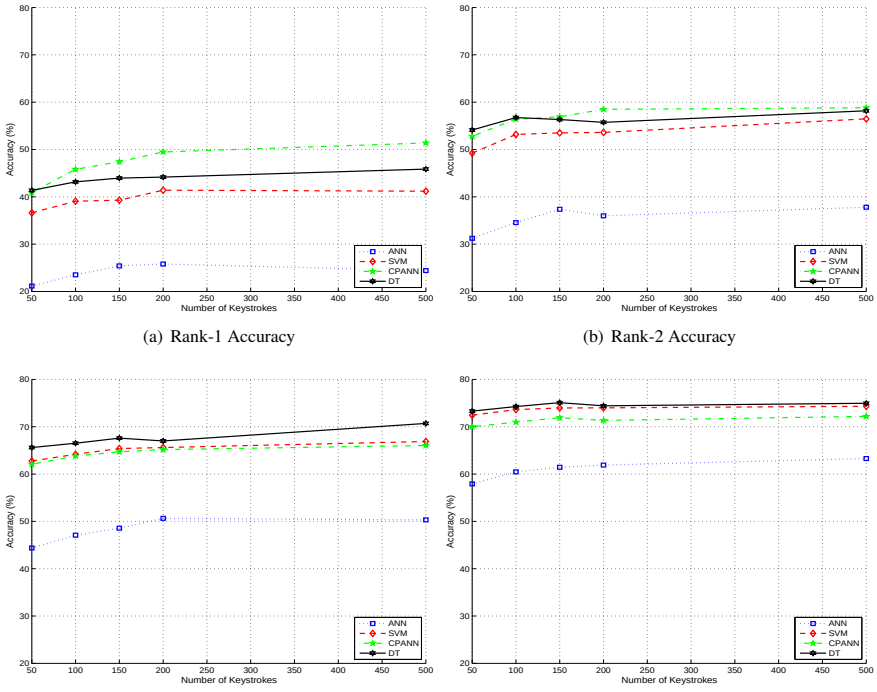
From the above analysis we can see that S1 performs better than S2, while the amount of work in S1 is also lower. For this reason we will not consider S2 for further analysis. When comparing S3 and S1, we note that the amount of work for S3 is higher than for S1, but the performance for S3 is similar to that of S1.

We can see that increasing the value of k will increase the recognition accuracy, but it will saturate after a certain value of k. Therefore, we have decided to use $k = 25$ for further analysis.

C.4.2 Feature Selection

In this section, we analyze various keystroke feature selection strategies for person identification. The details of the extracted features for this research can be found in Section C.3. We used three different feature sets in our analysis:

- **FE1:** In this setting, the feature vector will be $f_i = (A_i, d_i)$, *i.e.* the value and the duration;
- **FE2:** In this setting, the feature vector will be all the keystroke features extracted for our research *i.e.* $f_i = (A_i, A_{i+1}, d_i, l_i^{rp}, l_i^{rr}, l_i^{pp})$;



(a) Rank-1 Accuracy

(b) Rank-2 Accuracy

(c) Rank-4 Accuracy

(d) Rank-8 Accuracy

Figure C.2: Results obtained from S2 for different ranks *i.e.* different r values and $k = 25$.

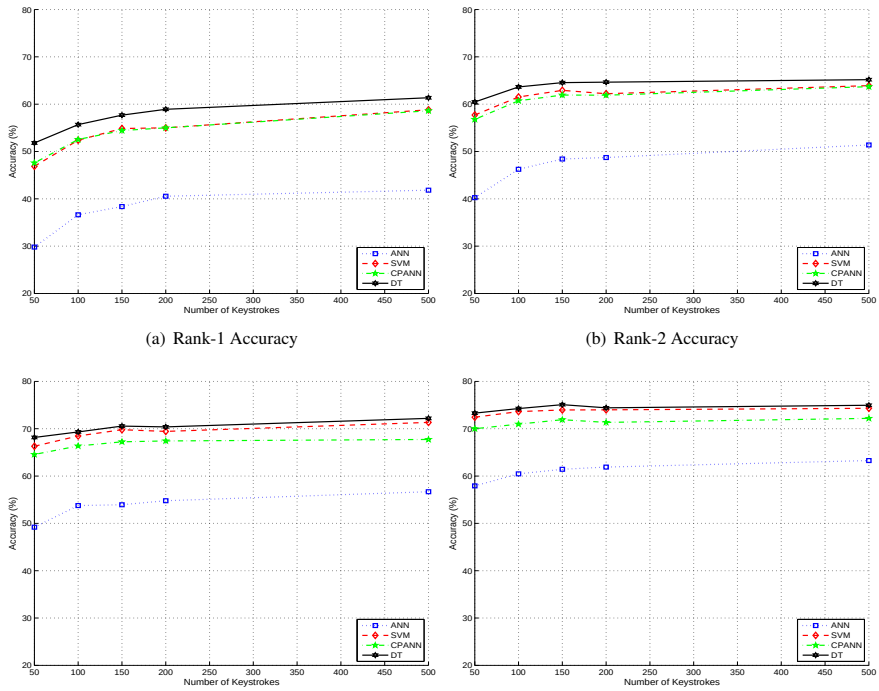
- **FE3:** In this setting, we have used the feature selection technique proposed by Ververidis *et al.* [148].

Figure C.5 shows the identification accuracy when using the above mentioned feature settings with the identification schemes S1 and S3. We clearly see from this figure that the FE1 feature setting performs better than the other settings for each of the classifiers with all different identification schemes. We can also observe that the DT classifier performs better than the other classifiers irrespective of the feature settings and identification schemes. When we use FE2 feature setting, we can see that SVM and CPANN classifiers failed unexpectedly for both identification schemes (see Figures C.5(c) and C.5(d)). In case of FE3, we can find large accuracy differences for S1 and S3 for the CPANN classifier by looking at the Figures C.5(e) and C.5(f).

We can conclude from the above analysis that FE1 is the most robust feature setting. Therefore, we will not consider feature settings FE2 and FE3 in our further analysis. We can also observe that the ANN classifier has the worst performance when using FE1 with identification schemes S1 and S3 (see Figures C.5(a) and C.5(b)). Therefore, we will also not consider the ANN classifier for further analysis.

C.4.3 Multi-Classifier Fusion (MCF)

In case of MCF, we denote by $(c^1, c^2, c^3) = (\lambda_p, \rho_p, \tau_p)$, where λ_p, ρ_p and τ_p the score values for CPANN, SVM and DT respectively (when using FE1 as a feature setting) for $p = 1, 2, ..., m$. Then the score value sc_j is a weighted sum of the three separate scores for Algorithms 10.1 and 10.3. In



(a) Rank-1 Accuracy

(b) Rank-2 Accuracy

(c) Rank-4 Accuracy

(d) Rank-8 Accuracy

Figure C.3: Results obtained from S3 for different ranks, where $c = 8$ and $k = 25$.

