EFFECTIVE MANAGEMENT OF LEARNER INTEGRITY IN ONLINE ASSESSMENTS: AN APPROACH USING KEYSTROKE DYNAMICS

H. A. A. K. M. Seneviratne

Registration No. 158054D

Degree of Doctor of Philosophy

Department of Computer Science & Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

June 2026

EFFECTIVE MANAGEMENT OF LEARNER INTEGRITY IN ONLINE ASSESSMENTS: AN APPROACH USING KEYSTROKE DYNAMICS

H. A. A. K. M. Seneviratne

Registration No. 158054D

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Department of Computer Science & Engineering

Faculty of Engineering

University of Moratuwa

Sri Lanka

June 2026

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date: 2026.06.10

The above candidate has carried out research for the PhD thesis under our supervision. We confirm that the declaration made above by the student is true and correct.

Name of Supervisors: Eng. Prof. Indika Perera

Eng. Prof. Shantha Fernando

Signatures of the Supervisors: Date: 2026-06-10

Date: 2026-06-10

DEDICATION

I dedicate this work to everyone in all parts of the world, who have contributed and are contributing, to promote, uphold, and maintain merit systems within academia.

ACKNOWLEDGEMENT

I would begin by expressing my deep gratitude to my supervisors Eng. Prof. Indika Perera and Eng. Prof. Shantha Fernando. Eng. Prof. Perera who served as my communicating supervisor, extended continued guidance and mentoring to me during my doctoral research. It marked a period of lasting impressions filled with inspiration, understanding, and practical wisdom, all of which turned out to be vital ingredients in developing a research mindset. Eng. Prof. Fernando provided valuable guidance and encouragement at the stage of formulating my research proposal and in refining the publications that were made connected with my research, as well as the thesis. I thank both of them wholeheartedly.

The valued critiques received from my distinguished panel of examiners comprising Eng. Prof. Dileeka Dias, Eng. Prof. Sanath Jayasena, and Eng. Prof. Indika Perera, at all the progress reviews which were extensive and thorough, helped me immensely in achieving each milestone leading to the completion of my research.

The research assistance received from Eng. Imesha Sudasingha in the initial development of a keystroke logger for exclusive use in my experiments is gratefully acknowledged. The insights provided by Prof. Nandana Rajatheva in making this logger lean and the assistance provided by Dr. Thushan Sivalingam in developing and testing a prototype is also gratefully remembered.

I wish to express my sincere appreciation of the pivotal role played by Eng. Devinda Abayagunawardana and Mr. Brian Dunsford, in engaging the services of volunteers to roleplay as subjects in my experiments, which enabled the collection of lifeline data for the research. Their role and commitment in keeping the volunteers adequately briefed while maintaining their anonymity is gratefully admired. On the same note, I extend my thanks to all those volunteers whose identities are not known to me, both individually and collectively, for devoting their time and effort to roleplay in my experiments.

Finally, I would like to thank my wife Manjula, my son Dinindu, my daughter Chinthani, and my daughter in law Panchalie who were towers of strength, for their optimistic companionship and understanding.

ABSTRACT

Educational institutions worldwide face a serious issue when conducting online assessments due to certain candidates cheating. Review of related work in the domains of learner integrity and technology assisted fraud prevention indicated gaps in the application of technology for maintaining learner integrity during online assessments. This thesis addressed these gaps by conceptualizing a system for the effective management of learner integrity during online assessments. The use of the behavioral biometric called 'keystroke dynamics' was thoroughly reviewed and how it could be adapted to authenticate candidates continuously and non-obtrusively while they take online assessments with no accessories deployed at the remote site was conceptualized. The thesis describes how the proof of this concept was established through experiments. This includes the setting up of a fictitious online testing institution and deploying a group of volunteers to roleplay as candidates. Tests were conducted online to simulate a structured initial registration to acquire biometrics for profile creation. Novel features extracted from the acquired biometrics were used to train anomaly detection models employing a variety of binary classifiers. This was followed by online tests that simulated genuine test taking as well as fraudulent test taking. These scenarios were played back offline while continuous authentication was exercised using the trained classifiers. Both conventional metrics and metrics applicable to continuous authentication that have been established recently were used for the performance assessment of the classifiers. Quantitative analysis of the test results indicate that the system is capable of detecting impersonation with high accuracy while other fraudulent acts are also detected with dependable accuracy. The system is also capable of the absolute and profile independent detection of clipboard transfer of text during online assessments. The thesis also discusses schemes for improving the overall detection capability of the system and suggests several directions to inspire future work.

Keywords: Learner Integrity, Continuous Authentication, Keystroke Dynamics.

TABLE OF CONTENTS

Declaration i
Dedication ii
Acknowledgement iii
Abstract iv
Table of Contents v
List of Figures xii
List of Tables xvi
List of Abbreviations xvii
List of Appendices xviii
CHAPTER 1 1
INTRODUCTION 1
1.1 Problem with online assessments 1
1.2 Attempts to prevent cheating at online assessments 2
1.3 Research Problem and the Research Questions 6
1.4 Research objectives 7
1.5 Outline of research work undertaken 7
1.6 Expected scientific and social outcome of the research 8
1.7 Organization of the thesis 8
CHAPTER 2 11
CRITICAL REVIEW OF RELATED WORK 11
2.1 Chapter description 11
2.2 Learner integrity 11
2.2.1 Brief history of academic dishonesty 12
2.2.2 Emergence of cheating within the sphere of online learning 13
2.2.3 Attempts to curb technology aided academic dishonesty 16
2.3 Overview of biometrics and biometric authentication 18
2.3.1 Physiological biometrics 19
2.3.2 Behavioural biometrics 20
2.3.2.1 Keystroke dynamics 21

2.3.2.2 Rhythm-Click Characteristics 24
2.4 Applications of Keystroke Dynamics 25
2.4.1 Principles and Techniques 25
2.4.2 Strengthening conventional authentication 27
2.4.3 Use as a Soft Biometric 28
2.4.4 Use of non-temporal features 29
2.4.5 Other applications of keystroke dynamics 30
2.5 Continuous Authentication (CA) 30
2.6 Performance assessment of biometric systems 32
2.7 Benchmark Datasets 33
2.8 Applicability to online authentication 34
2.9 Performance assessment of continuous authentication systems 34
2.10 Device Dependence 35
2.11 Challenges in applying authentication to Learner Integrity Management 36
2.12 Challenges posed by time variance and strategies for coping 37
2.13 Challenges posed by the availability of Large Language Models 39
2.14 Ethical considerations and legal implications 39
2.15 Conclusions drawn from the review of related work and their relevance to management of learner integrity 40
2.16 Recapitulation 44
CHAPTER 3 45
THE RESEARCH METHODOLOGY 45
3.1 Chapter description 45
3.2 Research methodology adopted 45
3.2.1 Approach to the research 45
3.2.2 Design of the research 45
3.2.3 Methods used for data collection 45
3.2.4 Strategy for collecting subjects 45
3.2.5 Methods used for the analysis of data gathered 46
3.2.6 Ethical and administrative considerations 46
3.2.7 Scope of the research and its limitations 46
3.3 Distinctive challenges associated with online assessments 47

3.4 Suitability of Keystroke Dynamics as the biometric 47
3.5 Use of Lockdown Browsers 48
3.6 The authentication system architecture and workflow 48
3.7 Description of the features to be extracted 53
3.7.1 The 'Dwell Time Vector' 54
3.7.2 The 'Hot Sequence Vector' 54
3.7.3 The 'Quick Sequence Vector' 55
3.7.4 The 'Far Sequence Vector' 55
3.7.5 The 'Pause Duration Ensemble' 56
3.7.6 Extraction of non-temporal features 59
3.8 Creation of the Candidate's Profile 59
3.9 Continuous Authentication during subsequent tests 60
3.10 Method of anomaly detection 60
3.11 Training the classifiers 66
3.12 Conformity to the specifications 68
3.13 Legal implications and ethical considerations 69
3.14 Recapitulation 70
CHAPTER 4 71
THE EXPERIMENTS 71
4.1 Chapter description 71
4.2 The experimental protocol 71
4.3 The data acquisition phase 72
4.3.1 Description of the tests 72
4.3.2 Selection of volunteers for roleplay 77
4.4 The profile building phase 79
4.4.1 Extraction of the 'Dwell Time Vector' 79
4.4.2 Extraction of the 'Flight Time Related Features' 80
4.4.3 Extraction of non-temporal features 84
4.5 The detection model development phase 85
4.5.1 Model development for dwell time related features 86
4.5.1.1 Generation of training data for 'genuine test taking' 86
4.5.1.2 Generation of training data for 'fraudulent test taking' 87

4.5.1.3 Building the composite dataset 88
4.5.1.4 Selection of roleplayers for simulation of 'fraudulent test taking' 88
4.5.1.5 Choosing a classifier model 89
4.5.2 Model development for the remaining features 89
4.5.3 Selecting the best classifier 90
4.5.4 Consistency with the concept 91
4.6 The analysis phase 92
4.6.1 Repeatability of experiments 92
4.6.2 Simulation of impersonation 93
4.6.3 Simulation of looking up and real time collaboration 95
4.6.4 Use of non-temporal features 96
4.7 Supplementary experiments 96
4.7.1 Protocol analysis of the client server communication during the acquisition of keystroke data 96
4.7.1.1 Examining the nature of the data transfer 97
4.7.1.2 Examining the effect of the volume of text typed 97
4.7.1.3 Testing of a lean keystroke logging system 98
4.7.2 Examination of clipboard transfers during online test taking 101
4.8 Choice of techniques and methods 103
4.9 Resources used in the experiments 103
4.10 Recapitulation 105
CHAPTER 5 107
INTERPRETATION OF RESULTS 107
5.1 Chapter description 107
5.2 Interpretation of the validity of the profiles created 107
5.2.1 Validity of the experimental subjects 107
5.2.2 Validity of the mock test 107
5.2.3 Validity of the features extracted 108
5.2.4 Validity of the training datasets 109
5.2.5 Validity of selection of the best classifiers 111
5.2.6 Overall validity of profiles created 114
5.3 Interpretation of results of the simulation of genuine test taking 114

5.3.1 Accuracy of genuine detections 114
5.3.2 False Rejection Worst Interval (FRWI) for genuine test taking 115
5.3.3 Validity of non-temporal features 115
5.3.4 Overall interpretation of results of the simulation of genuine test taking 116
5.4 Interpretation of results of the simulation of impersonation 116
5.4.1 Accuracy of imposter detection..... 120
5.4.2 False Acceptance Worst Interval (FAWI) during impersonation 121
5.4.3 Time to First Imposter Detection (T2FD) during impersonation..... 122
5.4.4 Unusability of non-temporal metrics in imposter detection..... 123
5.4.5 Overall interpretation of results of the simulation of impersonated test taking..... 123
5.5 Interpretation of results of the simulation of lookup..... 124
5.5.1 Accuracy of detecting lookup 124
5.5.2 Time related metrics during lookup 126
5.5.3 Overall interpretation of results of simulation of test taking while looking up..... 127
5.6 Interpretation of results of the simulation of collaboration..... 127
5.6.1 Accuracy of detecting collaboration 127
5.6.2 Time related metrics during collaboration 129
5.6.3 Overall interpretation of results of simulation of test taking while collaborating..... 130
5.7 Comparison of performance when detecting different fraudulent acts 131
5.8 Interpretation of results of the Protocol Analysis experiments 131
5.9 Interpretation of Results of the Clipboard Transfer Experiment..... 137
5.10 Key findings 140
5.11 Recapitulation..... 141
CHAPTER 6 143
DISCUSSION 143
6.1 Chapter description 143
6.2 Performance benchmark comparison 143
6.3 Establishment of Proof of the Concept 145
6.4 Revisiting the research questions 146

6.5 The drawbacks in the experiments 150
6.5.1 Low accuracy in detecting fraudulent acts other than impersonation ... 150
6.5.2 Non-usability of the biometrics of certain subjects 151
6.5.3 Inadequate replication of human factors during tests in the experiments 153
6.6 Suggestions for improving the authentication system 154
6.6.1 Alternative classifiers for detection of fraudulent acts other than impersonation 154
6.6.2 Use of machine learning and Artificial Intelligence 154
6.6.3 Improvements to the profile creation process 155
6.6.4 Improvements to the biometrics acquisition process 155
6.6.5 Improvements to the client – server communication during online tests and the processing of keystroke logs 156
6.7 Identification of points of failure 157
6.8 Key takeaways from the discussion 158
6.9 Recapitulation 158
CHAPTER 7 159
CONCLUSION 159
7.1 Chapter Description 159
7.2 Overall conclusion made through the research 159
7.3 Contributions of the research 160
7.4 Directions for future work 160
7.4.1 Additional features of non-obtrusive nature 160
7.4.2 Improved classifiers 162
7.4.3 Profile updating 163
7.4.4 Mitigation strategies 163
7.4.5 Population analysis 164
7.4.6 Device dependence 164
7.4.7 Adaptation of the system to other languages 165
7.5 Recapitulation 165
REFERENCES 167
BIBLIOGRAPHY 177
APPENDIX - A 179

ALGORITHMS USED IN THE EXPERIMENTS 179
Algorithm – 1: Extraction of dwell time vector from biometrics: 179
Algorithm – 2: Extraction of pause duration ensemble from biometrics: 182
Algorithm – 3: Extraction of flight time vectors (generic) from biometrics: .. 184
Algorithm – 4: Generation of training data for genuine test taking: 190
Algorithm – 5: Generation of training data for fraudulent test taking: 191
Algorithm – 6: Simulation of genuine and impersonated test taking: 192
Algorithm – 7: Simulation of fraudulent test taking: 193
Algorithm – 8: Selection of anomalous data sources for a roleplayer: 194
Algorithm – 9: Generation of training dataset for a roleplayer for a given feature: 195
Algorithm – 10: Generic algorithm to train and test a classifier: 197
Algorithm – 11: Selection of best classifier for a vector based feature: 198
Algorithm – 12: Selection of best classifier for an ensemble based feature: ... 200
APPENDIX - B 203
INDEX OF SOURCE LISTINGS 203
APPENDIX - C 221
RELATED PUBLICATIONS 221

LIST OF FIGURES

Figure #	Description	Page
2.1	Growth trend of research on 'keystroke dynamics'	23
2.2	Definitions of timing measurements on keystroke events	27
2.3	Variation of assessment metrics with sharpness of authentication system	33
2.4	Application of the 'Fraud Diamond' to the scenario of online assessments.	42
2.5	'Learner Integrity Managed' Online Testing System	43
3.1	Workflow of the proposed authentication system	49
3.2	The architecture of the proposed authentication system	50
3.3	First stage of offline profile creation – Reference vector extraction	51
3.4	Second stage of offline profile creation – Training data generation	52
3.5	Final stage of offline profile creation – Classifier training	53
3.6	Measurement of epochs for key events	53
3.7	Allocation of keys to each finger in each hand, for touch typing.	56
3.8	Algorithm to extract 'Dwell Time Vector'	57
3.9	Distinct clustering of pause durations of two persons	58
3.10	Algorithm to extract 'Pause Duration Ensemble'	58
3.11	3D/2D analogy for the formation of the vector pair μ_s and $\widehat{\mu_s}$ and the angle of departure θ.	61
3.12	Generation of a stream of θ values while the candidate types	62
3.13	Anomaly detection using the angle of departure	63
3.14	Different scenarios of occurrence for the 'Angle of Departure'	64
3.15	Anomaly detection using the mean value of pause durations	66
3.16	Algorithm for creating training data for binary classifier for dwell time related features	67
4.1	The phases of the experiments	71
4.2	A sample screen capture showing the response panel for a typical question in the online test interface	72
4.3	Contents of an example file storing the key event data	74