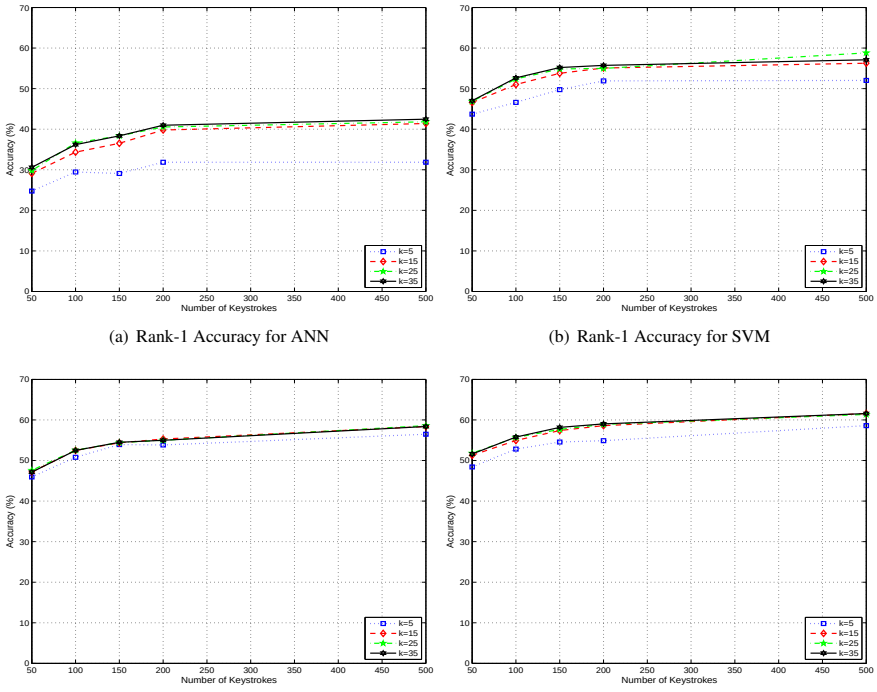
particular

$$sc_p = (\sum_{q=1}^{3} w_q c_p^q)/(\sum_{q=1}^{3} w_q)$$

where w_p denote the weights for the weighted fusion technique. We used the following different MCF settings for this analysis with S1 and S3:

- **ALL:** In this setting we included all three classifiers;
- **SVM-DT:** In this setting we excluded MCF without CPANN, *i.e.* $w_1 = 0$;
- **CPANN-DT:** In this setting we excluded MCF without SVM, *i.e.* $w_2 = 0$;
- **SVM-CPANN:** In this setting we excluded MCF without DT, *i.e.* $w_3 = 0$.

Figure C.6 shows the results obtained from the MCF analysis with the above mentioned settings from S1 and S3 schemes. We can see the clear improvement of the identification accuracy for MCF when compared to a single classifier by comparing Figures C.6(a) and C.5(a) and Figures C.6(b) and C.5(b). The S1 scheme is performing better than the S3 scheme for every MCF setting, which can be observed by comparing Figures C.6(a) and C.6(b). We also note that when we are using all three classifiers for MCF the performance was best. Therefore will we focus on the *ALL* MCF setting for the remainder of the analysis.



(a) Rank-1 Accuracy for ANN

(b) Rank-1 Accuracy for SVM

(c) Rank-1 Accuracy for CPANN

(d) Rank-1 Accuracy for DT

Figure C.4: Results obtained from S3 for different k values.

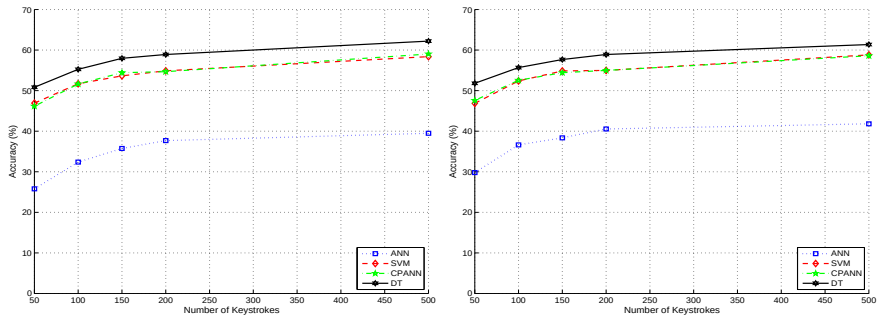
C.4.4 Typing Hand

In this section, we will analyze the identification accuracy for different handedness of the test samples for our best classification settings (*i.e.* FE1+ALL MCF) with S1 and S3. As mentioned earlier does the test dataset for our research consist of three different typing hand samples (see Section C.3 for more details).

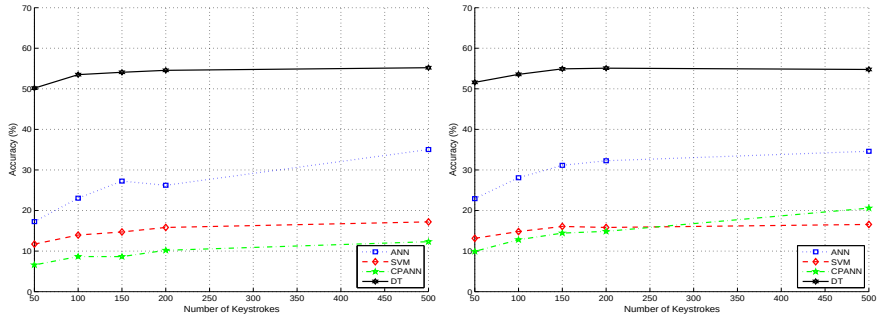
- **Both Hands:** In this analysis we only used the test samples that represent normal typing when using both hands. Figure C.7(a) shows the results obtained from this analysis and we can see that the performance of S1 and S3 are very similar.
- **Right Hand:** In this analysis we only used the test samples that are typed with the right hand. Figure C.7(b) shows the results we obtained from this analysis and we can see that S1 performs better than S3.
- **Left Hand:** In this analysis we only used the test samples that are typed with the left hand. Figure C.7(c) shows the results we obtained from this analysis, and similar to the *Right Hand* analysis we can again observe that S1 performs better than S3.

C.4.5 Comparison with previous research

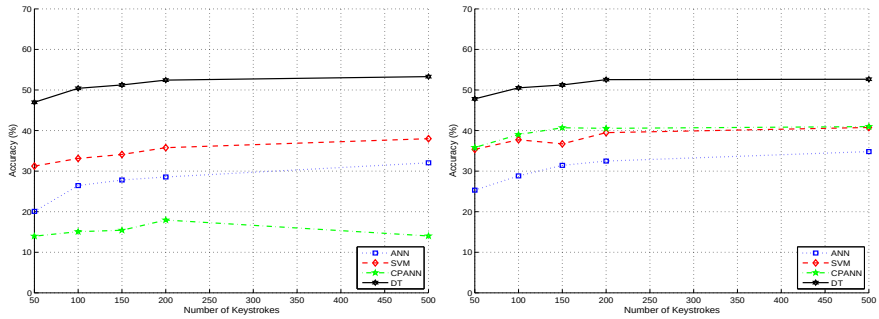
We are going to compare our research results with the results obtained from previous research done on the same dataset. As we have mentioned before, this dataset was used in the ”One-handed



(a) FE1 + S1 (b) FE1 + S3



(c) FE2 + S1 (d) FE2 + S3

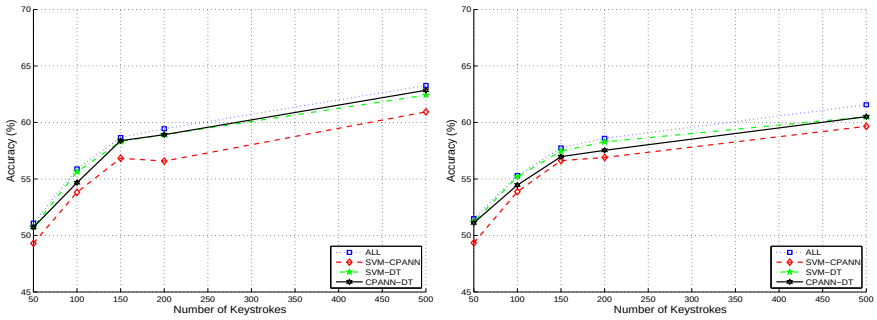


(e) FE3 + S1 (f) FE3 + S3

Figure C.5: Results obtained from the keystroke feature analysis.

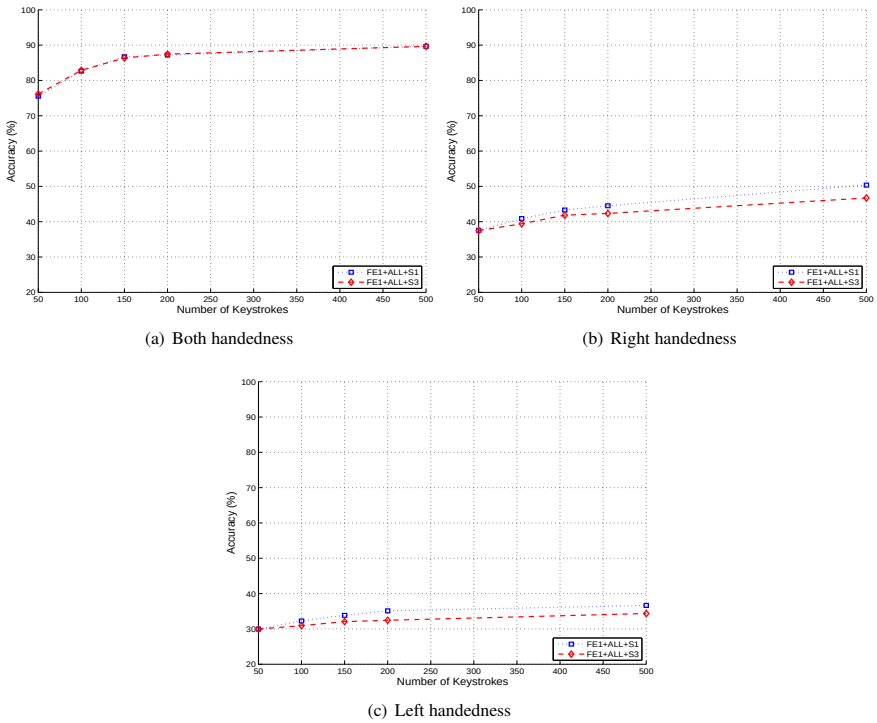
Keystroke Biometric Identification Competition" of the 8^{th} IAPR International Conference on Biometrics (ICB'15) with 500-keystrokes as a segment for evaluation (*i.e.* $m = 500$). We take the competition results [91] to compare with our results.

Table C.1 shows the comparison between our best identification accuracy with the top three positions in the competition. In this table, we can clearly see that both of our techniques perform better than the best position in the competition and the $[Baseline]$ performance for this dataset was obtained by applying a similar technique as proposed by Tappert *et al.* [144]. Both of our techniques achieved 89.7% identification rate for *Both Hands*, which is an improvement of 6.9%



(a) MCF + S1 (b) MCF + S3

Figure C.6: Result obtained for MCF with S1 and S3.



(a) Both handedness (b) Right handedness

(c) Left handedness

Figure C.7: Identification accuracy obtained from handedness experiment.

when compared to the best results in the competition in this category. In the *Left Hand* category, our first technique (*i.e.* FE1+ALL MCF+S1) achieved 36.6% accuracy, which is an improvement of 6.1% when compared to the best result in the competition in this category. Our second technique (*i.e.* FE1+ALL MCF+S3) achieved an accuracy of 34.4%, which is still almost 4% higher than the best results in [91]. Finally, for the *Right Hand* category, our first technique achieved 50.4% accuracy, which is the improvement of 10.2% when compared to the best result in the competition in this

Table C.1: Comparison with previous research.

Name	Both (%)	Left (%)	Right (%)
GUC-Norway	82.8	30.5	40.2
Sudalai Rajkumar	82.8	27.5	32.1
ATVS-Madrid	69.5	16.8	20.4
$[Baseline]$	61.1	6.2	9.5
Our (FE1+ALL MCF+S1)	**89.7**	**36.6**	**50.4**
Our (FE1+ALL MCF+S3)	**89.7**	**34.4**	**46.7**

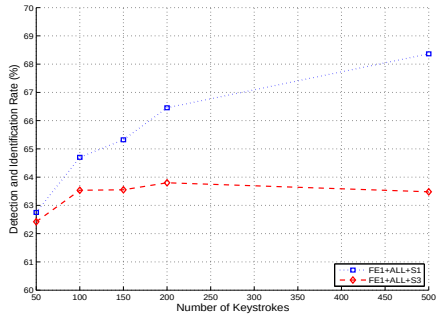


Figure C.8: Detection and Identification Rate (DIR) for S1 and S3.

category. Our second technique scored slightly lower than the first with a 46.7% accuracy.

C.4.6 Open-set Experiment

We also performed an open-set analysis, where 50% of the users is known to the system and the other 50% are completely unknown to the system. In this experiment, the set of users I for all the schemes will be $I = \{1, 2 \ldots \frac{N-1}{2}\}$ (see Algorithms 10.1 and 10.3). To measure the *Detection and Identification Rate (DIR)*, we used a threshold (*i.e.* T_{open}) that will decide whether the user is within the group of known users or not. If the $User_{score} \geq T_{open}$ (see Algorithms 10.1 or 10.3 for $User_{score}$) then we say that the user is within the set of known users, otherwise he/she is said to be an unknown user. If we find that the user is known to the system, then the system will establish the identity of the user. The DIR is the summation of the following two values:

- **True ID (TID):** Where $User_{score} \geq T_{open}$ *i.e.* adversary is within the known user set and correctly identified.
- **True Not In (TNotIn):** Where $User_{score} < T_{open}$ *i.e.* adversary was indeed not in the known user set.

Figure C.8 shows the DIR obtained from this analysis for S1 and S3 with FE1 and ALL MCF. We can see that the DIR is higher for S1 when compared to S3 for any given size of the keystroke block.

C.4.7 Discussion

We can summarize the findings from our research as follows:

- We can clearly see from our analysis that the keystroke duration of a given key is the most stable feature for user identification;

- We found that S1 is the most robust technique during our analysis. Also, this technique has a lower computational complexity than other proposed schemes;
- Of the four classifiers used in this research is DT is the most robust classifier on the given dataset;
- MCF can improve the identification accuracy;
- We can see that there is high recognition accuracy in case of normal both hands typing compared to single hand typing. Therefore, we can say that keystroke dynamics has the potential to identify a person, provided that the data is based on the normal typing behaviour of a user.
- We obtained an overall identification accuracy of 63.3% for the closed-set analysis (see ALL in Figure C.6(a)) and a DIR of 68.4% for the open-set analysis (see Figure C.8). These results were obtained without considering typing hand. If we consider the typing hand, then the identification accuracy is 89.7% for normal both hands typing.