Figure #	Description	Page
4.4	Simulation of lookup during an online test	76
4.5	Simulation of real time collaboration during an online test	77
4.6	Relative frequencies of the highest occurring 20 sequences of two letters in English text	80
4.7	Structure of the 3-D array used to store hot sequences	82
4.8	Structure of 2-D array for storing the parameters of 'hot sequences'	82
4.9	Structure of the data frame for storing 'Quick Sequences'	83
4.10	Algorithm to generate datapoints for genuine test taking	86
4.11	Algorithm to generate datapoints for fraudulent test taking	87
4.12	Algorithm for simulation of continuous authentication amidst impersonation	93
4.13	Algorithm for simulation of continuous authentication amidst looking up	95
4.14	The test character sequence typed into the answer box of question 1 of Test 01, just before submission	97
4.15	Protocol Analyzer screen with filter 'HTTP' selected, showing the data exchange while the test sequence was typed	98
4.16	5-byte data structure used to encode a single key event	100
4.17	Algorithm for lean coding of a single key event	101
5.1	Frequency distribution of the size of keystroke logfile across the population of 38 roleplayers	108
5.2	Frequency distribution of the size of 'Pause Duration Ensemble' file across the population of 38 roleplayers	109
5.3	Scatter plots of training datasets based on 'Angle of Departure' between Dwell Time Vectors	110
5.4	Scatter plots of training datasets based on 'Angle of Departure' between Hot Sequence Flight Time Vectors	110
5.5	Scatter plots of training datasets based on 'Pause Duration Ensembles'	111
5.6	Frequency distribution of FRR (%) across the population of roleplayers	114
5.7	Frequency distribution of FRWI across the population of roleplayers	115
5.8	Frequency distribution of FAR (best performance) across the population of roleplayers	120

Figure #	Description	Page
5.9	Frequency distribution of average FAR across the population of roleplayers	121
5.10	Frequency distribution of average worst case False Acceptance Interval (in minutes) when roleplayers impersonate each other	122
5.11	Frequency distribution of the T2FD (minutes) when roleplayers impersonate each other	123
5.12	Frequency Distribution of FAR across 34 roleplayers when they take online test while looking up	125
5.13	Frequency Distribution of FAWI while looking up, across 20 roleplayers whose test taking exhibit an anomaly detection accuracy of more than 40%	126
5.14	Frequency Distribution of T2FD while looking up, across 20 roleplayers whose test taking exhibit an anomaly detection accuracy of more than 40%	127
5.15	Frequency distribution of FAR across 26 roleplayers taking test while collaborating	129
5.16	Frequency Distribution of T2FD across 24 roleplayers whose test taking exhibits a non-zero anomaly detection accuracy	129
5.17	Frequency Distribution of FAWI across 24 roleplayers whose test taking exhibits a non-zero anomaly detection accuracy	130
5.18	Contents of the JSON file holding the keystroke events transmitted from the browser client.	132
5.19	Protocol analyzer screen showing the loopback traffic while the test sequence was typed	133
5.20	Anatomy of the first HTTP datagram that carried the keystroke data of the test sequence to the server	134
5.21	The contents of the byte file resulting from the modified keystroke logging	135
5.22	Data on the wire when the test sequence ‘1234ABCD’ was typed	136
5.23	Variation of HTTP payload size with the number of keys typed for both types of keystroke logging	136
5.24	Response panel of the modified Test 01 to examine text transfer via the clipboard	138
5.25	The responses captured separately on the JSON file and the answer file.	139
6.1	Algorithm used by the back end script for logging keystroke data	152

Figure #	Description	Page
6.2	Algorithm for converting ‘byte file’ to a JSON log	156

LIST OF TABLES

Table #	Description	Page
1.1	Countermeasures for fraudulent practices during online assessments	03
1.2	Effectiveness of technical countermeasures	04
3.1	Comparison of proposed system with other commonly used biometric authentication systems	69
4.1	Completion of roleplays by the volunteers	79
4.2	6-bit code to encode the distinct keys on a QWERTY keyboard	99
5.1	Test completion by the roleplayers and usability of biometrics	107
5.2	List of classifiers for a single feature	112
5.3	Number of roleplayers for whom each classifier has been selected as the best	113
5.4	Comparison of non-temporal features extracted from Tests 01 and 02	116
5.5	Best performance metrics of simulated impersonations by 37 imposters	117
5.6	Average values of performance metrics of simulated impersonations by 37 imposters	118
5.7	The variance of average test session parameters across the population of 38 roleplayers, when impersonated by 37 roleplayers	119
5.8	Performance metrics of test taking by 34 roleplayers while looking up	124
5.9	Performance metrics of test taking by 26 roleplayers while collaborating	128
5.10	Comparison of detection performance for different fraudulent acts	131
5.11	Observations made upon different scenarios of text input	137
6.1	Performance comparison with a benchmark system	145
6.2	Evaluation of Authentication Methods deployable during online tests	147

LIST OF ABBREVIATIONS

Abbreviation	Expansion / Description
2FA	Two-Factor Authentication
AI	Artificial Intelligence
ANGA	Average Number of Genuine Actions
ANIA	Average Number of Imposter Actions
ASCII	American Standard Code for Information Interchange
.CSV	Comma-Separated Values (extension for files containing tabular data)
CA	Continuous Authentication
CBAS	Continuous Biometric Authentication System
CBT	Computer Based Testing
DB	Benchmark Dataset (*abbreviated in French*)
DT	Dwell Time
ECMA	European Computer Manufacturers Association
ECG	Electrocardiogram
EER	Equal Error Rate
ENISA	European Network and Information Security Agency
FAR	False Acceptance Rate
FAWI	False Acceptance Worst Interval
FRR	False Rejection Rate
FRWI	False Rejection Worst Interval
FT	Flight Time
FTP	File Transfer Protocol
GenAI	Generative AI
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LLM	Large Language Model
LMS	Learning Management System
LR	Logistic Regression
LTS	Long Term Support
MOOC	Massive Open Online Course
NB	Naïve Bayes
NLP	Natural Language Processing
PHP	PHP : Hypertext Preprocessor (*recursive acronym*)
RF	Random Forest
SFTP	Secure FTP
SVM	Support Vector Machine
T2FD	Time to First Detection
TCR	Time to Correct Reject
URL	Uniform Resource Locator
UTC	Universal Time Coordinated (or Coordinated Universal Time)
WPM	Words per Minute
XAI	Explainable AI

LIST OF APPENDICES

Appendix #	Description	Page
Appendix - A	Algorithms Used in the Experiments	179
Appendix - B	Index of Source Listings	203
Appendix - C	Related Publications	221

CHAPTER 1

INTRODUCTION

1.1 Problem with online assessments

Knowledge society has created an unprecedented need for learning. Due to the gross inadequacy in the capacity of conventional institutions to meet this need, the demand for distance learning by off campus students keeps growing exponentially. A study by Akintolu and Adewoye [1] based on a study conducted in 2021 focusing on the African education system confirms this exponential growth. To cope with this trend, institutions that offer distance learning have increasingly adopted eLearning technologies. Adewole and Fatoye [2] state that eLearning is changing the way education is perceived and implemented. Qayyum and Zawacki-Richter [3] state that advances in information and communication technology along with the widespread use of the Internet as a common access medium and cheaper possession of smart mobile devices have reinforced this trend.

However, evaluating student performance, which is an integral part of any academic programme, has become a major issue when the delivery mode is eLearning. Conducting assessments and evaluating the performance of students have always been challenges when it comes to distance learning. In an article titled 'Assessing Large Cohorts' published by the BILT [4], the challenge of conducting assessments for large groups is discussed. The off-campus student population of many institutions is generally larger than the regular student population by several orders of magnitude. This is indicated in the work of Adewole and Fatoye [2] cited previously. The simultaneous evaluation of a large number of candidates through a common test had created many logistic problems. The advent of eLearning has only aggravated the problem to a point where the candidate numbers are so large that simultaneous common tests of the conventional paper and pencil type are no longer practicable. Referring to online courses and Massive Open Online Courses (MOOCs), P. van Vliet [5] discusses the issue faced by instructors in scaling up assessment while maintaining relevant quality standards.

Consequently, online assessments were developed. One would expect online assessments to be the counterpart of conventional assessments in the same manner that eLearning was the counterpart of conventional classroom programme delivery. However, this did not happen in practice. The reason was the tendency by some fraction of the candidate population to cheat while undergoing online assessments. This poses a major challenge to the conducting institutions. The reasons for this tendency are parts of a social problem. Need for accelerated career development in a competitive world, economic pressures, and prestige are some of these. Holden et al. [6], in a research review on the academic integrity of online assessment, discuss individual, institutional, medium-related, and assessment-specific factors that cause

candidates to cheat. It may be deduced that the mere existence of room for the student populace to resort to fraudulent practices tends to undermine the overall effectiveness of online assessments irrespective of the size of the fraction of candidates who actually resort to such practices.

As discussed in the work of J. R. Young [7], the practice of cheating during online assessments has prevailed since the advent of online assessments. The academic community has been aware of the problem. Despite the best efforts by test administrations to counter cheating at online assessments, it remains a major challenge for them. This is primarily due to their inability to keep effective tab on candidates while a test is being administered remotely.

1.2 Attempts to prevent cheating at online assessments

The earliest countermeasures that were used to curb cheating at online assessments were mostly non-technical. As pointed out in the work of J. Adkins et al. [8] and D. Vincent [9], these were nothing more than mere adaptations of controls typically employed in conventional classroom tests to prevent cheating. But the online scenario permits additional room for cheating, making the effectiveness of these countermeasures inevitably limited.

As elaborated in the work of O. L. Holden et al. [6], the most common fraudulent practices in online assessment that are widespread are listed below:

a. **Impersonation**, where the candidate gets a competent colleague to pose as him/her and undergo the test on his/her behalf.

b. **Collaboration**, where the candidate undergoes the test with real-time assistance of competent colleagues.

c. **Tapping**, where the candidate unduly acquires prior knowledge about an assessment mainly from parts of the test captured by colleagues who have undergone the same assessment earlier.

d. **Look up**, where the candidate unduly obtains cues while undergoing the assessment by browsing physical material like books or material either stored on or accessed online using the device used for the assessment itself, or using a separate device.

Out of the above, tapping could be completely eliminated through the purely administrative measure of making the tests concurrent for all candidates. Hence, the need does not arise to counter it particularly. The other 3 remain to be countered.

As identified in the work of T. Michael and M. Williams [10] and as described by N. Chadaga [11], some possible countermeasures for the remaining 3 fraudulent practices are summarized in Table 1.1:

Table 1.1: Countermeasures for fraudulent practices during online assessments

Fraudulent practice	Non-technological countermeasures	Technological countermeasures
Impersonation	Remote Proctoring	360° scan of remote examination site Web cam monitoring Physiological biometrics Behavioural biometrics
Collaboration	Remote Proctoring	360° scan of remote site Webcam monitoring Behavioural biometrics Use of lockdown browsers
Look up	Frequent alteration of tests Checking for plagiarism	360° scan of remote site Webcam monitoring/gaze detection Use of lockdown browsers System call analysis

User authentication is an important element of any restricted online system, and today many robust authentication systems are available to control access so that only genuine registered users can gain access to online systems. However conventional authentication systems have an important limitation within the context of online assessments. They rely heavily on the user's cooperation to maintain his/her credentials confidential. In other restricted online applications, the user has no incentive to disclose credentials to a third party, and he/she would not do so in his/her own interest. In contrast, in online assessments, this situation is inverted. As disclosed in the work of K. Mugambi et. al. [12], a candidate with the will to cheat would willingly share his credentials with a collaborator. The test administration has no way of controlling this. Consequently, the task of maintaining learner integrity in online assessments is much more complex and challenging.

Authentication at login alone would be grossly insufficient because a candidate could login to the system initially and then hand over the session to the imposter. Hence, a system that could continuously authenticate the user throughout the session is required. The guidelines authored by D. Temoshok et al. for the National Institute of Standards and Technology (NIST) of the US Department of Commerce [13] stipulate that authentication during online systems should be continuous. They refer to this process as 'session monitoring'.

Considering this peculiarity of the online test scenario coupled with the requirement that the authentication process should not get in the way of the test itself and that a candidate with the will to cheat is unlikely to cooperate with the institution and instead

would attempt to evade detection, it may be concluded that a system used for the authentication of candidates during online tests should at the minimum conform to the following specifications:

i. It should not require or depend on the cooperation of the user and preferably should work in the background, invisible to the user.

ii. It should not require additional hardware or accessories to be deployed at the user site and should not exert demands on resources such as connection bandwidth.

iii. It should provide a means for the continuous authentication of the user.

These specifications render some of the technological countermeasures listed in Table 1.1 ineffective, as indicated in Table 1.2 below:

Table 1.2: Effectiveness of technological countermeasures

Technological Countermeasure	Evaluation
360° scan of remote site	Requires user cooperation, large connection bandwidth, and deployment of accessories. Also, not conducive for continuous authentication.
Webcam monitoring/gaze detection	Requires user cooperation. Demand for bandwidth may get in the way of the session.
Physiological biometrics	Requires user cooperation, large connection bandwidth, and deployment of accessories. Requires the holding of large amounts of template data at the server end. Also, not conducive for continuous authentication.
System call analysis	Requires processes to be run on the user's device which may provide cues about the authentication process to the user. Also, it may slowdown the session.

Accordingly, the only two remaining countermeasures that are effective, namely, the use of "Lockdown Browsers" and "Behavioral Biometrics" emerge as ingredients for developing an effective system to curb cheating during online assessments.

Lockdown browsers have the ability to restrict all fraudulent practices except impersonation. Such browsers as described by N. Chadaga [5] are commercially available. They are usually paired with the Learning Management System (LMS) of the test administration. They permit only the minimum number of connections to the

test administrator's server that is required for the online test, prevent the running of any other application on the candidate's device, disable all ports on the candidate's device except those required for the session, and disconnect all communications other than the connections supporting the session. These inhibitions curtail the candidate's ability to collaborate or lookup using the same device. Accordingly, the deployment of a lockdown browser in itself would be a considerable deterrent for cheating during online assessments. However, a lockdown browser does not prevent physical collaboration or looking up from books or from another device. Lockdown browsers also cannot prevent impersonation.

Impersonation can, however, be effectively deterred using Behavioral Biometrics. As opposed to Physiological Biometrics such as fingerprint scanning and iris scanning, the acquisition of which are inherently obtrusive, Behavioral Biometrics capture certain behaviors of an individual that are found to be inherent and unique to the individual. If the candidate attempts to collaborate or lookup during the session, the candidate's natural behavior would alter to an extent that provides a cue that the candidate may not be responding to the assessment genuinely or individually.

The most common behavioral biometrics used for online authentication are 'Keystroke Dynamics' and 'Rhythm Click Characteristics'. As defined by Chourasia and Joeg [14] keystroke dynamics refer to the typing patterns and mannerisms used by an individual when using computer keyboards, which have been found to be unique to an individual. Rhythm click characteristics refer to similar traits in the use of pointing devices. A. Ahmed and I. Traoré [15] define rhythm click characteristics as the characteristics of the actions received from an input pointing device for a specific user. Since online assessments invariably involve the use of keyboards and pointing devices for the candidate to interact with the session, these biometrics provide a convenient, inexpensive, and transparent means of user authentication while not requiring the user's cooperation or the deployment of application-specific accessories. In addition, their acquisition is non-obtrusive.

The work of C. A. Fidas et al. [16] is a longitudinal study of a remote proctoring system for ensuring trust in online learning environments. This study examines both oral and written online tests, while the written tests include handwritten as well as digitally written tests. Both 'keystroke dynamics' and 'rhythm click characteristics' are used as authentication modalities. However, a video conferencing system is used for direct remote surveillance of the test site. Therefore, this system is also obtrusive, suffers from the constraints of requiring user cooperation, requires the deployment of accessories, and has heavy demand on system resources as mentioned in Table 1.2. Nevertheless, it provides the motivation to examine the effectiveness of keystroke dynamics and rhythm click characteristics as standalone authentication modalities, which would thereby conform to the aforesaid specifications.