C.5 Summary

In this research we have focused on identifying a person based on his typing behaviour. We used three identification schemes with PUC and shown that scheme S1 gives the best results. We have shown that duration features are more stable than a combination of durations and latencies.

Finally, a conclusion that can be drawn from this research is that identification of a person when he/she is using both hands performs rather well *i.e.* normal users typing behaviour (approximately 90% accuracy). However, the same models cannot be used to then identify a person with a high accuracy when only one hand is used for typing. It is obvious that typing with only one hand will influence the typing behaviour, but despite this we can still identify users with 36.6% accuracy when they use the left hand and 50.4% accuracy when they use the right hand. The better performance for the right hand is most likely due to the fact that most people are right-handed and hence typing with the right hand resembles the natural typing behaviour better than typing with the left hand. We also performed an open-set analysis where 50% of the users are unknown to the system and obtained DIR of 68.4%.

The objective of using typing behaviour to identify a person is to use it as tool for cyber-forensics. In the future we plan to perform an experiment on real world cyber-forensics data and investigate how it can be used as forensics evidence in court. In the future we will also investigate how well we can improve the performance for one handed typing. A first step will be to determine, based on the typing sample if the user typed with one or two hands, which might most likely be done easiest by considering total typing time.

Bibliography

[1] ACHARYA, S., FRIDMAN, A., BRENNAN, P., JUOLA, P., GREENSTADT, R., AND KAM, M. User authentication through biometric sensors and decision fusion. In *47th Annual Conf. on Information Sciences and Systems* (2013), IEEE, pp. 1–6. 44, 79, 86, 89, 147

[2] AHMED, A., AND TRAORE, I. A new biometric technology based on mouse dynamics. *IEEE Transactions on Dependable and Secure Computing 4*, 3 (2007), 165–179. 15, 16, 40, 66

[3] AHMED, A. A., AND TRAORE, I. Biometric recognition based on free-text keystroke dynamics. *IEEE Transactions on Cybernetics 44*, 4 (2014), 458–472. 15, 16, 29, 44, 172

[4] AL SOLAMI, E., BOYD, C., CLARK, A., AND AHMED, I. User-representative feature selection for keystroke dynamics. In *5th Int. Conf. on Network and System Security (NSS'11)* (2011), pp. 229–233. 15, 16

[5] ALEXANDER, J., COCKBURN, A., AND LOBB, R. Appmonitor: A tool for recording user actions in unmodified windows applications. *Behavior Research Methods 40*, 2 (2008), 413–421. 147, 158

[6] ALI, T. *Biometric score calibration for forensic face recognition*. Ph.D. thesis, University of Twente, 2014. 144

[7] ANTAL, M., BOKOR, Z., AND SZABÓ, L. Z. Information revealed from scrolling interactions on mobile devices. *Pattern Recognition Letters 56* (2015), 7 – 13. 45, 46, 47, 130

[8] ARAUJO, L., SUCUPIRA, L.H.R., J., LIZARRAGA, M., LING, L., AND YABU-UTI, J. B. T. User authentication through typing biometrics features. *IEEE Transactions on Signal Processing 53*, 2 (2005), 851–855. 44, 172

[9] AZZINI, A., MARRARA, S., SASSI, R., AND SCOTTI, F. A fuzzy approach to multimodal biometric continuous authentication. *Fuzzy Optimization and Decision Making 7*, 3 (2008), 243–256. 17

[10] BAILEY, K. O., OKOLICA, J. S., AND PETERSON, G. L. User identification and authentication using multi-modal behavioral biometrics. *Computers & Security 43* (2014), 77 – 89. 15, 16, 44, 79, 86, 89, 147

[11] BANERJEE, S., AND WOODARD, D. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research 7* (2012), 116–139. 12, 161, 171, 172

[12] BARROS, R., BASGALUPP, M., DE CARVALHO, A., AND FREITAS, A. A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42*, 3 (2012), 291–312. 111

[13] BERGADANO, F., GUNETTI, D., AND PICARDI, C. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security 5*, 4 (2002), 367–397. 44

[14] BERGADANO, F., GUNETTI, D., AND PICARDI, C. Identity verification through dynamic keystroke analysis. *Intelligent Data Analysis 7*, 5 (2003), 469–496. 171

[15] BERGNER, R. M. What is behavior? and so what? *New Ideas in Psychology 29*, 2 (2011), 147 – 155. 21, 30

[16] BISHOP, C. M. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995. 51

[17] BO, C., ZHANG, L., JUNG, T., HAN, J., LI, X.-Y., AND WANG, Y. Continuous user identification via touch and movement behavioral biometrics. In *2014 IEEE Int. Performance Computing and Communications Conference* (2014), pp. 1–8. 16, 17

[18] BOLLE, R. M., RATHA, N. K., AND PANKANTI, S. Error analysis of pattern recognition systems - the subsets bootstrap. *Computer Vision and Image Understanding 93*, 1 (2004), 1 – 33. 30

[19] BOURS, P. Continuous keystroke dynamics: A different perspective towards biometric evaluation. *Information Security Technical Report 17* (2012), 36–43. 15, 16, 17, 22, 23, 29, 31, 171, 172

[20] BOURS, P., AND BARGHOUTHI, H. Continuous authentication using biometric keystroke dynamics. In *Norwegian Information Securiry Conference (NISK'09)* (2009), pp. 1–7. 31

[21] BOURS, P., AND MONDAL, S. *Continuous Authentication with Keystroke Dynamics*. Science Gate Publishing, 2015, ch. Recent Advances in User Authentication Using Keystroke Dynamics Biometrics, pp. 41–58. 6, 67

[22] BOURS, P., AND MONDAL, S. Performance evaluation of continuous authentication systems. *IET Biometrics* (2015), 1–7. 6, 29

[23] BRENNAN, M., AFROZ, S., AND GREENSTADT, R. Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. *ACM Transactions on Information and System Security 15*, 3 (2012), 12:1–12:22. 144

[24] BROCARDO, M., TRAORE, I., AND WOUNGANG, I. Toward a framework for continuous authentication using stylometry. In *IEEE 28th Int. Conf. on Advanced Information Networking and Applications (AINA'14)* (2014), pp. 106–115. 144

[25] BURGES, C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery 2*, 2 (1998), 121–167. 51

[26] CAI, Z., SHEN, C., WANG, M., SONG, Y., AND WANG, J. Mobile authentication through touch-behavior features. In *Biometric Recognition*, vol. 8232 of *Lecture Notes in Computer Science*. Springer, 2013, pp. 386–393. 14

[27] CHANG, C.-C., AND LIN, C.-J. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2*, 3 (2011), 27:1–27:27. 51

[28] CLARKE, N., AND FURNELL, S. Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security 6*, 1 (2007), 1–14. 14

[29] COLLINS, J., AND STEWART, I. Coupled nonlinear oscillators and the symmetries of animal gaits. *Journal of Nonlinear Science 3*, 1 (1993), 349–392. 142