As will be seen in the next chapter, 'Keystroke Dynamics' has been a ripe research area as far back as the late 90s. However, its primary focus has been on access control

in conventional online applications that mostly use “static authentication” where the keystroke dynamics exhibited when typing only a predetermined sequence of characters are analyzed. Rhythm Click Characteristics is relatively new and has been used to an even lesser extent. Prevention of cheating during online assessments would require “dynamic” as well as “continuous” authentication. Hence, this research was inspired by the room to expand the present boundaries of the knowledge on keystroke dynamics to particularly address the needs of a continuous authentication system that could effectively manage learner integrity during online assessments.

1.3 Research Problem and the Research Questions

The research problem that this study strived to address was, that:

> ***“There is no existing, institution-ready, system for the effective management of learner integrity during online assessments through the non-obtrusive and continuous authentication of candidates.”***

The ‘institution-readiness’ as stated in the research problem refers to the ability of the institution to deploy an authentication system that does not exert irrational demands on existing resources used for online tests, unilaterally, with no cooperation required of the user.

Further, the establishment of such a system should be an ‘Engineering Solution’ to the research problem where appropriate technologies are deployed to provide the solution rather than merely extending conventional administrative integrity management measures to the domain of online tests.

Also, while appreciating the importance and impact of non-technological measures towards maintaining learner integrity, such as value inculcation and promotion of honour codes among learners, such measures are not included as part of the proposed system.

Accordingly, this research should answer the following questions:

RQ1 **What non-obtrusive, institution-ready methods capable of providing continuous authentication are available for managing learner integrity during online assessments?**

RQ2 **Does keystroke dynamics based continuous authentication when used in conjunction with lockdown browsers form the basis for the establishment of a system for the effective management of learner integrity during online assessments?**

RQ3 **How should such a system be designed?**

RQ4 How should the effectiveness of the designed system be verified?

When attempting to answer these research questions, it was assumed that lockdown browser technology is mature and that such browsers are at the disposal of the test administration. It was already seen that Keystroke Dynamics has the potential to detect the fraudulent practices that a lockdown browser cannot prevent. The two complement each other. Consequently, it was assumed that if a dynamic and continuous authentication system based on keystroke dynamics could work effectively in conjunction with an 'industry standard conventional browser', its effective working in conjunction with a lockdown browser would be beyond question. This permitted the research activities to be carried out using industry-standard browsers.

1.4 Research objectives

The objectives this research strived to accomplish were as follows:

- To **identify** suitable non-obtrusive, institution-ready, modalities capable of providing continuous authentication for the management of learner integrity during online tests.
- To **verify** whether keystroke dynamics based continuous authentication when used in conjunction with lockdown browsers form the basis for the establishment of a system for the effective management of learner integrity during online assessments.
- To **develop** a concept for the design of such a system.
- To **determine** how the effectiveness of a system designed based on the concept could be verified.
- To **establish** proof of the developed concept by verifying the effectiveness of an experimental system based on the concept.

1.5 Outline of research work undertaken

The following research activities were undertaken in pursuit of solutions for the research problem stated above:

i. Critical review of work carried out by other researchers in the knowledge domains of learner integrity, continuous authentication, and keystroke dynamics.
ii. Identifying research gaps in the aforesaid knowledge domains.
iii. Developing suitable concepts to address the identified gaps.
iv. Designing suitable experiments to establish proof of the concept.
v. Collecting suitable subjects for the experiments.
vi. Conducting the experiments.
vii. Analysis of results.

1.6 Expected scientific and social outcome of the research

1.6.1 Scientific outcome

At the time of invoking this research, the primary scientific outcome envisaged was the determination of whether the monitoring of keystroke dynamics is an effective means of continuously detecting impersonation, collaboration, looking up, or any combination of these fraudulent practices during online assessment sessions. If the findings establish that it is effective, then there would be accompanying findings as to how the keystroke dynamics should be analyzed to derive the required detections. If the findings establish to the contrary, that is, if monitoring of keystroke dynamics is not an effective means of continuous user authentication during online tests, then there would be accompanying findings as to its limitations. Either way the frontiers of the present knowledge domain of behavioral biometrics would be expanded.

1.6.2 Social outcome

It was also envisaged that if the research succeeds in establishing keystroke dynamics based continuous authentication as an effective means for continuous detection of fraudulent practices during online assessments, the knowledge thus created and the concepts proved would eventually find their way into the applications used in online assessment, which would deter any fraudulent practices on the part of the candidates. This in turn would enable candidates who take the tests genuinely to score well, based purely on merit. Therefore, the integrity of online assessments would be preserved and enhanced, enabling test administrators to deploy online testing as an integral part of their assessment procedures. Their evaluation systems could be automated, would be reliable, and would gain public confidence. The important social outcome would be the availability of a transparent and truly merit-based assessment system within academia that would enhance the confidence of the wider student population.

1.7 Organization of the thesis

For the purpose of effective presentation, the thesis from here onwards is organized in the following manner:

Chapter 2 contains the account of the attempt made to grasp the existing knowledge base on which the research problem is centered. This is achieved through the review of sources comprising previous studies made on the subjects of learner integrity, keystroke dynamics, and related topics; subject matter expert opinions; and web resources.

Chapter 3 presents the research methodology employed in this research for the establishment of a system for managing learner integrity during online tests, including the proposal of a concept.

Chapter 4 is devoted to the detailed description of how the research methodology was applied. It provides details of experimental work carried out to prove the concept presented in Chapter 3, the protocols adhered to, the data collected, the analysis conducted, and the methods used.

Chapter 5 is devoted to the interpretation of the results derived from the experiments described in Chapter 4.

Chapter 6 comprises a discussion on the overall research and the establishment of proof of the concept. A benchmark comparison of performance is also provided. The research questions posed in section 1.3 are answered in this chapter. It also identifies the drawbacks in the system used in the experiments and discusses ways of overcoming them.

Chapter 7 presents the overall conclusion of the thesis, summarizes the contributions made through this research work, and suggests some directions for future work arising from this research.

The above chapters are followed by the list of references and bibliography used in this research.

The 3 appendices that follow provide detailed algorithms used in the experiments; an index of source listings of all software programs used in the experiments; and details of related publications made during the course of the research.

CHAPTER 2

CRITICAL REVIEW OF RELATED WORK

2.1 Chapter description

This chapter is devoted to the review of related work of others, to gain an understanding of the broad subject of 'Learner Integrity' and the challenges encountered in maintaining it. It also examines studies carried out on 'Technology Assisted Academic Dishonesty' with particular focus on its implications on 'Online Assessments' and 'Technological Countermeasures', to grasp their current status and to stablish the direction of the proposed research.

In making these reviews, it is kept in mind that the system proposed to be established should provide an 'engineering solution' to the research problem through the deployment of appropriate technology. Therefore studies dealing with the use of technology for cheating as well as its prevention are treated with interest.

2.2 Learner integrity

An article published by B. Fang in 2012 [6] cites one of the earliest definitions of Learner Integrity. In fact, it defines the opposite or the lack of 'Learner Integrity' namely, 'Academic Dishonesty'. This work cites A. Mullens ("Cheating to win: Some administrators, faculty and students are taking steps to promote a culture of academic honesty"), where Academic Dishonesty has been defined as "…anything that gives a student an unearned advantage over another".

The salient point here is that the advantage is 'unearned' and therefore 'undue'. The word 'anything' included in the definition greatly broadens the spectrum of acts of academic dishonesty, with no reference to the time or place they are committed, or to the nature of the acts such as being technological or non-technological, or to the reasons that motivate such acts. Conversely, it could be inferred that the opposite of academic dishonesty which is 'Academic Integrity' on the part of the students or 'Learner Integrity' as referred to in the current study should have the opposite attributes. While academic dishonesty as defined in this paper is broad, limitless, and not readily predictable, it could be inferred that learner integrity should be specific, confined, and readily observable through clear outcomes, implying that 'Academic Integrity' is something an institution could enforce, foster, and encourage.

The paper emphasizes that punitive approaches towards ensuring learner integrity could alienate students from the institution and recommends positive reinforcement instead.

The paper also makes an important point that curbing academic dishonesty is particularly important in promoting integrity in all spheres of life. Since academic institutions groom citizens to take up responsibilities in the real world comprising the state institutions, the corporate sector, industries, non-governmental sector etc. it is learner integrity that would later develop into overall integrity. It points out that although students may be motivated towards academic dishonesty due to factors confined to academic institutions, lack of measures to enforce and promote learner integrity could result in the cultivation of attitudes and values that encourages dishonesty in real life, even in the absence of such motivating factors.

The paper also highlights the necessity for a clear understanding and agreement between the institution and the students as to what learner integrity is. It also deals with strategies to fight academic dishonesty. It lists 6 strategies where one of them is the 'use of preventive technologies' while the other 5 are all non-technological administrative measures.

2.2.1 Brief history of academic dishonesty

The work of McCabe, Treviño and another [18] is a systematic review of research on academic dishonesty in institutions carried out over a decade.

As uncovered by this study, the earliest known report of large scale cheating by college and university students in the United States had been made in 1964. The acts involved cheating at off-classroom activities such as homework and assignments. By 1993, that is over a span of 3 decades, cheating at examinations had increased significantly surpassing all other acts of cheating. The acts included copying, using crib notes, and collaborating during the exam.

This study identifies several factors that motivates students to engage in academic dishonesty. It classifies these factors as 'contextual' and 'individual'. The contextual factors include inadequate measures adapted by the institution to curb cheating, cheating by peers, institutional size, and inadequate severity of punitive measures on identified culprits. Individual factors include low mid-program GPA, engaging in extra-curricular activities, family pressure, economic pressures, age, and gender.

Prevention of cheating in academic institutions is dealt with in this study as well. The preventive measures that are recommended are all administrative in nature and include the implementation of honour codes, clear communication of institutional expectations on behaviour during examinations, minimizing opportunities for cheating, and measures aimed at inculcating honesty and character. No mention is made on the use of technology for cheating or for its prevention.

The work of A. R. But et al. [19], is yet another systematic review of research on the broad topic of 'Academic Integrity'. It has focused on the period from 1996 to 2023 and has employed bigram analysis based on Natural Language Processing (NLP) to analyze 1000 research articles published during that period. The findings acknowledge the infusion of technology into the teaching process and mentions the advent of

software tools such as ‘Turnitin’ for the detection of plagiarism. This is an example of the early use of technology for ensuring learner integrity. The paper makes no mention of the use of technology on the part of cheating students. The contents of this paper infer that technology assisted enhancement of learner integrity has emerged sometime between 1996 and 2023.

The work of T. Bretag et al. [20] is a systematic review of the academic integrity policies in the Australian higher education system. It reviews the published academic integrity policies of 39 Australian universities. It identifies five elements that an academic integrity policy of an institution should possess for it to be ranked as exemplary namely, access, approach, responsibility, detail, and support. Under the element of detail, it specifies what the institution should publish as breaches of academic integrity and how these breaches are detected. The use of text-matching software is cited as an example. This indicates the recognition of the role of technology in promoting learner integrity by this study. However, the role of technology as perceived here is on detecting off-classroom and asynchronous cheating such as copying from other sources for inclusion in assignment submissions. The inadequacy in the recognition of the role of technology in ensuring learner integrity is evident elsewhere in the publication where ‘enablers’ of academic integrity within an institutions are listed. The list includes ‘teachers’, ‘librarians’ ‘research colleagues’ etc. whereas ‘technologists’ and ‘technical personnel’ are omitted. There is also no evidence of the recognition of the possibility for ‘technology assisted cheating’.

The publication also quotes the following formal definition of ‘academic integrity’ on which its reviews had been based:

> *“Academic integrity is about mastering the art of scholarship. Scholarship involves researching, understanding and building upon the work of others and requires that you give credit where it is due and acknowledge the contributions of others to your own intellectual efforts. At its core, academic integrity requires honesty. This involves being responsible for ethical scholarship and for knowing what academic dishonesty is and how to avoid it."*
>
> *(University of Tasmania, 2010)*

This indicates the perception that ‘academic integrity’ is a value or a character attribute that should be possessed by and inculcated in an individual rather than something an institution could enforce through organizational measures, including technology.

2.2.2 Emergence of cheating within the sphere of online learning

The work of K. Sevnarayan and K. Maphoto [21] is indicative of the research focus on cheating within the sphere of online education. Using a South African distance learning and eLearning institution as the subject, it examines the factors that contribute

to cheating by students in an online learning context through qualitative research. It identifies that acts of cheating students during online assessments correspond to the methods of cheating encountered in traditional examinations such as copying and collaborating. It states that the opportunities for such cheating has been enhanced within the online scenario due to the availability of more sources, more convenient methods of copying, and more convenient and more easily concealed avenues of collaboration. This in fact, is technology assisted cheating and this study formally recognizes its existence.

Another significant finding of this study is that cases of cheating have increased with the advent of online learning. It acknowledges that online education provides more opportunities for cheating. The study also concludes that the lack of face-to-face supervision in online education and in online assessments in particular is a major contributory factor for increased cheating. This indicates that students with the motivation to cheat but are prevented from doing so within a conventional test environment would freely indulge in cheating during online tests, given the lax environment it provides.

The study also recommends a multifaceted approach to curb cheating and to enhance academic integrity. These include two technological countermeasures namely, the use of software assisted text-similarity identification to detect copying, and eProctoring to detect fraudulent acts during online tests.

K. Mate and J. Weidenhofer [22] have reviewed strategies for effective online assessments with particular focus on biomedical sciences . They also make the observation that academic dishonesty has increased with the use of online assessments. They quote the increased opportunity to cheat at online tests compared to traditional physical examinations as the reason.

The work of Kell et al. [23], is yet another qualitative research on the subject of academic integrity in the health science sector, using a South African University as its subject. It uses an interpretivist qualitative approach and the exploration about academic integrity is guided by a concept known as the 'Fraud Diamond'. The fraud diamond is an extension of an older concept known as the 'Fraud Triangle' (D. Cressey 1950, cited in [23]) that is used to predict individual inclination towards fraud. The three sides of an imaginary triangle are treated as representing three factors that causes a person to perform a fraudulent act under given circumstances, namely, motivation, opportunity, and rationalization. Fraud is committed when the triangle is complete. Taking the example of cheating at an online test, motivation could be parental pressure for good academic performance; opportunity could be the absence of proctoring in the online environment; and rationalization could be the knowledge that many other colleagues also indulge in cheating. The fraud diamond adds a fourth side to make the triangle a diamond and that side being the factor of 'capability'. This additional factor is significant to the current research where the sources of capability for the cheating student to indulge in fraud is being studied. Technology either provides or increases this capability.

The study has observed cheating during online assessments as being ‘innovative’. This innovativeness that is mostly derived through technology has been found to account for better performance during online tests.

The fraud diamond provides useful insights for the current study. Since the aim is to establish a system for the management of learner integrity during online assessments, figuring out ways to deal with the factors representing the sides of the fraud diamond could lead to ways and means of managing learner integrity.

Since the system is primarily technological, it cannot address the two factors of ‘motivation’ and ‘rationalization’ directly. However it can readily address the remaining two factors of ‘opportunity’ and ‘capability’. Shrinking the opportunity space for fraud and limiting the capability to indulge in fraud could form a good basis for the required system.

The work of Huang et al. [24], is primarily an attempt to identify relationships between individual traits and online academic dishonesty. Its findings provide important understanding of the role of technology as an ‘enabler’ as well as a ‘deterrent’ for fraud during online tests. One such finding is that students in the science stream are more inclined than students in the humanities stream towards cheating at online tests. Similar findings have been made related to students who spend more Internet time as opposed to those who spend less time. These findings point to the cheating students’ ability to exploit technology to indulge in fraud.

Technology aided academic dishonesty has reached its pinnacle with the recent advents in the field of Artificial Intelligence (AI). The study by C. Chan [25], involving 393 students from universities in Hong Kong, is an attempt to gauge the students’ perception of what the researcher calls ‘AI-giarism’. It provides important insights into the influence of AI on academic dishonesty and the new options for fraudulent practices it has placed at the disposal of cheating students. The possibility to use Generative AI (GenAI) tools for the convenient creation of content has radically changed the landscape of academic integrity and poses new challenges in maintaining it.

The paper highlights that before the advent of GenAI, AI in general has had many legitimate applications in areas such as medicine, administration, and entertainment, which in turn, has fueled its development. It states that the undue use of GenAI is primarily found within the academic sphere. It describes how the norms of academic integrity and the commonly accepted definitions of fraudulent acts like plagiarism are being questioned. Investing effort for content creation without the use of GenAI is being considered wasteful or futile in certain contexts. Prior to the advent of GenAI, copying involved considerable effort on the part of the cheating student where relevant sources had to be located, the content of interest had to be extracted, and cosmetic alterations had to be applied before submission to hide evidence of the origins. In contrast, GenAI tools when used cleverly, create almost submission ready content, considerably easing the plagiarizing task, earning the name ‘AI-giarism’. This paper defines AI-giarism as follows:

"AI-giarism refers to the unethical practice of using artificial intelligence technology, particularly generative language models, to generate content that is plagiarized either from original human-authored work or directly from AI-generated content, without appropriate acknowledgement of the original sources or AI's contribution."

This study does not distinguish between asynchronous and real time AI-giarism. However, the latter is obviously a major enabler of fraud during online assessments and particularly in those that require text based responses. It is a special case of the fraudulent act of 'lookup' described in Chapter 1.

The paper also identifies that students have a varied understanding about the legitimacy of AI-giarism and have widely contrasting views as to how AI-giarism amounts to academic misconduct. This is completely different to the situation concerning 'classical plagiarism' where the culprit is well aware that it is a fraudulent practice. The rapidly evolving nature of GenAI has only increased this ambiguity.

These four studies lead to the conclusion that technology has served as a significant enabler for the emergence of cheating within the sphere of online learning.

2.2.3 Attempts to curb technology aided academic dishonesty

In 2013, the US Department of Education had conducted a 'Testing Integrity Symposium' to address the issues of academic dishonesty in the testing and evaluation processes of the US education system. The report of this symposium [26] indicates the gravity of the issue of academic dishonesty. It also provides several recommendations for the prevention of irregularities in academic testing. The recommendations concerning online and technology-based assessments are of interest. The report recommends that Computer Based Testing (CBT) be conducted in a proctored environment and online tests be taken on a 'secure browser'.

Technology aided management of learner integrity is seemingly more challenging than technology aided academic dishonesty. The study by C. Chan [25] cited above briefly discusses the use of AI tools to detect cheating. It mentions that AI tools provide reliable detection of classical plagiarism, but may prove less reliable when detecting AI generated content.

The work of S. Pudasaini et al. [27] focuses on the use of technology for detecting asynchronous AI-giarism such as the use of Large Language Models (LLM) for creation of content for submission in response to assignments. It also mentions the availability of AI tools that can be used to paraphrase or make cosmetic alterations to AI generated content, to hide its origin. By using the two in cascade, detection could be evaded. The paper cites the results of a survey conducted in USA (E. Stern and L. Havlicek , "Academic Misconduct: Results of Faculty and Undergraduate Student

Surveys" on JSTOR, as cited by Pudasaini et al. in [27]) where 82% of undergraduate students have admitted their indulgence in some form of academic dishonesty during assignment submission.

The paper discusses several methods that could be employed for the detection of AI generated content. These include watermarking, zero-shot based methods, and binary classifier learning machines. Several studies that use each of these methods and having different detection accuracies are cited. It also mentions the availability of 'AI generated content detection evasion tools' which enables the cheater to make AI generated content undetectable. Competing use of technology for academic integrity and academic dishonesty is observed in such cases. The paper claims that the reliability of tools for AI generated content detection is low.

The study focuses on the detection of asynchronous AI-giarism. It does not discuss the detection of AI-giarism committed during online tests.

The work of D. Kundu et al. [28] is another study on the detection of asynchronous AI-giarism. However it uses an approach significantly different to the methods discussed in the work of Chan [25]. It uses 'keystroke dynamics' to identify differences between the typing patterns when a person types original content and content developed with AI assistance. The methods described by Chan [25] focus on the content of the submissions and use content matching as the primary technique of detecting plagiarism. The difference in the method used by Kundu et at. [28] is that instead of concentrating on content, it focuses on how the content was typed. It is based on the presumption that the typing behaviour of a person is scenario dependent. This is confirmed through experimental results where detection accuracies as high as 80% have been achieved.

The relevance of this research to the current study is the potential use of 'keystroke dynamics' for the detection of fraudulent acts although not within the scenario of online tests.

The work of F. Sabrina et al. [29] is a literature review on 'Ensuring Academic Integrity in Online Assessments'. They conclude among other things that ensuring academic integrity in online assessments is complex and that a single measure cannot curb all kinds of academic misconduct. It recommends the adoption of multi-faceted strategies.

The objective of the current study is to establish a system for the effective management of learner integrity during online tests. When this is compared with the status of technology assisted fraud prevention for the management of learner integrity as indicated in the studies that were reviewed, the following gaps could be identified.

- As far as online tests are concerned, detection of plagiarism aids in the detection of only the fraudulent practice of lookup. It does not aid in the detection of the fraudulent acts of impersonation or real time collaboration during online tests.

- Although keystroke dynamics based detections have been experimentally found to work well in scenarios involving asynchronous submissions, its effectiveness in the detection of fraudulent practices during online tests is presently not verified.
- Curbing academic dishonesty during online tests is limited to confining the candidates' online activities to the test using 'secure browsers'.

The above, coupled with the requirements identified in Chapter 1 that the system should be non-obtrusive, institution-ready, and capable of providing continuous authentication, provides the motivation to study the behavioural biometric named 'keystroke dynamics' in depth, with the possibility of its use as the primary detection mechanism in the system to be established. Accordingly, the subject of 'biometrics' is reviewed in general, proceeding to behavioural biometrics, and converging on their application in scenarios concerning learner integrity.

2.3 Overview of biometrics and biometric authentication

According to R. Hamaamin et al. [30], biometrics are the physical human characteristics and traits that are unique to an individual. Biometric authentication is the process of verifying the identity of an individual before granting access to a restricted system through the accurate and reliable recognition of one or more of such biometrics. Biometrics also refer to the systematic process of acquiring those characteristics and traits from the individual. The biometrics are initially acquired at the time an individual is 'registered' as a person who is admissible to the restricted system. These are then used to create a template, token, or profile uniquely associated with that individual and is stored within the system. When that individual, attempts to access the system subsequently, the biometrics are acquired once again and are compared against the ones stored in the profile. If sufficient correlation exists between the two sets of biometrics, the individual is recognized as the person registered and is granted access.

According to the "ITU-T Technology Watch Report of 2009" published by the International Telecommunication Union [31], a biometric should satisfy 4 prerequisites, to be used for authentication in the aforesaid manner.

First prerequisite is universality. That is, the metric should be present in the entire population of subjects or individuals. Second prerequisite is uniqueness. That is, the values or characteristics of the metric while being present in all individuals should nevertheless be significantly different from one individual to another so that it may be used for unique identification. Third prerequisite is collectability. That is, the metric should be readily acquirable. Fourth prerequisite is permanence or time invariance. That is, the biometric present within the individual should not vary significantly over

time so that their correlation with the initially captured instances of the metric is not degraded.

The most common and the oldest biometric that has had the largest usage is the “fingerprint”. The unique nature of fingerprints for identifying individuals had been known as way back as 1888. In that year, Francis Galton, in his paper titled “Personal Identification” [32], hypothesized that the tiny ridges on the skin of the fingers and palm of a human hand formed patterns of such diversity and complexity that they formed a large space of permutations to make the patterns of two arbitrarily chosen persons so different and uncorrelated to an extent that the two persons are uniquely identifiable. He further elaborated on his findings in three subsequent papers titled “Patterns in Thumb and Finger Marks”, "Method of Indexing Finger Marks.", and “Identification by Finger Tips" all published in 1891. In 1892 he published the book “Finger Prints” that particularly focused on the potential of using fingerprints as a reliable form of personal identification [33]. This spurred its worldwide usage, first in the area of crime detection and later, with the advent of computer aided image processing, in more common applications like “recording of physical presence” and “access control”.

The term ‘fingerprints’ was synonymous with ‘biometrics’ for a long time that followed, with no other biometric been identified for use. Nevertheless, as stated in the ITU report [31] cited earlier, due to the widespread use of ‘fingerprints’, the basic steps of biometric authentication comprising the initial capture, some form of template storage, and verification at the time of access using similarity tests became the standardized workflow in authentication systems.

Nearly a century later, “Retinal Scanning” as described in the work of R. Hill and R. Laub [34] and “Iris Recognition” as described in the work of L. Flom and A. Safir [35], both centered on the human eye, came into being as additional personal identification systems based on biometrics. They are now widely used in critical and zero tolerance applications like “immigration control” and “access control to sterile areas”.

The previously cited ITU report [31] states that Subsequently, a variety of other biometrics like ear scan, face scan, finger geometry, heartbeat, and scent or odour, came into use, culminating in DNA (Deoxyribonucleic acid) testing, the ultimate indicator for pinpointing individuality.

2.3.1 Physiological biometrics

The applications like fingerprint scanning, iris recognition, and retinal scanning are all members of the general class of biometrics called “Physiological Biometrics”. As stated in the work of I. Alsaadi [36], they involve biometric parameters directly linked to a person’s physiology. All examples ranging from fingerprints to DNA that were discussed earlier belong to this class. All these technologies have presently found

widespread usage in access control and user authentication under the "something you are …." category.

As further elaborated in the work of Alsaadi [36], an important characteristic of all types of Physiological Biometrics is that they are inevitably obtrusive. They cannot be acquired without the user being aware of it or without his cooperation. Capturing them often require specialized devices and technologies. Some need to be acquired under specific environmental conditions. This implies that physiological biometrics are not suitable for systems that require the authentication of persons to be carried out without the person being aware of it or without requiring the person's explicit cooperation. Galbally, Fierrez, et al. [37], cite the possibility for an intruder to submit the biometrics of the genuine user through 'forced biometric presentation' or 'coercion attack' also as a disadvantage.

2.3.2 Behavioural biometrics

An alternative class of biometrics is "Behavioral Biometrics". As stated in the work of M. Sharma and H. Elmiligi [38], when using this class of biometrics, certain unique peculiarities exhibited by individuals when performing certain activities both consciously and unconsciously are captured. Eye blinking patterns, walking, and Electrocardiogram (ECG) are quoted as examples.

An informative source on behavioral biometrics is the work of R. Yampolskiy and V. Govindaraju [39]. It contains an analysis of 28 behavioral biometrics under several aspects including the time spent on capturing them and the hardware required.

A generalized algorithm that may be adapted for authentication using any behavioral biometric is also presented.

As stated in the previously cited ITU report [31], an important advantage with certain behavioral biometrics is that they could be captured non-obtrusively and without requiring the user's cooperation, thus making them more suitable for user authentication and intrusion detection. While physiological biometrics are almost always obtrusive, certain behavioral biometrics are totally non-obtrusive. Examples are keystroke dynamics and Gait[1]. Not all of them require specialized capturing devices or technologies.

G. Hogben, in a briefing bulletin published on behalf of the European Network and Information Security Agency (ENISA) in 2008 [40] identifies five distinct factors that are important for the successful implementation of a behavioral biometric system. The equipment required has been listed as the first factor. Use of equipment makes the biometrics acquisition conspicuous. This publication also mentions the negative

[1] Characteristics peculiar to a person's limb movement while walking

reaction on the part of the end-user community towards the obtrusiveness introduced by the equipment used for acquisition. The work does not generalize behavioral biometrics as being non-obtrusive compared to physiological counterparts but states that there is a high variance between the obtrusiveness associated with the capture of different biometrics.

The advantage of behavioral biometrics over physiological biometrics are also mentioned in the previously cited work of Yampolskiy and Govindaraju [39]. The authors have established through surveys that users prefer behavioral biometrics to physiological biometrics due to its zero or low obtrusiveness.

M. El-Abed et al. [41] have conducted research on the users' acceptance and satisfaction on biometric systems. Using a panel of 100 volunteers they have conducted a survey on the acceptability of keystroke dynamics and facial recognition, the first being a behavioral biometric and the second, a physiological biometric. They have established that users prefer keystroke dynamics based systems which are inherently non-obtrusive.

2.3.2.1 Keystroke dynamics

N. Chourasia and P. Joeg [14], define 'keystroke dynamics' as an 'automated method of identifying an individual based on the manner and the rhythm of the typing of the user'. They also assert that it is a behavioural biometric. They identify keystroke duration, keystroke latency, typing speed, typing error rate, force exerted on keys, and preference of modifier keys, as authentication features in keystroke dynamics.

Nowadays, a keyboard would usually mean a computer keyboard and keystroke dynamics would generally refer to the unique typing traits of a user when interacting with a computer via the keyboard.

A widely cited work that provides an extensive survey on keystroke dynamics biometrics are the work of Pin Shen Teh et al**.** [42]. It contains the following description about keystroke dynamics which is one of the earliest and is also very comprehensive:

> *"Keystroke dynamics refers to the process of measuring and assessing human's typing rhythm on digital devices. Such device, to name a few, usually refers to a computer keyboard, mobile phone, or touch screen panel. A form of digital footprint is created upon human interaction with these devices. These signatures are believed to be rich in cognitive qualities which is fairly unique to each individual and holds huge potential as personal identifier"*

According to this paper, the immediate predecessor to keystroke dynamics is very old and dates back to the era of telegraph communication. The early telegraph operators

were known to have been able to recognize their colleague at the remote end by carefully monitoring the unique traits in the Morse beeps caused by the colleague's finger manipulation of the telegraph key. Being able to identify the remote operator from the manner he manipulates the telegraph key with no other indication about it, fully conforms to the definition of behavioral biometric identification. Hence it is one of the earliest known applications of behavioral biometrics and is also the predecessor to keystroke dynamics in its present sense.

An accepted principle in the field of 'Ergonomics', the science that deals with the design of systems for convenient user-system interaction, is that in order to yield standardized usage of a device, it is necessary to design the device ergonomically, that is, with the convenience of the user as a design objective. This is highlighted in the work of M. Helander [43]. Devices designed in that manner would cause all users to soon be accustomed to a standard usage pattern. This implies the converse that a device with poor ergonomic design would exhibit highly variant usage patterns among users. Ciobanu et al. [44] argue that the QWERTY[2] keyboard layout is not ergonomic. Stating that the QWERTY design layout was standardized in 1873 to minimize the clashing of key hammers when mechanical typewriters were used, they argue that this optimization is irrelevant for computer keyboards which do not drive any mechanics. When these two are considered jointly, it could be inferred that 'keystroke dynamics', being the usage traits of people using a computer keyboard which is a device with poor ergonomics, is bound to have a high variance in those traits, forming a large permutational space of such traits that accommodates traits that are likely to be unique to individuals.

The work of S. Banerjee and D. Woodard [45], includes a description of the 'psycho-physical process' related to typing. It breaks the typing process into 3 distinct stages including the role of the brain as a short term buffer and the issue of commands for muscle movement. These further account for the differences among typing traits between typists.

Romain Giot, et al. of GREYC Research Lab, [46] state that the concept of keystroke dynamics had been conceived in 1975 and its use had been successfully verified in the early 1980s. They state that in the earliest systems, a user's individual profile had been established only after observing the typing of a fairly long sequence of characters.

The work of Giot, et al.[46] also describes two types of authentication that could be realized using keystroke dynamics. One is called 'Static Authentication'. In static authentication, the user is required to type a fixed character sequence. Also, it is monitored only at a particular instance of the session, usually at the time of login. The two factor authentication of passwords falls into this category. During enrolment the user may be required to input the password many times (as much as 20) for the

[2] The keyboard descriptor QWERTY is derived from the character sequence Q-W-E-R-T-Y, found on the 6 leftmost keys in the top row of keys on that type of keyboard.

keystroke dynamics to be correctly acquired. During subsequent logins, in addition to matching the password sequence, the keystroke dynamics are also matched. Since it is impossible for an imposter with knowledge of the password to replicate the keystroke dynamics of the genuine user, the system can detect the unauthorized access. The other type is called 'Dynamic Authentication' where the authentication does not rely on a fixed sequence of characters. When using this type, the monitoring need not be confined to a particular instance of the session. The typing could be continuously monitored. This is called 'Continuous Authentication'.