[30] CROUSE, D., HAN, H., CHANDRA, D., BARBELLO, B., AND JAIN, A. K. Continuous authentication of mobile user: Fusion of face image and inertial measurement unit data. In *Int. Conf. on Biometrics (ICB'15)* (2015), pp. 135–142. 17

[31] DAVOUDI, H., AND KABIR, E. A new distance measure for free text keystroke authentication. In *14th Int. CSI Computer Conference (CSICC'09)* (2009), IEEE, pp. 570–575. 15, 16

[32] DAVOUDI, H., AND KABIR, E. Modification of the relative distance for free text keystroke authentication. In *5th Int. Symposium on Telecommunications (IST'10)* (2010), pp. 547–551. 15, 16

[33] DE LUCA, A., HANG, A., BRUDY, F., LINDNER, C., AND HUSSMANN, H. Touch me once and i know it's you!: Implicit authentication based on touch screen patterns. In *SIGCHI Conf. on Human Factors in Computing Systems* (2012), CHI '12, ACM, pp. 987–996. 14

[34] DECANN, B., AND ROSS, A. "has this person been encountered before?": Modeling an anonymous identification system. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW'12)* (2012), pp. 89–96. 30

[35] DECANN, B., AND ROSS, A. De-duplication errors in a biometric system: An investigative study. In *IEEE Int. Workshop on Information Forensics and Security (WIFS'13)* (2013), pp. 43–48. 30

[36] DEUTSCHMANN, I., AND LINDHOLM, J. Behavioral biometrics for darpa's active authentication program. In *IEEE Int. Conf. of the Biometrics Special Interest Group (BIOSIG'13)* (2013), IEEE, pp. 1–8. 23

[37] DOWLAND, P. S., AND FURNELL, S. M. A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In *Security and Protection in Information Processing Systems*, vol. 147 of *IFIP The International Federation for Information Processing*. Springer US, 2004, pp. 275–289. 15, 16, 29, 43, 44, 172

[38] EPP, C., LIPPOLD, M., AND MANDRYK, R. Identifying emotional states using keystroke dynamics. In *The 2011 Annual Conf. on Human Factors in Computing Systems (CHI'11)* (2011), pp. 715–724. 143

[39] EVERITT, R., AND MCOWAN, P. W. Java-based internet biometric authentication system. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 9 (2003), 1166–1172. 13

[40] FATHY, M. E., PATEL, V. M., YEH, T., ZHANG, Y., CHELLAPPA, R., AND DAVIS, L. S. Screen-based active user authentication. *Pattern Recognition Letters 42* (2014), 122 – 127. 17

[41] FEHER, C., ELOVICI, Y., MOSKOVITCH, R., ROKACH, L., AND SCHCLAR, A. User identity verification via mouse dynamics. *Information Sciences 201*, 0 (2012), 19 – 36. 15, 16, 29, 36, 41, 44, 45

[42] FENG, T., YANG, J., YAN, Z., TAPIA, E. M., AND SHI, W. Tips: Context-aware implicit user identification using touch screen in uncontrolled environments. In *15th Workshop on Mobile Computing Systems and Applications* (2014), ACM, pp. 9:1–9:6. 16, 17

[43] FENG, T., ZHAO, X., CARBUNAR, B., AND SHI, W. Continuous mobile authentication using virtual key typing biometrics. In *12th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications (TRUSTCOM '13)* (2013), IEEE, pp. 1547–1552. 16, 17

[44] FERREIRA, J., AND SANTOS, H. Keystroke dynamics for continuous access control enforcement. In *Int. Conf. on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC'12)* (2012), pp. 216–223. 15, 16

[45] FILHO, J. R. M., AND FREIRE, E. O. On the equalization of keystroke timing histograms. *Pattern Recognition Letters 27*, 13 (2006), 1440–1446. 15, 16

[46] FRANK, M., BIEDERT, R., MA, E., MARTINOVIC, I., AND SONG, D. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE Transactions on Information Forensics and Security 8*, 1 (2013), 136–148. 16, 17, 45, 48, 95

[47] FRIDMAN, L., STOLERMAN, A., ACHARYA, S., BRENNAN, P., JUOLA, P., GREENSTADT, R., AND KAM, M. Multi-modal decision fusion for continuous authentication. *Computers & Electrical Engineering 41* (2015), 142 – 156. 15, 16

[48] FURNELL, S. M., MORRISSEY, J. P., SANDERS, P. W., AND STOCKEL, C. T. Applications of keystroke analysis for improved login security and continuous user authentication. In *Information Systems Security*, S. K. Katsikas and D. Gritzalis, Eds. Chapman & Hall, Ltd., 1996, pp. 283–294. 15, 16

[49] GAINES, R. S., LISOWSKI, W., PRESS, S. J., AND SHAPIRO, N. Authentication by keystroke timing: Some preliminary results. Tech. rep., RAND Corporation, 1980. 12, 14

[50] GAMBOA, H., AND FERREIRA, V. Widam - web interaction display and monitoring. In *Proc. 5th Int. Conf. on Enterprise Information Systems* (2003), pp. 1–7. 147, 158

[51] GAMBOA, H., AND FRED, A. A behavioral biometric system based on human-computer interaction. *Proceedings of SPIE, Biometric Technology for Human Identification 5404* (2004), 381–392. 13

[52] GAMBOA, H., FRED, A. L. N., AND JAIN, A. K. Webbiometrics: User verification via web interaction. In *Biometrics Symposium, Biometric Consortium Conference* (2007), IEEE, pp. 1–6. 15, 16

[53] GARG, A., VIDYARAMAN, S., UPADHYAYA, S., AND KWIAT, K. Usim: a user behavior simulation framework for training and testing idses in gui based systems. In *39th Annual Simulation Symposium* (2006), pp. 1–8. 147, 158

[54] GIOT, R., DORIZZI, B., AND ROSENBERGER, C. Analysis of template update strategies for keystroke dynamics. In *IEEE Workshop on Computational Intelligence in Biometrics and Identity Management (CIBIM'11)* (2011), pp. 21–28. 141

[55] GIOT, R., NINASSI, A., EL-ABED, M., AND ROSENBERGER, C. Analysis of the acquisition process for keystroke dynamics. In *Int. Conf. of the Biometrics Special Interest Group (BIOSIG'12)* (2012), pp. 1 –6. 162, 165

[56] GOVINDARAJAN, S., GASTI, P., AND BALAGANI, K. Secure privacy-preserving protocols for outsourcing continuous authentication of smartphone users with touch data. In *IEEE 6th Int. Conf. on Biometrics: Theory, Applications and Systems (BTAS'13)* (2013), pp. 1–8. 16, 17

[57] GUENNOUN, M., ABBAD, N., TALOM, J., RAHMAN, M., AND EL-KHATIB, K. Continuous authentication by electrocardiogram data. In *IEEE Toronto Int. Conf. Science and Technology for Humanity (TIC-STH'09)* (2009), pp. 40–42. 17

[58] GUNETTI, D., AND PICARDI, C. Keystroke analysis of free text. *ACM Transactions on Information and System Security 8*, 3 (2005), 312–347. 15, 16

[59] HARUN, N., WOO, W. L., AND DLAY, S. S. Performance of keystroke biometrics authentication system using artificial neural network (ann) and distance classifier method. In *Int. Conf. on Computer and Communication Engineering (ICCCE'10)* (2010), IEEE, pp. 1–6. 15, 16

[60] HEMPSTALK, K. *Continuous typist verification using machine learning*. Ph.D. thesis, The University of Waikato, 2009. 15, 16

[61] HODGE, V., AND AUSTIN, J. A survey of outlier detection methodologies. *Artificial Intelligence Review 22*, 2 (2004), 85–126. 143

[62] ISO. ISO 8601:2004 Data elements and interchange formats – Information interchange – Representation of dates and times. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=40874, 2004. 156, 157

[63] JAGADEESAN, H., AND HSIAO, M. A novel approach to design of user re-authentication systems. In *IEEE 3rd Int. Conf. on Biometrics: Theory, Applications, and Systems, (BTAS'09)* (2009), pp. 1–6. 15, 16, 44, 79, 86, 89, 147

[64] JAIN, A., DUIN, R. P. W., AND MAO, J. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 1 (2000), 4–37. 52

[65] JAIN, A., ROSS, A., AND PRABHAKAR, S. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology 14*, 1 (2004), 4–20. 10, 11, 12

[66] JANAKIRAMAN, R., AND SIM, T. Keystroke dynamics in a general setting. In *Advances in Biometrics*, vol. 4642 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 584–593. 15, 16, 162

[67] JOHNSON, P., LAZARICK, R., MARASCO, E., NEWTON, E., ROSS, A., AND SCHUCKERS, S. Emerging performance metrics for liveness detection in biometric systems. Tech. rep., Biometrics Metrics Report v3.0 - West Point, 2012. 11

[68] JONES, M., AND MEWHORT, D. J. K. Case-sensitive letter and bigram frequency counts from large-scale english corpora. *Behavior research methods, instruments, & computers 36*, 3 (2004), 388–396. 163

[69] KANG, P., AND CHO, S. Keystroke dynamics-based user authentication using long and free text strings from various input devices. *Information Sciences 308* (2015), 72 – 93. 15, 16

[70] KARNAN, M., AKILA, M., AND KRISHNARAJ, N. Biometric personal authentication using keystroke dynamics: A review. *Applied Soft Computing 11*, 2 (2011), 1565 – 1573. 161, 171

[71] KHAN, H., AND HENGARTNER, U. Towards application-centric implicit authentication on smartphones. In *15th Workshop on Mobile Computing Systems and Applications* (2014), ACM, pp. 10:1–10:6. 97

[72] KHAN, M., TSAI, P.-W., PAN, J.-S., AND LIAO, B.-Y. Biometric driven initiative system for passive continuous authentication. In *7th Int. Conf. on Information Assurance and Security (IAS'11)* (2011), pp. 139–144. 17

[73] KILLOURHY, K., AND MAXION, R. Comparing anomaly detectors for keystroke dynamics. In *39th Annual Int. Conf. on Dependable Systems and Networks* (2009), IEEE, pp. 125–134. 161

[74] KIM, S., CHA, S.-H., MONACO, J., AND TAPPERT, C. A correlation method for handling infrequent data in keystroke biometric systems. In *Int. Workshop on Biometrics and Forensics (IWBF'14)* (2014), pp. 1–6. 142

[75] KITTLER, J., HATEF, M., DUIN, R. P. W., AND MATAS, J. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 3 (1998), 226–239. 52, 123

[76] KOLAKOWSKA, A. User authentication based on keystroke dynamics analysis. In *Computer Recognition Systems 4*, vol. 95 of *Advances in Intelligent and Soft Computing*. Springer Berlin Heidelberg, 2011, pp. 667–675. 15, 16

[77] KUKREJA, U., STEVENSON, W., AND RITTER, F. Rui: Recording user input from interfaces under windows and mac os x. *Behavior Research Methods 38*, 4 (2006), 656–659. 147, 153, 158

[78] LAZAR, C., TAMINAU, J., MEGANCK, S., STEENHOFF, D., COLETTA, A., MOLTER, C., DE SCHAETZEN, V., DUQUE, R., BERSINI, H., AND NOWE, A. A survey on filter techniques for feature selection in gene expression microarray analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 9*, 4 (2012), 1106–1119. 78

[79] LEHMANN, E. L., AND ROMANO, J. P. *Testing statistical hypotheses*. Springer Texts in Statistics. Springer, 2005. 63, 95

[80] LI, L., ZHAO, X., AND XUE, G. Unobservable re-authentication for smartphones. In *20th Annual Network & Distributed System Security Symposium* (2013), The Internet Society, pp. 1–16. 16, 17

[81] LIN, C.-C., CHANG, C.-C., AND LIANG, D. A new non-intrusive authentication approach for data protection based on mouse dynamics. In *Int. Symposium on Biometrics and Security Technologies (ISBAST'12)* (2012), IEEE, pp. 9–14. 15, 16

[82] LOCKLEAR, H., GOVINDARAJAN, S., SITOVA, Z., GOODKIND, A., BRIZAN, D. G., ROSENBERG, A., PHOHA, V. V., GASTI, P., AND BALAGANI, K. S. Continuous authentication with cognition-centric text production and revision features. In *IEEE Int. Joint Conf. on Biometrics (IJCB'14)* (2014), pp. 1–8. 15, 16

[83] MACKENZIE, I. S., KAUPPINEN, T., AND SILFVERBERG, M. Accuracy measures for evaluating computer pointing devices. In *The SIGCHI Conf. on Human Factors in Computing Systems (CHI '01)* (2001), ACM, pp. 9–16. 45

[84] MELSSEN, W., WEHRENS, R., AND BUYDENS, L. Supervised kohonen networks for classification problems. *Chemometrics and Intelligent Laboratory Systems 83*, 2 (2006), 99 – 113. 69

[85] MENEZES, A. J., OORSCHOT, P. C. V., AND VANSTONE, S. A. *Handbook of Applied Cryptography*, 5th ed. CRC Press, Inc., 2001. 153

[86] MENG, Y., WONG, D. S., AND KWOK, L.-F. Design of touch dynamics based user authentication with an adaptive mechanism on mobile phones. In *29th Annual ACM Symposium on Applied Computing (SAC '14)* (2014), ACM, pp. 1680–1687. 14

[87] MESSERMAN, A., MUSTAFIC, T., CAMTEPE, S. A., AND ALBAYRAK, S. Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics. In *Int. Joint Conf. on Biometrics (IJCB'11)* (2011), IEEE, pp. 1–8. 15, 16, 29, 44

[88] MITCHELL, M. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998. 71

[89] MONACO, J., BAKELMAN, N., CHA, S.-H., AND TAPPERT, C. Recent advances in the development of a long-text-input keystroke biometric authentication system for arbitrary text input. In *European Intelligence and Security Informatics Conference (EISIC'13)* (2013), pp. 60–66. 15, 16

[90] MONACO, J. V., BAKELMAN, N., CHA, S.-H., AND TAPPERT, C. C. Developing a keystroke biometric system for continual authentication of computer users. In *European Intelligence and Security Informatics Conference (EISIC'12)* (2012), IEEE, pp. 210–216. 29, 44