The majority of keystroke dynamics based applications rely on time-based data, that is the time differences between keyboard events. This is because timing data are the most easily captured with no requirement for additional hardware or accessories, which is a primary concern in making the authentication process invisible to the user. However, the capture of other metrics connected with key depressions such as the pressure exerted on the keys could provide metrics which are even more particular to a user, thus increasing the effectiveness of the authentication process by many fold. H. Nonaka and M. Kurihara [47] have examined the sensing of pressure exerted on keys in addition to the time measurements as a means of improving the effectiveness of keystroke dynamics authentication. They have used two pressure sensors comprising piezo-resistive films placed under two selected keys. While the increased effectiveness of the authentication is beyond doubt, the system becoming visible to the user due to the additional accessories would limit the deployment only to applications where the registered user's consent is available.

The work of P. Pisani and A. Lorena [48] is a systematic review of research on the subject 'Keystroke Dynamics'. An important observation made in this study is the level of research interest on the subject. Figure 2.1 shows the number of publications made on the subject from 1998 to 2012.

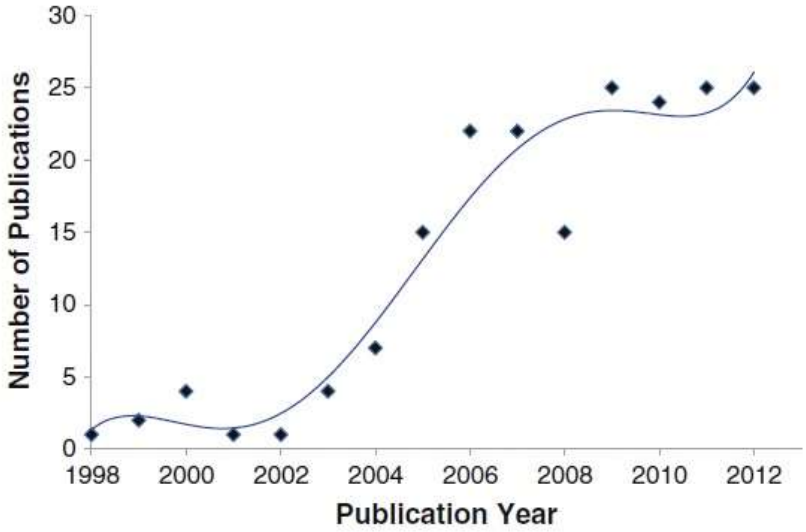


Fig. 2.1: Growth trend of research on 'keystroke dynamics' (Source: work of P. Pisani and A. Lorena [48])

As seen from the figure, research interest has grown rapidly between 2002 and 2008 and has saturated somewhat thereafter. This implies that the current study is being undertaken at a time the subject has reached considerable maturity.

2.3.2.2 Rhythm-Click Characteristics

Rhythm-Click characteristics could be regarded as the non-keyboard input device equivalent of keystroke dynamics.

The work of Sayed et al. [49] is an example of a study on the use of rhythm-click characteristics for user authentication. It particularly focuses on static user authentication at initial login using rhythm-click characteristics. It uses the words 'mouse dynamics', indicating its analogy to keystroke dynamics.

In the work of Ahmed and Traoré [15], mouse dynamics is described as 'the characteristics of the actions received from the mouse input device for a specific user while interacting with a specific graphical user interface'.

The work of Khan et al. [50] is a survey on the use of rhythm-click characteristics for user authentication. It includes widget interaction and GUI interactions on touchscreens also into the domain of rhythm-click characteristics in addition to mouse dynamics. It concludes that user authentication using rhythm-click characteristics is relatively new.

The above studies infer that, as in the case of keystroke behavior, the manner that an individual is inclined to use the buttons on a pointing device, the stylus on a touch sensitive screen, or the numeric keypad of a mobile device, would have characteristics and traits that are inherent to the individual.

H. Gamboa and A. Fred [51] have proposed a classification scheme that analyzes the rhythm-click patterns of a two button pointing device which they have found to be sufficiently powerful for user identification. In their experiment, they have used a web based memory game which the subjects were required to play. Data for the training phase had been collected from 50 users playing the game for 10 hours. Since the game causes all players to produce identical series of clicks, it is in effect, static authentication. In addition to the timing information they have also used the 'spatial information' of the movements of the pointing device. They had realized an Equal Error Rate (EER[3]) of 2% using a 30 seconds interaction period, 1% using a 60 seconds interaction period and 0.5% using a 90 seconds interaction period. Since 'spatial data' has been used, this system strictly does not belong to 'rhythm-click characteristics' category. However, it still qualifies as an effective behavioral biometric.

[3] EER, is a metric used to assess the performance of authentication systems. Its definition is covered in section 2.6.

Ting-Yi Chang et al. [52] have also conducted a study similar to the aforesaid one in many aspects. Instead of using a game as the collecting element these researchers had used two popular tunes to provide target rhythms for clicking the pointing device. The rhythm clicks are captured and converted to a template which the authors of this paper argue as being unique to the user. They have proposed this system to be used for Two-Factor Authentication (2FA) in conjunction with a password scheme. They claim that it would be very effective because even an imposter who has gained knowledge of the password would still be unable to spoof the rhythm click characteristics of the genuine user. As the rhythm clicks are captured only at the time of login and because the same target rhythm is used, this too is a case of static authentication.

The last two studies described above, propose rhythm-click characteristics as a supplement to other authentication techniques and not as a standalone authentication scheme.

These work indicate that rhythm-click characteristics could be treated as a promising behavioral biometric scheme that is still in the stage of formation.

2.4 Applications of Keystroke Dynamics

2.4.1 Principles and Techniques

There are many ways in which effective static authentication has been realized using keystroke dynamics. Sungzoon Cho et al. [53] have presented a web-based system that exploits keystroke dynamics and uses neural networks for authentication. During registration, the user is required to type in a fixed sequence '*n*' characters long which results in the extraction of a timing vector of dimension '*2n + 1*'. An auto-associative neural network is then used to create a template that is unique to the user which is used for authentication at subsequent logins.

K. Killourhy and R. Maxion have published their findings [54] on the comparative performance of 14 different algorithms that are in industrial use for keystroke dynamics based static authentication. The list of these algorithms is as follows:

i. Manhattan
ii. Manhattan (scaled)
iii. Manhattan (filter)
iv. Mahalanobis
v. Mahalanobis (normed)
vi. Neural network (standard)
vii. Neural network (auto-associative)
viii. Nearest Neighbour (Mahalanobis)
ix. Euclidean
x. Euclidean (normed)
xi. Outlier Count (z-score)

xii. Support Vector Machine (one-class)
xiii. K Means
xiv. Fuzzy Logic

The paper offers a concise description on the operating principle of each type of algorithm and thus constitutes an informative source on the subject.

The authors had implemented each algorithm using R. They have tested each of the algorithm for the 2-factor password authentication of 51 individuals and found that the Manhattan (scaled) algorithm gives the best performance in terms of EER with an average value of 9.6%. The nearest neighbor (Mahalanobis) and outlier count algorithms follow with average EER of 10% and 10.2% respectively. The poorest performance in terms of EER had been found to be the standard neural network with a value of 82.8%. The paper also presents the algorithms ranked on the zero miss False Rejection Rate (FRR[4]).

As mentioned earlier, the most commonly used and most non-obtrusively captured keystroke biometrics are the timing vectors of the typing sequence. The following explanation is derived from the previously cited work of Pin Shen Teh et al**.** [42] which contains a detailed description of the measurements that are captured.

When a person types on a keyboard, there are two 'events' associated with every keystroke made. When a key is pressed, a 'key down' event occurs. When a pressed key is released, a 'key up' event occurs.

There are basically two timing measurements that could be extracted from the key events:

- Dwell time (DT)
- Flight time (FT)

Dwell time DT_i is the time that the 'i^{th} key' is held depressed. Flight time in general is the time between two key events, say $i-1$ and i. There are 4 different types of flight times as illustrated in Figure 2.2.

When two consecutive key events are considered, the time measurements are called di-graphs. When measurements are between keys 'n' depressions apart, they are called n-graphs. n-graphs may be defined by suitably extending the above definition of flight time.

The dwell time would inevitably have some device dependence. This means that the value for the same typist could be slightly different on a different key board depending on the mechanical properties of the depression mechanism of each key. However, to a larger extent, it would depend on the typist's typing traits and therefore its effectiveness as a biometric is not diminished.

[4] Definition is given in section 2.6 ahead.

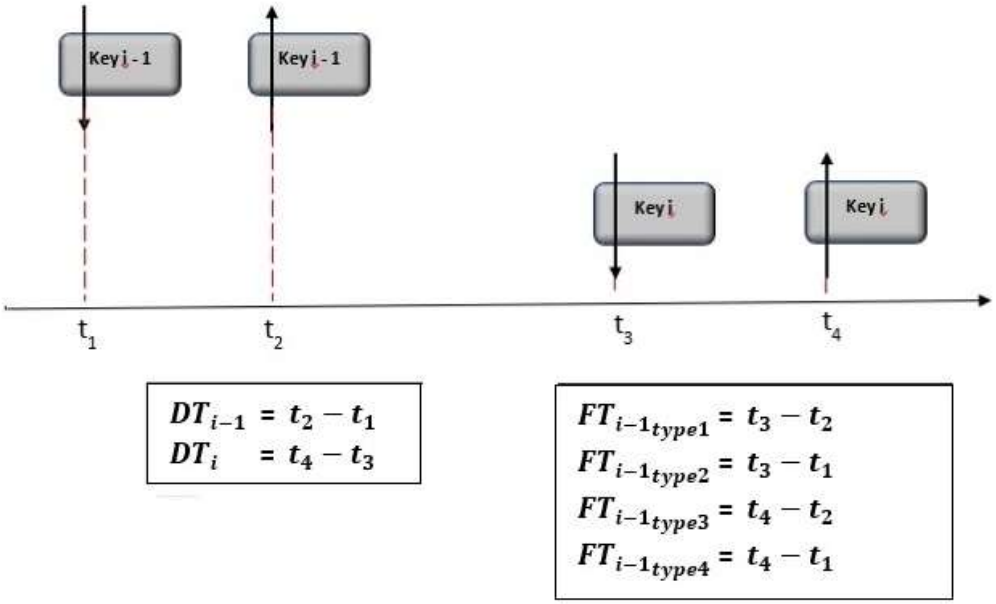


Fig. 2.2: Definitions of timing measurements on keystroke events

In contrast, the flight times of type 1 between two key presses will almost entirely depend on the typist and the two keys. It will have negligible device dependence due to the influence of the different physical sizes of keyboards and the difference in location of certain keys on different keyboards.

For a sequence of 's' characters there would be two vectors extracted as follows:

$$V_{DT} = \{DT_1, DT_2, DT_3, \ldots\ DT_s\} \quad \ldots\ldots\ldots (2.1)$$

$$V_{FT} = \{FT_1, FT_2, FT_3, \ldots\ FT_{s-1}\} \quad \ldots\ldots\ldots (2.2)$$

These vectors are quasi-analog because technology is available to capture these timings with millisecond accuracy.

2.4.2 Strengthening conventional authentication

Review of many studies on keystroke dynamics show that the most common use of keystroke dynamics is the strengthening of conventional static authentication. Using it for independent authentication on its own is not so widespread.

The work of Wahab et al. [55] investigates the use of 2FA for login and for account recovery. In addition to typing the password, the user is prompted to type in a 'fixed character sequence'. When used in this manner, keystroke dynamics are used in the fixed authentication mode described earlier. The researchers have managed to reduce authentication friction by 66.3%. Authentication friction as defined by Glass et al. [56], is a measure of the inconvenience endured by a user in the process of being authenticated.

An alternative that could reduce authentication friction is to acquire keystroke dynamics of the target population from the typing during their routine work. Such collection would require a long observation period. An example is the work of F. Monrose and A. Rubin [57]] where data for profiling had been acquired over a period of 11 months for a keystroke dynamics based password authentication system.

Another study by Giot, Rosenberger et al. [58] uses 'passphrase authentication' using keystroke dynamics at initial login. In addition to the password, the user is prompted to type a passphrase provided by the system administrator. The keystroke dynamics captured from the passphrase is compared with a template created earlier. Template creation has used the repeated typing of a token passphrase only 5 times, resulting in considerably low friction.

Alternatively, the password itself could be used as the source of biometrics instead of using a passphrase. The work of Sluganovic et al. [59] is an example of such a system. This provides additional protection in three ways. Firstly, knowledge of the password would be required for both stages of authentication. Secondly the user templates would have better decorrelation from each other because passwords of two users will generally be different. Thirdly, the biometric authentication is invisible to the user. However the authentication friction at initial registration in this system had been high, requiring the password to be entered 200 times to form the templates.

The work of Araújo et al. [60] is another example of a system where the keystroke dynamics of the password itself had been used for fixed authentication at login. Registration had required the typing of the password 10 times or less.

2.4.3 Use as a Soft Biometric

In the work of Jain et al. [61] soft biometrics are defined as 'traits and characteristics that provide some information about an individual, but lack the distinctiveness and permanence to sufficiently differentiate any two individuals'.

Accordingly, Soft biometrics are not concerned with the unique identification of a user. Instead, it is concerned with classifying the user into one of two or more classes. For example, the system could identify the user's gender.

When an authentication system is used in conjunction with a soft biometric, it greatly enhances the effectiveness due to the imposter's inability to simultaneously belong to a large soft biometric space of which no knowledge is available, in order to evade detection.

The work of Jain et al. [61] constitutes one of the earliest research in the sub field of soft biometrics. The paper presents a framework for integrating the ancillary information such as gender and age which are generally collected from users at enrollment, with the output of a primary biometric system such as fingerprint scanning. Using experiments on 263 persons, they have shown that through the use of ancillary

information of users such as their gender, height, or ethnicity, the personal identification capability of a fingerprint system could be improved by nearly 5%.

Another use of soft biometrics is 'two stage identification' of subjects. A soft biometric trait is initially used to classify the subject. Once this classification is achieved a more elaborate matching is attempted among subjects of that class. In an authentication scenario, rejection may be decided immediately upon the failure of the first classification. This improves the overall efficiency of identification or authentication. The work of Arigbabu et al. [62], is an example of the use of this technique in facial recognition.

Keystroke dynamics may also be used as a soft biometric. R. Giot and C. Rosenberger, in a research carried out in 2014 [63] have demonstrated the recognition of the gender of a user through keystroke dynamics. A predefined text has been used to collect biometric input. They have been able to achieve an EER of 20% by incorporating soft biometrics with static authentication based on keystroke dynamics.

Research on gender recognition using keystroke dynamics as a soft biometric have also been carried out by A. Buker et al. [64]. A. Pentel [65] has successfully classified persons on their dextrality[5], using their keystroke dynamics.

Idrus et al. [66] have developed a benchmark testing suite that uses keystroke dynamics in the 'soft biometrics' mode. They present a dataset containing the keystroke biometric data of 110 users with 100 samples per user classified by gender, age, and dextrality.

2.4.4 Use of non-temporal features

The work of Kumar et al. [67] defines non-temporal features as keystroke dynamics features that are not directly derived from the time measurements of individual keystrokes. Examples are the typing speed, error rate, and the error correction mode. The paper discusses the computation of typing speed from keystroke data. Non-temporal features may be combined with time related keystroke dynamics to enhance the capability of an authentication system. This has been done in the central study presented in this paper where the typing speed had been used together with the time related features.

According to Kumar et al. [67], typing speed is a popular indicator used to assess and rate professional typists. In the typing trade it is usually measured in 'words per minute (WPM)'. Since a 'word' does not have a standard length, 'characters per minute' becomes a more accurate indicator in scientific contexts. There is also a practice that considers a word as 5 consecutive characters and computes the 'words per minute' as one fifth of the 'characters per minute'. The error correction mode is also a personal

[5] Being left-handed or right-handed.

trait of the typist. Some typists prefer to use the delete (DEL) key to rubout a mistake starting from left, while others prefer to use the BACKSPACE key to rubout the mistake starting from the right. Some would use arrow keys to get to the mistaken word while others may prefer to use the pointing device. Over a long time, each person will get accustomed to a certain mode.

In the work of I. Tsimperidis et al. [68], in which a classification system had been developed to classify people on their educational level, non-temporal features had been used to reinforce the keystroke dynamics based classifier to obtain an accuracy of 86.8%.

2.4.5 Other applications of keystroke dynamics

Keystroke dynamics have been applied in areas other than single user authentication and access management.

The work of Epp et al. [69] is a research in the field of intelligent computing and emotionally-aware systems. They have been motivated by the fact that most emotion sensing schemes are invasive and expensive in deployment. They have focused on keystroke dynamics as a means of overcoming those limitations. 15 distinct emotional states including joy, anger, frustration, confidence and relaxation had been identified. The keystroke biometrics had been captured by getting each user to type a randomly selected piece of text from a single book. The authors claim that 84% accuracy has been achieved in detecting certain emotional states.

Another research by Giot et al [70] has focused on the use of keystroke dynamics for authentication in 'Collaborative Systems'. Collaborative systems are those where members of a team (say, a team working on the same project) use the same access credentials. This paper proposes the use of a single common template built upon the keystroke dynamics of all members using score fusion techniques, which would admit all team members while rejecting any other.

The work of Shadaman et al. [71] describes more applications of keystroke dynamics such as the prediction of typists' native tongue, detection of typist fatigue, and the detection of certain ailments and health decline.

2.5 Continuous Authentication (CA)

Even before the work of D. Temoshok et al. [13] cited earlier, which formally stipulates that authentication during online systems should be continuous, there have been studies that conclude about the necessity for continuous authentication of online sessions. A notable feature of these studies is that 'keystroke dynamics' is not identified as a modality for such authentication.

In the work of K. Niinuma, A. Jain [72] carried out in 2010, authentication only at the instance of initial access is identified as a security flaw due to the possibility of subsequent user switches. It highlights the need for continuous authentication throughout the session and proposes a webcam based authentication system. In their review of previous work, keystroke dynamics based continuous authentication is explicitly omitted due to lack of use cases. This implies that although continuous authentication was a research focus in 2010, keystroke dynamics being used for that purpose was not.

In September 2016 Ioannis C. Stylios and 3 others have published a paper titled "A Review of Continuous Authentication Using Behavioral Biometrics" [73]. It is basically a literature review. It highlights that fixed authentication systems are inadequate and ineffective in protecting systems from intrusions. It proposes the use of systems that continuously monitor the interaction and usage. The paper reviews a total of 46 research papers and particularly focuses on mobile devices and use of behavioral biometrics. However, the absence of any mention about 'keystroke dynamics' is significant. This suggests that keystroke dynamics, while having a large space of use cases in 'fixed authentication', does not appear to be having a similar share of applications in 'continuous authentication'.

Charles C. Tappert et al. [74] have published a research paper on the use of keystroke dynamics with long text input for user authentication. Use of long text input for authentication does not necessarily imply that the authentication is continuous. Yet, it is a step towards that direction because continuous authentication would require the processing of long text input. In this study, sequences as long as 650 characters have been used. Using statistical parameters of the key timings of the long character sequence and using a Euclidean distance nearest neighbor classifier, the authors have been able to create an effective authentication system. This approach could be adaptable to the continuous authentication scenario.

H. Çeker and S. Upadhyaya [75] have also conducted a research that deals with long text based authentication using keystroke dynamics. They have used a one class - Support Vector Machine based approach. The data collection interface used in their experiment forces users to provide inputs at least 500 characters long. They claim that their system could be used for continuous authentication of users.

The work of P. Bours and H. Barghouthi [76] comes very close to the requirement for continuous authentication. Keystroke dynamics acquired from strokes of a single key as well as strokes resulting from combinations of several keys have been used to create templates. Experiments have been carried out on 25 participants. They too have used statistical parameters of the keystroke biometrics as in the previously quoted research.

The doctoral research of Eesa Alsolamy [77] aims to develop a generic model for Continuous Biometric Authentication Systems (CBAS). It also proposes 4 statistical-based techniques for feature extraction. It classifies existing CBAS into two classes depending on the availability or non-availability of imposter data while introducing a

new class. The experiments use sliding window techniques to deal with the continuous stream of text input. The literature review of this thesis states that there is only a small number of published work dealing with 'free text' based continuous authentication,

T. Sim and R. Janakiraman [78] have studied the suitability of di-graphs for free-text key stroke dynamics. They have showed that di-graphs should be 'word-specific' for their use to be successful in keystroke dynamics. This is an important finding. It suggests that picking out specific character sequences (for instance, the most frequent words) from the user's input stream and analyzing their metrics would be a better approach rather than analyzing every character sequence typed.

2.6 Performance assessment of biometric systems

The reviews discussed in this section on how the performance of a biometric system could be assessed and how the performance of two systems could be compared, are generic, and are not limited to behavioral biometrics or any particular biometric.

The following 3 metrics defined in the ENISA briefing cited earlier [40], have been traditionally used to gauge the performance of an access control system.

- False Acceptance Rate (FAR)
- False Rejection Rate (FRR)
- Equal Error Rate (EER)

False acceptance rate or the miss rate is the percentage of individuals who are not authorized to use the system but have been wrongly cleared to access the system. False rejection rate or the false alarm rate, is the percentage of individuals who are authorized to access the system but have been wrongly denied access to the system. Both are consequences of the authentication system failing to perform correctly.

The relationship between these metrics are dealt with in many research articles and the following explanation is derived from the work of Al-Hassam et al. [79].

As illustrated in Figure 2.3, the parameter 'X' is generic and represents the sharpness or the sensitivity of the authentication mechanism. When X = 0, that is when the system is not used, FAR would be 100% while FRR would be 0%. Gradually increasing 'X' would make the system more and more capable of detecting imposters and the FAR would drop. In a practical system, however much 'X' is increased, the FAR is unlikely to reach absolute zero.

False rejection may not occur until 'X' increases beyond some minimum value. As 'X' increases further, the system becomes sharper in its capability to detect imposters, but also begins to reject some genuine subjects. This will increase with increasing 'X'.

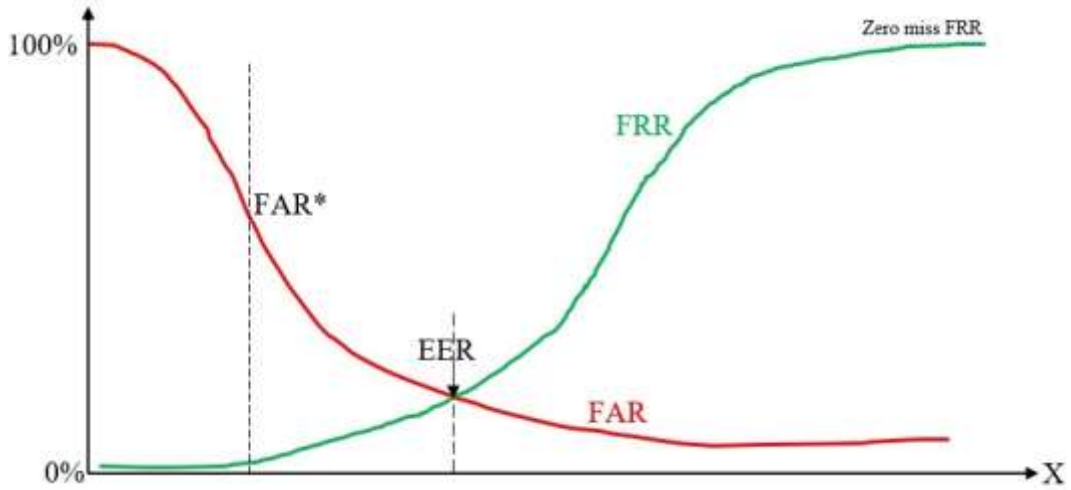


Fig. 2.3: Variation of assessment metrics with sharpness of the authentication system

The equal error rate or EER, is the value of FAR and FRR where the two become equal or where the two curves intersect. It is a common practice to set 'X' to this value for optimum performance of the system. Lower the EER, better the system. For this reason, the EER is often considered as an important metric for performance assessment of biometric systems.

The value of FAR just as the FRR begins to pick-up is also an indicator of good performance. This value (shown as FAR* in Figure 2.3) is an indication as to how good the system is at rejecting imposters when it is at the verge of rejecting genuine subjects. Hence the value FAR* is a threshold and lower the FAR*, better the system.

On the average, physiological biometric systems could exhibit a lower EER and function more reliably, according to the previously cited ITU report [31]. The growing preference for behavioral biometrics over physiological biometrics for access management is purely due to their non-obtrusive nature and their not requiring specialized devices.

2.7 Benchmark Datasets

An important resource for research on keystroke dynamics are the 'Benchmark Data Sets (DB)'. Benchmark datasets include keystroke dynamics of volunteers (usually anonymous) usually captured during routine typing. They are stored securely and is accessible to researchers upon request.

According to Giot et al. [46], the following are 5 popular benchmark datasets available to the research community:

- DB1 Chaves
- DB2 DSN2009

- DB3 GREYC Alpha
- DB4 (For pressure sensitive Keystroke Biometrics)
- DB5 (Most recent)

Giot et al. also present a benchmark testing suite for research on keystroke dynamics [80]. It comprises the GREYC dataset listed above and a software application called 'GREYC – keystroke' which may be used to create keystroke dynamics databases.

Keystroke dynamics benchmark datasets are very useful in conducting experiments and in evaluating the performance of systems. Their primary usefulness stems from the elimination of the need to capture data from subjects solely for the purpose of experimentation.

2.8 Applicability to online authentication

Most of the work reviewed up to now are those where the experimental protocols do not particularly deal with authentication during online access. In most cases, keystroke data acquisition and authentication had been done locally.

The work of Cho et al. [53] cited earlier is an example where the test apparatus itself was web based and therefore involved authentication for online access. The keystroke data had been captured on the client side and conveyed to the server for authentication.

The additional consideration applicable when using keystroke dynamics for online authentication is this client-server scenario where the authentication is performed by a server side program whereas keystroke data are captured on the client side.

2.9 Performance assessment of continuous authentication systems

In section 2.6, the metrics used for assessing the performance of access control systems were discussed. Those metrics are usually applicable to biometrics used for initial access control. The access control industry has now realized that those metrics (often called 'traditional metrics') are inadequate to assess the performance of continuous authentication systems.

T. Sim, in his webinar [81] highlights the importance of 'time related metrics' in continuous authentication. Since the subject is continuously monitored, if impersonation has been achieved after initial clearance by some means, then how soon would that be detected, is of interest. Similarly, since no system is perfect, a continuous authentication system may also exhibit some number of false acceptances as well as false rejections. How long do such false detections persist is another matter of interest. Traditional metrics give no indications of these. Therefore, additional metrics have been defined particularly for assessing continuous authentication systems.

In a paper titled 'Performance Evaluation of Continuous Authentication Systems' published by P. Bours and S. Mondal in 2015 [82], two new metrics for the exclusive performance evaluation of continuous authentication systems are proposed. They are the Average Number of Imposter Actions (ANIA) and the Average Number of Genuine Actions (ANGA). While the metrics are generic to any continuous authentication system, their experiments are based on keystroke dynamics and rhythm click characteristics. Both metrics are computed over a span of time. For the system to be good, ANIA should be low while ANGA should be high.

The work of Shadaman et al. [71] cited earlier, also describes how these two parameters ANIA and ANGA may be applied when keystroke dynamics are particularly used for continuous authentication.

The work of Senarath, et. al. [83] titled 'Re-evaluating Keystroke Dynamics for Continuous Authentication' is another example that uses time based metrics for evaluating continuous authentication systems. They use an alternative set of metrics. They are the False Rejection Worst Interval (FRWI), False Acceptance Worst Interval (FAWI), and the Time to Correct Reject (TCR). False rejection intervals are the time intervals a genuine user is rejected continuously while false acceptance intervals are the time intervals an imposter goes undetected continuously. The FRWI and FAWI are the longest of such intervals respectively. Time to Correct Reject is the time elapsed before an imposter who has managed to gain initial access is subsequently detected by the system for the first time. These indicators which are time based are considered as being more relevant because they provide more realistic indications about the effectiveness of continuous authentication systems. For the system to be good, all three metrics must be as low as possible.

2.10 Device Dependence

The dependence of keystroke dynamics of a user on the device being used is another area that has received research focus. This is reflected in the previously cited work of Pin Shen Teh et al, [42] and Romain Giot, et al. [46].

Ciobanu et al. [44] state that although the basic key layout on a QWERTY keyboard is standardized, the locations of many other keys frequently required for input are not standardized. When a user switches from a keyboard he is generally familiar with to another, it would inevitably cause some changes in his keystroke behavior due to additional pausing and the difference in the ergonomics of the keys.

Trojahn et al. [84] have examined the extension of keystroke dynamics to touchscreen keypads. Their experiment had involved the typing of a fixed sequence of 17 lowercase characters by 152 individuals on the same touchscreen device. Their findings show that the use of keystroke dynamics for user authentication could be extended to

touchscreens. However, the authentication referred to in this research is not the authentication for online access but the control of access to the device itself.

The work of Ahmed et al. [85] discusses the authentication of users for mobile phones and handheld devices with pushbutton keypads. They mention that the typing behavior of users on such devices is likely to be different from those on a conventional physical keyboard due to the tendency to use a fewer number of fingers and having to hold the device in hand while typing. The authors state that the keystroke dynamics combination space on such devices would be much limited and could be inadequate to recover unique traits for profile creation. The access control referred to in this study is also the access to the device and not the access to applications or online access.

2.11 Challenges in applying authentication to Learner Integrity Management

The main difference between the scenarios of authenticating candidates undergoing online assessment and those authenticating user access to other protected online systems is that in the former, there is collaboration between the registered user and the imposter.

In the work of I. Buhan and P. Hartel [86] which deals with biometrics based access control to physical facilities, the authorized user knowingly providing his/her biometric sample to the imposter is called ‘collusion’. C. Gottschlich, in his work [87] dealing with ‘Fingerprint Verification’, refers to it as ‘skilled imposter attacks’. Neither of these work deal with online assessments in particular, which implies that the inducement for an authorized user to enlist an imposter may exist in any access control system.

The work of Coghlan et al. [88] deals with online proctoring technologies and refers to the impersonation of candidates by competent colleagues as ‘contract cheating’ and ‘proxy test-taking’. The proctoring referred to here is eProctoring that was explained in 2.2.2.

Impersonation is only one of the challenges the online test administration needs to deal with. Collaboration, tapping and look up during the test also needs to be curbed. This is recognized in the work of K. Becker and H. Meng [89] where both ‘proxy test-taking’ and ‘unauthorized lookup of content’ are dealt with. However the detection scheme proposed is ‘offline collusion analysis’ performed after the test.

It is also evident from the foregoing that authentication done at a particular instance during a session cannot prevent cheating. This implies that fixed authentication is not suitable for managing learner integrity in online assessments.

What is necessary for effective management of learner integrity is a deterrent towards cheating and not necessarily a zero FAR system. When candidate community is aware

that the testing system is capable of detecting fraudulent practices during the test, they would be reluctant to attempt such acts. In the studies [17]. [18], [19], and [20] reviewed earlier, it was revealed that having an effective fraud detection system in place and communicating that to the student community as part of the academic integrity policy of the institution would considerably deter fraudulent acts. This is different to the case of prevention of unauthorized access to critical online applications where the intruder is a professional cybercriminal with a host of tools at his disposal who would go to any length to break the system. This view is supported by another study by D. McCabe and L. Treviño [90], where distinction is made between student cheating at online assessments and the intrusion into secure services carrying high stakes by cybercriminals. Opportunity acts as a major motivator in the former while the latter is mainly motivated by financial gain.