[91] MONACO, J. V., PEREZ, G., TAPPERT, C. C., BOURS, P., MONDAL, S., RAJKUMAR, S., MORALES, A., FIERREZ, J., AND ORTEGA-GARCIA, J. One-handed keystroke biometric identification competition. In *Int. Conf. on Biometrics (ICB'15)* (2015), IEEE, p. 7. 172, 173, 178, 179

[92] MONDAL, S., AND BOURS, P. Continuous authentication using behavioural biometrics. In *Collaborative European Research Conference (CERC'13)* (2013), pp. 130–140. 7, 9

[93] MONDAL, S., AND BOURS, P. Continuous authentication using mouse dynamics. In *Int. Conf. of the Biometrics Special Interest Group (BIOSIG'13)* (2013), IEEE, pp. 1–12. 7, 33, 51

[94] MONDAL, S., AND BOURS, P. Continuous authentication using fuzzy logic. In *7th Int. Conf. on Security of Information and Networks (SIN'14)* (2014), ACM, pp. 231–238. 7, 67

[95] MONDAL, S., AND BOURS, P. A computational approach to the continuous authentication biometric system. *Information Sciences 304* (2015), 28 – 53. 5, 21, 51

[96] MONDAL, S., AND BOURS, P. Context independent continuous authentication using behavioural biometrics. In *IEEE Int. Conf. on Identity, Security and Behavior Analysis (ISBA'15)* (2015), IEEE, pp. 1–8. 7, 75

[97] MONDAL, S., AND BOURS, P. Continuous authentication and identification for mobile devices: Combining security and forensics. In *7th IEEE Int. Workshop on Information Forensics and Security (WIFS'15)* (2015), IEEE, pp. 1–6. 8, 123

[98] MONDAL, S., AND BOURS, P. Continuous authentication in a real world settings. In *8th Int. Conf. on Advances in Pattern Recognition (ICAPR'15)* (2015), IEEE, pp. 1–6. 7, 75

[99] MONDAL, S., AND BOURS, P. Does context matter for the performance of continuous authentication biometric systems? an empirical study on mobile devices. In *Int. Conf. of the Biometrics Special Interest Group (BIOSIG'15)* (2015), IEEE, pp. 1–5. 7, 91

[100] MONDAL, S., AND BOURS, P. Swipe gesture based continuous authentication for mobile devices. In *Int. Conf. on Biometrics (ICB'15)* (2015), IEEE, pp. 458–465. 7, 91

[101] MONDAL, S., AND BOURS, P. Combining keystroke and mouse dynamics for continuous user authentication and identification. In *IEEE Int. Conf. on Identity, Security and Behavior Analysis (ISBA'16)* (2016), IEEE, pp. 1–8. 8, 111

[102] MONDAL, S., AND BOURS, P. Continuous user authentication and adversary identification: Combining security & forensics. Under Review in IEEE Transactions on Information Forensics & Security, 2016. 6, 101, 123

[103] MONDAL, S., AND BOURS, P. Person identification by keystroke dynamics using pairwise user coupling. Under Review in IEEE Transactions on Dependable and Secure Computing, 2016. 6, 171

[104] MONDAL, S., AND BOURS, P. A study on continuous authentication using a combination of keystroke and mouse biometrics. Under Review in Neurocomputing, 2016. 6, 75

[105] MONDAL, S., BOURS, P., AND IDRUS, S. Z. S. Complexity measurement of a password for keystroke dynamics: Preliminary study. In *6th Int. Conf. on Security of Information and Networks (SIN'13)* (2013), ACM, pp. 301–305. 7, 161

[106] MONDAL, S., BOURS, P., JOHANSEN, L., STENVI, R., AND ØVERBØ, M. *Importance of a Versatile Logging Tool for Behavioural Biometrics and Continuous Authentication Research*. IGI Global, 2015, ch. Handbook of Research on Homeland Security Threats and Countermeasures. 6, 147

[107] MONROSE, F., AND RUBIN, A. Authentication via keystroke dynamics. In *4th ACM Conf. on Computer and Communications Security (CCS'97)* (1997), ACM, pp. 48–56. 15, 16

[108] MONROSE, F., AND RUBIN, A. D. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems 16*, 4 (2000), 351–359. 161

[109] MOTLAGH, M. R. M. Continuous user identification. Master's thesis, Gjøvik University College, 2015. 143

[110] MUNCASTER, J., AND TURK, M. Continuous multimodal authentication using dynamic bayesian networks. In *2nd Workshop on Multimodal User Authentication* (2006), pp. 1–8. 17

[111] NABNEY, I. T. *Netlab: Algorithms for Pattern Recognition*, vol. XVIII of *Advances in Computer Vision and Pattern Recognition*. Springer, 2002. 51

[112] NAKKABI, Y., TRAOR, I., AND AHMED, A. Improving mouse dynamics biometric performance using variance reduction via extractors with separate features. *IEEE Transactions on Systems, Man And Cybernetics - Part A: Systems and Humans 40* (2010), 1345–1353. 15, 16, 29, 36, 39, 40, 43, 44, 66

[113] NGUYEN, T. T., LE, T. H., AND LE, B. H. Keystroke dynamics extraction by independent component analysis and bio-matrix for user authentication. In *11th Pacific Rim Int. Conf. on Artificial Intelligence (PRICAI'10)*, vol. 6230 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 477–486. 12

[114] NIINUMA, K., PARK, U., AND JAIN, A. Soft biometric traits for continuous user authentication. *IEEE Transactions on Information Forensics and Security 5*, 4 (2010), 771–780. 17

[115] NORTH, C. H., JOINES, J. A., KAY, M. G., HOUCK, C. R., AND HOUCK, C. R. A genetic algorithm for function optimization: A matlab implementation. Tech. rep., 1996. 80

[116] OBAIDAT, M., AND MACCHIAROLO, D. An online neural network system for computer access security. *IEEE Transactions on Industrial Electronics 40*, 2 (1993), 235–242. 171, 172

[117] ORDAL, P., GANZHORN, D., LU, D. V., FONG, W., AND NORWOOD, J. B. Continuous identity verification through keyboard biometrics. *Journal of Undergraduate Research 4*, 1 (2005), 20–24. 15, 16, 29, 44

[118] PARK, S., PARK, J., AND CHO, S. User authentication based on keystroke analysis of long free texts with a reduced number of features. In *Second Int. Conf. on Communication Systems, Networks and Applications (ICCSNA'10)* (2010), vol. 1, pp. 433–435. 15, 16

[119] POH, N., MARTIN, A., AND BENGIO, S. Performance generalization in biometric authentication using joint user-specific and sample bootstraps. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 3 (2007), 492–498. 30

[120] POUTSMA, A. Applying monte carlo techniques to language identification. *Language and Computers 45*, 1 (2002), 179–189. 144

[121] PUSARA, M. *An Examination of User Behavior for Re-Authentication*. Ph.D. thesis, CERIAS, Purdue University, 2007. 15, 16, 63, 84, 143

[122] RAHMAN, K., BALAGANI, K., AND PHOHA, V. Making impostor pass rates meaningless: A case of snoop-forge-replay attack on continuous cyber-behavioral verification with keystrokes. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW'11)* (2011), pp. 31–38. 15, 16

[123] RAHMAN, K., BALAGANI, K., AND PHOHA, V. Snoop-forge-replay attacks on continuous verification with keystrokes. *IEEE Transactions on Information Forensics and Security 8*, 3 (2013), 528–541. 142