Given these differences, it is now possible to revisit and refine the minimum specifications for a learner integrity management system of the type mentioned in Chapter 1.

i. The system should be totally invisible to the user and should run in the background. This invisibility should be further ensured through the following:
    a. Not requiring the cooperation of the user in any manner by way of having to comply with any conditions or provide any interactions, other than those required by the test.
    b. Not requiring additional hardware or accessories to be deployed at the user site.
    c. Not exerting undue demands on the resources such as connection bandwidth that may become an indication of the background processes.
ii. The system should provide for the continuous authentication of the user through the continuous monitoring of the keystroke dynamics of the user inputs provided in response to the test, as opposed to fixed authentication performed at a particular instance.

2.12 Challenges posed by time variance and strategies for coping

The variation of behavioral characteristics and traits of an individual with time has been established through many studies. The work of Liebers et al. [91] although focusing on a particular application, deals with the concept of ‘temporal drift’ which refers to the variation of biometrics over time. The observation could be generalized for any biometric. Biometrics are bound to vary with time. Age, increase in skills with experience and familiarity with the system are reasons for this. As a result, the correlation between the biometrics acquired and the template data that had been captured much earlier would gradually reduce.

When keystroke dynamics are particularly considered, this could be understood intuitively. An individual’s typing habits would change as he becomes more and more

skilled in the art of using the keyboard. Accordingly, the biometrics captured at a time he was relatively unskilled may no longer be valid. At the same time, typing speed may degrade with age due to degradation of sight, hearing, sense, and strength. Short term variations may occur due to fatigue, ambient conditions, and emotional status of user.

To cope with these changes so as to avoid false rejection of genuine users, a widely used technique is the regular update of user templates.

The work of Romain Giot et al. [92] deals with the analysis of two kinds of keystroke dynamics template update procedures. They are the 'Supervised' procedures and the 'Semi-supervised' procedures.

In supervised procedures, the user is enrolled repeatedly at frequent intervals so that his latest keystroke behaviors are captured. These are then added to the user's template. The drawbacks of this procedure is the inconvenience to the users in having to undergo repeated enrollment and, the need to have a dedicated person or system to enforce the repeated enrollment and add the latest biometrics to the template after ascertaining that they are from the genuine user.

In semi-supervised procedures there is no explicit re-enrollment. Instead, the user's keystroke dynamics are captured during the static authentication stage of his routine sessions themselves. Statistical techniques are used to ascertain whether the captured biometrics belong to the genuine user. There is no requirement for a third person or system to enforce re-enrollment. The disadvantage is the possibility of an imposter's biometrics also being captured and being appended to the template, thus contaminating the template and diminishing its reliability.

In the aforesaid work, the authors have proposed a scheme to increase the reliability of semi-supervised procedures. At each authorized login, a computation is made on the captured keystroke dynamics and using a threshold, a decision is made whether or not the captured pattern should be added to the template. This greatly reduces the possibility of imposter patterns being appended to the template.

A more recent work on template updating by Paulo Pisani et al.[93] uses 'ensemble learning' to make the templates immune to aging. Each time the templates created at initial registration are used for authentication, the templates adapt to changes in the biometrics of the registered user by continuing to learn in addition to detecting. This eliminates the need for re-registration.

The work of Poh et al. [94] deals with adaptive biometrics in general with no particular reference to keystroke dynamics or any other specific biometric. It makes an important conclusion that the effective adaptability of any biometric system to variance with time largely depends on the quality of the 'initial template'. It also proposes 'co-training' as an update strategy. Co-training is a semi-supervised machine learning method. As it is beyond the scope of the current study, its explanation is omitted from this thesis. The work of Bhatt et al. [95], although being a study on biometric authentication in

general and not particularly dealing with keystroke dynamics, provides a good explanation on the subject of co-training.

With reliable template update strategies, the authentication system could be made immune to both long term and short term time variances of keystroke dynamics.

2.13 Challenges posed by the availability of Large Language Models

A recent challenge to all spheres of learner integrity management is the undue use of Large Language Models (LLM) by cheating students to display competence not actually possessed by them at assessments through the convenient access of content and formulation of responses for submission. This was revealed in the reviews of studies [24], [25], and [26] cited earlier. When online tests are particularly considered, the free and wide availability of LLM have provided cheating candidates an additional source which they can conveniently use during online tests to source information and present text based responses without being detected.

T. Susnjak and T. McIntosh [96] have examined the effect of the availability of LLM on the integrity of online examinations. They have investigated the proficiency of LLMs in answering examination questions and discovered a popular LLM to be very proficient in answering examination questions in 12 subjects. They conclude that with such LLM at the disposal of the candidate community, examinations in the absence of effective proctoring has no validity. For remote examination, they propose solutions such as remote proctoring, use of viva-voce as a primary mode of examination, and the use of enhanced multimodal examination strategies, all of which are resource intensive and costly.

The challenge posed by the availability of LLM essentially needs to be countered as part of any system aimed at ensuring learner integrity during online tests.

2.14 Ethical considerations and legal implications

The use of keystroke dynamics as a source of biometrics for authentication essentially requires the acquisition of keystroke data. The process of recording keyboard activities is called ‘keystroke logging’ or ‘keylogging’.

There are two perceptions about keystroke logging. One perception is that it is essentially a malicious activity. This view is fueled by the knowledge that information elicited from keystroke logs can be used for stealing login credentials and for other types of unlawful intrusion into protected systems. The other perception is that keystroke logging may be used for legitimate as well as malicious purposes. Research has been spurred based on both perceptions.

Research that perceives keystroke logging as a malicious activity are primarily focused on either detecting keystroke logging activity during computer usage or the proactive development of keystroke loggers to detect vulnerabilities of protected systems.

The work of P. Santoki [97] is an example of an attempt to detect keystroke loggers. The paper includes details of large scale financial losses incurred during the period from 2005 to 2009 which have been attributed to breaches caused through keystroke logging. It also provides a comprehensive description of different types of keystroke loggers and discusses defenses against keystroke logging.

The work of S. Bejo et al. [98] is an example of a proactive development of a keystroke logger to verify the capability of malware detection systems in detecting keystroke logging.

A. Bhardwaj and S. Goundar [99] quotes parental monitoring, employee productivity monitoring, writing investigations, forensic investigations, and ethical hacking as example applications where keystroke logging is used for legitimate purposes. The work of R. Martín and M. Apfelthaler [100] makes explicit use of keystroke logs for the 'offline evaluation' of language translation work.

Based on the perception that keystroke logging can be used for hostile and malicious activities, many governments have restricted the use of keystroke loggers without consent as parts of their data protection legislation. The article by C. Ritzer [101] gives details of legal proceedings in Germany pertaining to restrictions on keystroke logging. The European Union and the US Federal Government have classified covert keystroke logging as being unlawful. Both security and privacy concerns have led to these enactments.

The work of A. Wengelin and V. Johansson [102] dealing with the legitimate use of keystroke logging to investigate the process of writing by students makes an important observation pertaining to keystroke logging in that keystroke logging generates 'massive amounts of data'. This observation is relevant to the current study due to the intended use of keystroke dynamics and therefore the need to acquire them, with the accompanying problem of having to deal with large communication payloads during online tests.

2.15 Conclusions drawn from the review of related work and their relevance to management of learner integrity

From the review of all the above cited research work related to 'learner integrity', the following conclusions were drawn:

- The problem of 'academic dishonesty' is old, dating back to early 60s.
- Academic dishonesty in online education emerged in the early 90s.

- Technology has been used for both acts of academic dishonesty as well as for their curtailment.
- Use of technology to maintain learner integrity in off-classroom activity is mostly limited to the detection of asynchronous plagiarism.
- Use of technology to maintain learner integrity during online tests is mostly limited to the extension of conventional onsite proctoring to the remote sites.

From the review of all the above cited research work related to 'keystroke dynamics', the following conclusions were drawn:

- Keystroke Dynamics have been in use as a behavioral biometric for nearly 50 years.
- Keystroke Dynamics have predominantly been used for initial authentication. Its use for continuous authentication is still developing.
- Keystroke Dynamics have been effectively used as a 'soft biometric' as well.
- Out of several biometric features that belong to the sub class of Keystroke Dynamics, the two features that are pure time measurements namely, the Dwell Time and the Flight Times, remain the primary features that have been predominantly researched and put into wide use.
- Keystroke dynamics datasets are at the disposal of the research community to perform experiments. They eliminate the need for gathering biometrics solely for experimentation.
- The model where there is an initial registration of subjects, creation of templates for the subjects from biometrics gathered during the registration, and use of the templates for authentication at subsequent engagement of the subjects, is very well established and has a large application base.
- Keystroke logging generates massive amounts of data.
- In keystroke dynamics related research work where biometrics have been captured from subjects solely for experimentation, there are cases where populations as large as 200 and as small as 25 have been used.
- The imposter detection models based on keystroke dynamics mainly use statistical analysis and trained machines.
- Continuous authentication systems need to be assessed using time related metrics such as Time to First Detection (T2FD), FAWI, and FRWI in addition to the traditional metrics like FAR.
- There are ethical and legal implications arising from the use of keystroke logging.
- The emergence of LLMs have posed additional challenges in maintaining learner integrity.

It was also seen that the application of technology to curb fraudulent practices during online assessments encountered in the reviewed work cannot be regarded as true 'engineering solutions' due to the absence of systematic identification of the root of the problem and the absence of an engineering approach to solve the problem.

The studies that were reviewed indicated the following gaps in the status of technology assisted fraud prevention for the management of learner integrity.

- As far as online tests are concerned, the current application of technology is limited to the offline detection of plagiarism. This aids in the limited detection of only the fraudulent acts of lookup. It does not aid in the detection of impersonation and real time collaboration.
- Although keystroke dynamics based detections have been experimentally found to work well in scenarios involving asynchronous submissions, its effectiveness in the detection of fraudulent practices during online tests is presently not verified.
- Curbing academic dishonesty during online tests is currently limited to confining the candidates' online activities to the test by enforcing the use of 'secure browsers' and to exercising 'remote proctoring' with heavy utilization of resources.

In the review of the study of Kell et al. [12] in 2.2.2, the concept of the 'fraud diamond' was encountered where the 4 factors motivation, opportunity, rationalization, and capability were modelled by the four sides of a diamond. It was inferred that an individual would engage in an act of fraud when the diamond is closed due to the presence of all 4 factors. Conversely, the elimination or restriction of any one or more of these factors has the potential to curtail the fraudulent act. For the establishment of a system to manage learner integrity during online tests, this model could be used to identify areas in which technology could be applied to restrict these factors.

The concept of the 'Fraud Diamond' proposed by Kell et al. [23] could be adapted to the scenario of online assessments as illustrated in Figure 2.4.

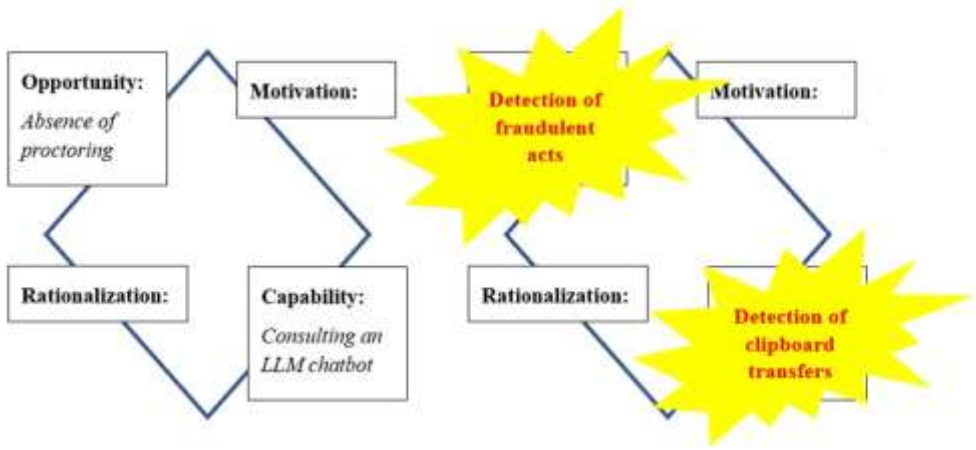


Fig. 2.4: Application of the 'Fraud Diamond' to the scenario of online assessments.

As illustrated in Figure 2.4, two sides of the diamond could be 'shattered' through the appropriate use of technology. For example, exercising surveillance through continuous authentication would limit the 'opportunity' for the cheating candidate to

engage in fraud evading detection by the institution. Clipboard transfer of text from LLM chatbots too could be detected through the appropriate use of technology, thus limiting the cheating candidate's 'capability' to engage in fraud . The factors of 'motivation' and 'rationalization' cannot be addressed through technology since they are subjective and are particular to the individual.

None of the research work reviewed above specifically address the above scenarios. To the best of the researcher's knowledge, use of keystroke dynamics for the detection of fraudulent practices during online tests has not been formally investigated earlier.

Accordingly, in the current study, the possibility of using keystroke dynamics, being the selected technology, to restrict the opportunity and the capability for cheating candidates to engage in fraud, was explored.

The current study identifies the inability of the test administration to 'detect the fraudulent practices at remote sites' as the root of the problem. In solving it, the current study recognizes that the mere extension of conventional proctoring to the remote site through the use of technology (e.g. the Automated Online Exam Proctoring System proposed by Atoum et al. [103]), is not effective nor appropriate. Accordingly, it aimed at providing the test administration surveillance over the candidates through the more efficient application of alternative technologies.

Figure 2.5 illustrates the learner integrity managed online testing system that was envisaged in the current study.

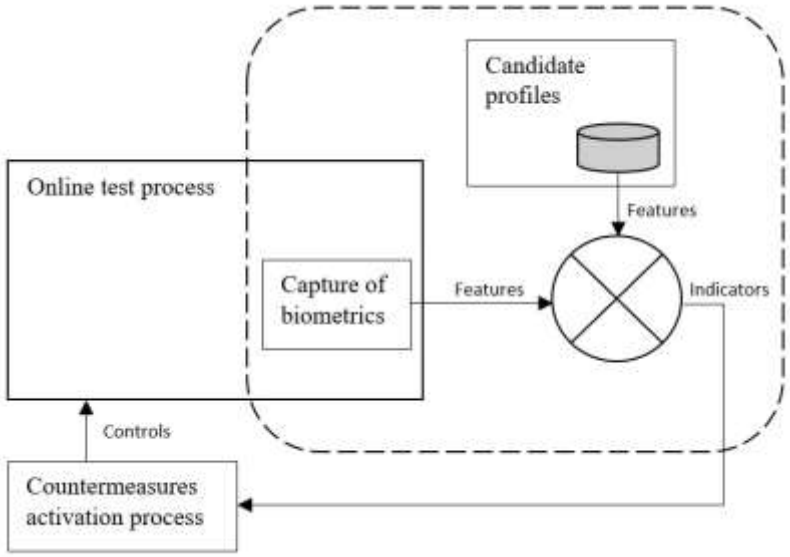


Fig. 2.5: 'Learner Integrity Managed' Online Testing System

The subsystem enclosed in dotted lines was the focus of this research. The concept that was proposed and experimentally verified was for this subsystem.

2.16 Recapitulation

This chapter described the review of research of others in the domain of 'learner integrity'. Eighteen studies conducted on 'Technology Assisted Academic Dishonesty' with particular focus on its implications on 'Online Assessments' and 'Technological Countermeasures', were reviewed. In the process, realization converged on the suitability of the behavioural biometric called 'keystroke dynamics' for authenticating candidates continuously during online tests non-obtrusively. Review of seventy six studies on behavioral biometrics in general and various aspects of keystroke dynamics in particular, was carried out. Many conclusions relevant to the current study were drawn from these reviews.