[124] RAMIM, M. M., AND LEVY, Y. Towards a framework of biometric exam authentication in e-learning environments. In *18th Int. Conf. on Information Resources Management Association (IRMA'07)* (2007). 13

[125] RAMOS-CASTRO, D., GONZALEZ-RODRIGUEZ, J., CHAMPOD, C., FIERREZ-AGUILAR, J., AND ORTEGA-GARCIA, J. Between-source modelling for likelihood ratio computation in forensic biometric recognition. In *Audio- and Video-Based Biometric Person Authentication*, T. Kanade, A. Jain, and N. Ratha, Eds., vol. 3546 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 1080–1089. 144

[126] ROSS, A., AND JAIN, A. Information fusion in biometrics. *Pattern Recognition Letters 24*, 13 (2003), 2115 – 2125. 9, 15, 52, 54

[127] ROTH, J., LIU, X., AND METAXAS, D. On continuous user authentication via typing behavior. *IEEE Transactions on Image Processing 23*, 10 (2014), 4611–4624. 17

[128] ROY, A., HALEVI, T., AND MEMON, N. An hmm-based behavior modeling approach for continuous mobile authentication. In *2014 IEEE Int. Conf. on Acoustics, Speech and Signal Processing* (2014), pp. 3789–3793. 95

[129] SAEVANEE, H., CLARKE, N., FURNELL, S., AND BISCIONE, V. Text-based active authentication for mobile devices. In *ICT Systems Security and Privacy Protection*, vol. 428. Springer Berlin Heidelberg, 2014, pp. 99–112. 16, 17

[130] SAYED, B., TRAORE, I., WOUNGANG, I., AND OBAIDAT, M. Biometric authentication using mouse gesture dynamics. *IEEE Systems Journal 7*, 2 (2013), 262–274. 45

[131] SCHULZ, D. A. Mouse curve biometrics. In *Biometrics Symposium, Biometric Consortium Conference* (2006), IEEE, pp. 1–6. 15, 16

[132] SERWADDA, A., PHOHA, V., AND WANG, Z. Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms. In *2013 IEEE 6th Int. Conf. on Biometrics: Theory, Applications and Systems (BTAS'13)* (2013), pp. 1–8. 14, 16, 17

[133] SHEN, C., CAI, Z., AND GUAN, X. Continuous authentication for mouse dynamics: A pattern-growth approach. In *42nd Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN'12)* (2012), pp. 1–12. 15, 16, 29, 44

[134] SHEN, C., CAI, Z., GUAN, X., SHA, H., AND DU, J. Feature analysis of mouse dynamics in identity authentication and monitoring. In *IEEE Int. Conf. on Communications (ICC'09)* (2009), pp. 1–5. 13

[135] SHEN, C., ZHANG, Y., CAI, Z., YU, T., AND GUAN, X. Touch-interaction behavior for continuous user authentication on smartphones. In *Int. Conf. on Biometrics (ICB'15)* (2015), pp. 157–162. 16, 17

[136] SHEPHERD, S. Continuous authentication by analysis of keyboard typing characteristics. In *European Convention on Security and Detection* (1995), pp. 111–114. 14, 172

[137] SHIMSHON, T., MOSKOVITCH, R., ROKACH, L., AND ELOVICI, Y. Continuous verification using keystroke dynamics. In *Int. Conf. on Computational Intelligence and Security (CIS'10)* (2010), pp. 411–415. 15, 16

[138] SIM, T., AND JANAKIRAMAN, R. Are digraphs good for free-text keystroke dynamics? In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'07)* (2007), IEEE, pp. 1–6. 15, 16

[139] SIM, T., ZHANG, S., JANAKIRAMAN, R., AND KUMAR, S. Continuous verification using multimodal biometrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 4 (2007), 687–700. 17

[140] SMIRNOV, N. Approximate distribution laws for random variables, constructed from empirical data. *Uspekhi Matematicheskikh Nauk 10* (1944), 179–206. 78

[141] SNELICK, R., ULUDAG, U., MINK, A., INDOVINA, M., AND JAIN, A. Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 3 (2005), 450–455. 52, 54

[142] STEWART, J., MONACO, J., CHA, S.-H., AND TAPPERT, C. An investigation of keystroke and stylometry traits for authenticating online test takers. In *Int. Joint Conf. on Biometrics (IJCB'11)* (2011), pp. 1–7. 15, 16

[143] TANG, L., YANG, C., AND WU, Z. User authentication based on force-sensing keypad by fuzzy c-means clustering method. *Journal of Computational Information Systems 6*, 11 (2010), 3659 – 3667. 12

[144] TAPPERT, C. C., VILLANI, M., AND CHA, S.-H. Keystroke biometric identification and authentication on long-text input. *Behavioral Biometrics for Human Identification: Intelligent Applications* (2010), 342–367. 171, 172, 178

[145] TRAORE, I., AND AHMED, A. A. E., Eds. *Continuous Authentication Using Biometrics: Data, Models, and Metrics*. IGI Global, 2012. 9

[146] TRAORE, I., WOUNGANG, I., OBAIDAT, M., NAKKABI, Y., AND LAI, I. Combining mouse and keystroke dynamics biometrics for risk-based authentication in web environments. In *4th Int. Conf. on Digital Home* (2012), pp. 138–145. 15, 16, 44, 79, 86, 89, 147

[147] UMPHRESS, D., AND WILLIAMS, G. Identity verification through keyboard characteristics. *International Journal of Man-Machine Studies 23*, 3 (1985), 263 – 273. 14, 172

[148] VERVERIDIS, D., AND KOTROPOULOS, C. Information loss of the mahalanobis distance in high dimensions: Application to feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 12 (2009), 2275–2281. xiii, 78, 82, 83, 92, 123, 175

[149] WAYMAN, J. L., JAIN, A., MALTONI, D., AND MAIO, D., Eds. *Biometric Systems: Technology, Design and Performance Evaluation*. Springer, 2005. 30

[150] WIKIPEDIA. Adapted from: `http://en.wikipedia.org/wiki/Keyboard_layout`. Last visited: 9/12/2012. 163

[151] YAMPOLSKIY, R. V., AND GOVINDARAJU, V. Behavioural biometrics: a survey and classification. *International Journal of Biometrics 1*, 1 (2008), 81–113. 11, 147, 171

[152] ZADEH, L. A. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems*. World Scientific, 1996. 68

[153] ZHANG, H., PATEL, V. M., AND CHELLAPPA, R. Robust multimodal recognition via multitask multivariate low-rank representations. In *IEEE Int. Conf. on Automatic Face and Gesture Recognition (FG'15)* (2015), pp. 1–8. 17

[154] ZHAO, X., FENG, T., AND SHI, W. Continuous mobile authentication using a novel graphic touch gesture feature. In *IEEE 6th Int. Conf. on Biometrics: Theory, Applications and Systems (BTAS'13)* (2013), pp. 1–6. 16, 17

[155] ZHENG, N., PALOSKI, A., AND WANG, H. An efficient user verification system via mouse movements. In *18th ACM Conf. on Computer and Communications Security (CCS'11)* (2011), ACM, pp. 139–150. 15, 16, 29, 44