It was further realized that the proposed system for the effective management of learner integrity during online assessments could be established through the application of appropriate technology to restrict the 'opportunity' and the 'capability' of candidates to engage in fraud during online tests, which would provide an 'engineering solution' to the research problem identified in Chapter 1. A leaner integrity managed online testing system was visualized with the areas to be focused in the research clearly identified.

CHAPTER 3

THE RESEARCH METHODOLOGY

3.1 Chapter description

This chapter is devoted to the detailed presentation of the methodology employed in this research inclusive of the proposal of a concept for the establishment of a system for the effective management of learner integrity during online assessments by exercising continuous authentication of the candidates using their keystroke dynamics. The concept addresses the gaps in the knowledge domains of learner integrity and keystroke biometrics based authentication that were identified in 2.15. The concept is presented in the manner it was envisaged before conducting any experiments.

3.2 Research methodology adopted

3.2.1 Approach to the research

All methods used in this study were 'quantitative'.

3.2.2 Design of the research

A concept was presented for a system to manage learner integrity during online tests. The proof of this concept was established through results of experiments. Where necessary, conclusions were supported from the results of studies of others, with due acknowledgement. No other instruments such as surveys or case studies were used.

3.2.3 Methods used for data collection

For the experiments, a fictitious test administering institution was set up with real volunteers roleplaying as candidates. All steps of the typical workflow in a test administering institution including the conduction of mock tests and continuous authentication during the taking of online tests were tested out within this fictitious institution. Test taking by the roleplayers that is genuine as well as fraudulent, were experimented. The data collected in the process and only those were used for analysis. Some experiments that did not involve data gathering from volunteers were also conducted to investigate certain aspects of the proposed system.

3.2.4 Strategy for collecting subjects

The volunteers for roleplaying as candidates were enlisted through 'indirect convenience sampling'.

3.2.5 Methods used for the analysis of data gathered

The collected data were used for offline simulation of various scenarios. Statistical analysis was applied on the results generated to determine whether they establish proof of the concept presented.

3.2.6 Ethical and administrative considerations

One of the conditions for granting of ethical clearance for this research by the university was that the anonymity of all the subjects shall be ensured. In compliance with this condition, the researcher was not involved in the process of enlisting volunteers which was carried out through two trusted intermediaries. The researcher did not have any direct communication with the volunteers and their identities were shielded from the researcher. No information pertaining to the volunteers were gathered. The volunteers were briefed in advance on the biometrics that will be collected and were given the option of withdrawing at any time during the data collection phase.

To the best of the knowledge of the researcher, there were no laws that explicitly prohibit the capture of keystroke data in Sri Lanka at the time of this study, up to the time of writing this thesis. Accordingly, the experiments used keystroke logging with the researcher having sole access to the data acquired. The collected keystroke data are treated as property of the university.

3.2.7 Scope of the research and its limitations

The concept proposed drew on the existing knowledge on the use of keystroke dynamics as an authentication factor and focused on the peculiarities of online tests compared to other access restricted systems. It focused on restricting the ‘opportunity’ for fraud during online tests and limiting the ‘capability’ of the cheating candidates to engage in fraud, as explained in 2.15.

The system to be established through this research for the management of learner integrity during online tests is limited to online tests that require text based responses. It is not extendable to tests that require no keyboard inputs such as objective tests that gather only multiple choice responses via radio buttons and tests that require the submission of files.

Since no personal data pertaining to the volunteers were gathered to ensure their anonymity, it was not possible to investigate the application of soft biometrics in the experiments.

3.3 Distinctive challenges associated with online assessments

The concept dealt with the following additional challenges entailed in the authentication of candidates during online assessments that are not encountered in other access control applications, as discussed in the earlier chapters.

a. There is collaboration between the registered candidate with the will to cheat and the imposter.
b. User switches subsequent to login and other fraudulent acts can be detected only by a continuous authentication process.
c. The possibility exists for the use of LLM as lookup sources.

The system to manage learner integrity during online tests was intended to conform to the specifications stated in 2.11.

3.4 Suitability of Keystroke Dynamics as the biometric

In the review of related work presented in chapter 2, the suitability of 'keystroke dynamics' for the biometric authentication of candidates taking online tests was established.

The following uncertainties exist while an online test is in progress:

i. Uncertainty as to whether the registered user is being impersonated by a competent colleague either throughout or sometimes during the session.
ii. Uncertainty as to whether the registered user is being unduly aided during the session.
iii. Uncertainty as to whether the registered user is unduly tapping or looking up material during the session.

It is reasonable to hypothesize that in all the scenarios listed above, the candidate's natural typing behavior would be altered, which is likely to be reflected in the candidate's keystroke dynamics. In the case of impersonation, altered keystroke dynamics is a given. In the other two cases, even though the biometrics are of the same person, they would be altered due to the indulgence in the fraudulent act. The study of Chan [25] which used keystroke dynamics for the detection of asynchronous 'AI-giarism' was also based on the same rationale. Since it cannot be predicted as to when this would happen during the session, the behavior must be monitored continuously.

The choice of 'keystroke dynamics' as the authenticating biometric was motivated by the following:

i. The possibility for their non-obtrusive acquisition, without the user being aware of it.
ii. The possibility for their acquisition without requiring any additional hardware or accessories.
iii. The possibility to acquire and monitor them from beginning to end of the session and not only at the instance of login.
iv. Their effectiveness in static authentication already been established as seen from work of others.

3.5 Use of Lockdown Browsers

In the introductory chapter, it was mentioned that lockdown browsers have the ability to restrict fraudulent practices. By insisting that the candidate uses a lockdown browser to take the online test, an institution could curtail fraudulent practices to a considerable extent.

The functionality of a lockdown browser is described in the research review of Holden et al. [104]. The article of N. Chadaga [11] cited earlier, also gives details. A lockdown browser will permit only the transport layer connections required for the online session. Therefore, the candidate is unable to open additional browser tabs for lookup from the Internet. After gaining access to the test, the browser tab used for the test will blow up to full screen and cannot be reduced. The candidate cannot open any other windows or tabs to run other applications while logged in. Screen captures, clipboard usage and printing are temporarily disabled. All communication ports and connections other than the one supporting the session running the test are temporarily disabled.

However, a lockdown browser cannot detect or prevent impersonation. It also cannot prevent the candidate from looking up or collaborating during the test using another device, or collaborating physically with a colleague present at site. Notwithstanding these limitations, the mandatory use of a lockdown browser or a dedicated application providing a user interface having the same capabilities can tremendously enhance the institution's capability to curtail fraud during online tests.

The concept presented here acknowledges the advantages of using lockdown browsers for online tests from the point of view that no practical access control system could claim to have an FAR of absolute zero and that access control should always be supplemented with every other possible means available. However, a lockdown browser was not considered to be a part of the authentication system proposed. It has the capability to detect fraudulent acts that a lockdown browser cannot prevent.

3.6 The authentication system architecture and workflow

It was proposed that the workflow of the system conforms to the widely used authentication model comprising the following:

i. An initial registration process where the biometrics of the registered candidate are acquired.
ii. Creation of a unique profile (template) for the candidate using the acquired biometrics.
iii. Continuous authentication of the candidate at subsequent sessions by assessing the degree of correlation between the biometrics captured continuously during that session and those contained in the profile.

Figure 3.1 illustrates this workflow.

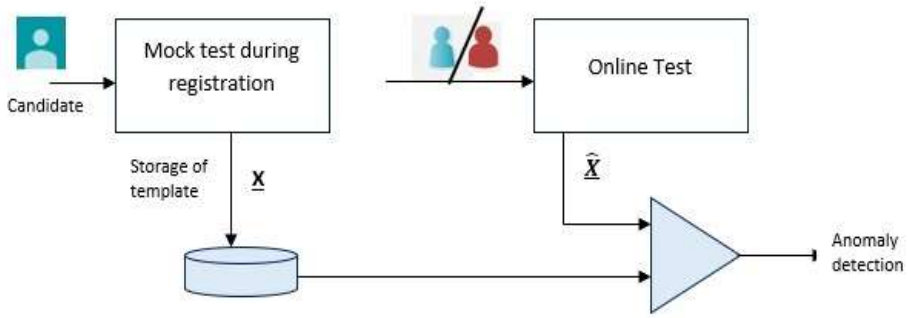


Fig. 3.1: Workflow of the proposed authentication system

As seen from the review of related work, one reason that is preventing the widespread use of keystroke dynamics based authentication is the difficulty for the online service providers in acquiring them from the client population. Prospective clients must experience minimum 'authentication friction' during registration and subsequent use. However, the need for repeated typing of a token sequence is essential to prepare usable templates for keystroke dynamics based access control. Therefore, the aforesaid workflow is generally not conducive in common access restricted online services.

On the other hand, educational and testing institutions can readily adapt to the above workflow. An institution's processes provide room for the convenient incorporation of all the steps of the above workflow. A mock test could be conducted as part of the registration process. During this mock test, provision could be made for the registrant to type both predefined character sequences as well as 'free text' of candidate's choice. The mock test could be conducted solely for the purpose of acquiring biometrics or could be combined with an initial assessment, a formative assessment, or similar assessments that are integral parts of the institution's programme. Either way, an educational institution should not have a difficulty in setting the stage for the capture of keystroke dynamics of its registrants.

The mock test essentially has to be held in a proctored environment where the registrant is physically present to verify the identity so as to ensure that the biometrics captured are actually those of the person being registered. The test need not be

conducted simultaneously for all registrants unlike an actual test. Accordingly, the conduction of a mock test should not pose any logistic problem to the institution.

The biometrics thus captured are used to extract features that are unique to the registrant and they become a part of the individual profile the institution maintains for each registrant. When conducting a subsequent test online, the biometrics of the candidate are again captured and the same features are extracted and compared with those in the profile. If there are no fraudulent acts during the test, the two sets of biometrics should correlate well. Any significant anomaly between the two would flag one, several, or any combination of the following at the candidate's end:

i. It is an imposter and not the registered candidate who is taking the test.
ii. It is the registered candidate who is taking the test and the candidate is looking up material (say, on another device or book) to aid in responding to the test.
iii. It is the registered candidate who is taking the test and the candidate is collaborating with a competent colleague in real time (say, by telecommunicating).

Figure 3.2 shows the architecture of the proposed authentication system.

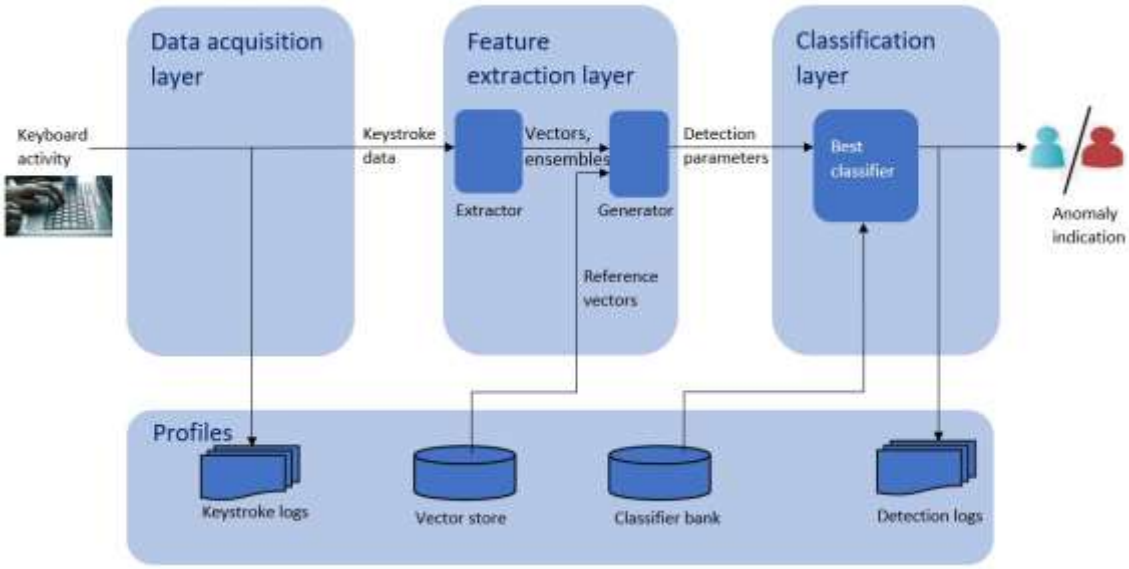


Fig. 3.2. The architecture of the proposed authentication system

As seen from the figure, it has 3 distinct layers, comprising the data acquisition layer, feature extraction layer, and the classification layer, which are functionally independent and have read/write access to the common storage system that stores the candidate profiles. The figure depicts the scenario of continuous authentication being exercised during an online test, where feature vectors and ensembles are extracted from the acquired keystroke data in real time, compared with reference vectors available in the candidate's profile, anomaly detection data are generated and fed to the 'best classifier' retrieved from the classifier bank of the candidate. The classifier

makes the classification decision in real time. Both the keystroke data acquired and the classification decision are stored for administrative purposes.

The creation of the user profile may also be understood within the same architecture.

Figure 3.3 depicts the scenario where in the first instance, the keystroke data are acquired in real time while the candidate undergoes the 'mock test' and are logged at the back end.

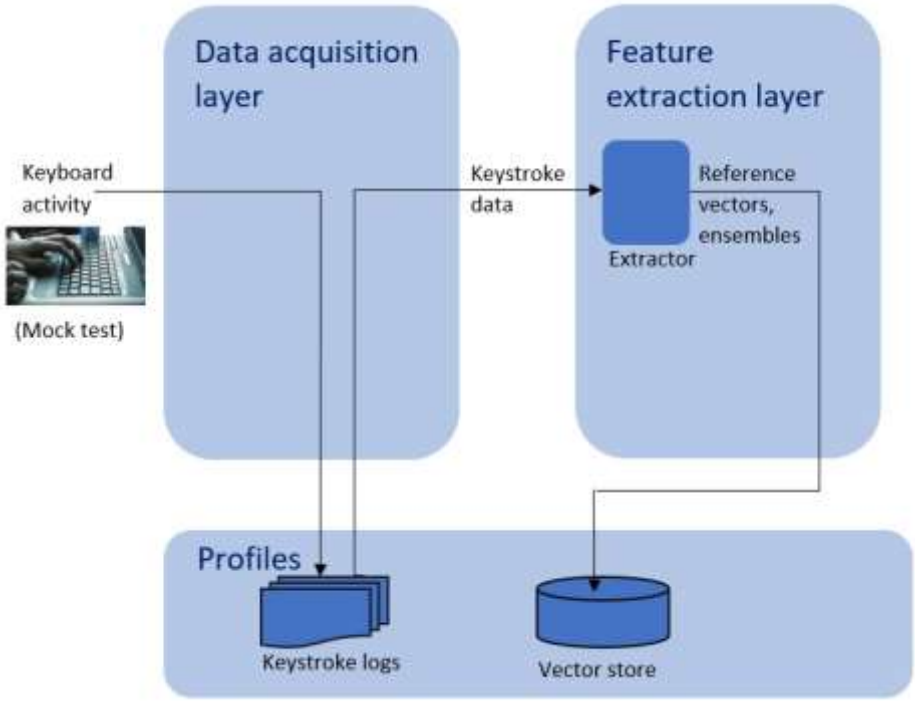


Fig. 3.3: First stage of offline profile creation – Reference vector extraction

This log is the primary source of identity for the candidate. The figure also shows how these data are retrieved, features are extracted from them, and stored. They represent the secondary source of identity for the candidate. This step which is a 'play back' of the mock test can be carried out asynchronously.

Figure 3.4 shows the next step of profile creation which too is asynchronous and involves yet another playback of the mock test.

In this step, the stream of actual detection parameters are generated using the reference vectors that are already available in the profile. In effect this simulates the taking of the mock test with continuous authentication. Although, detection parameters are generated, no authentication is possible since a classifier is not yet available. Instead, the generated detection parameters are stored for use as 'training and testing data' at a later stage.

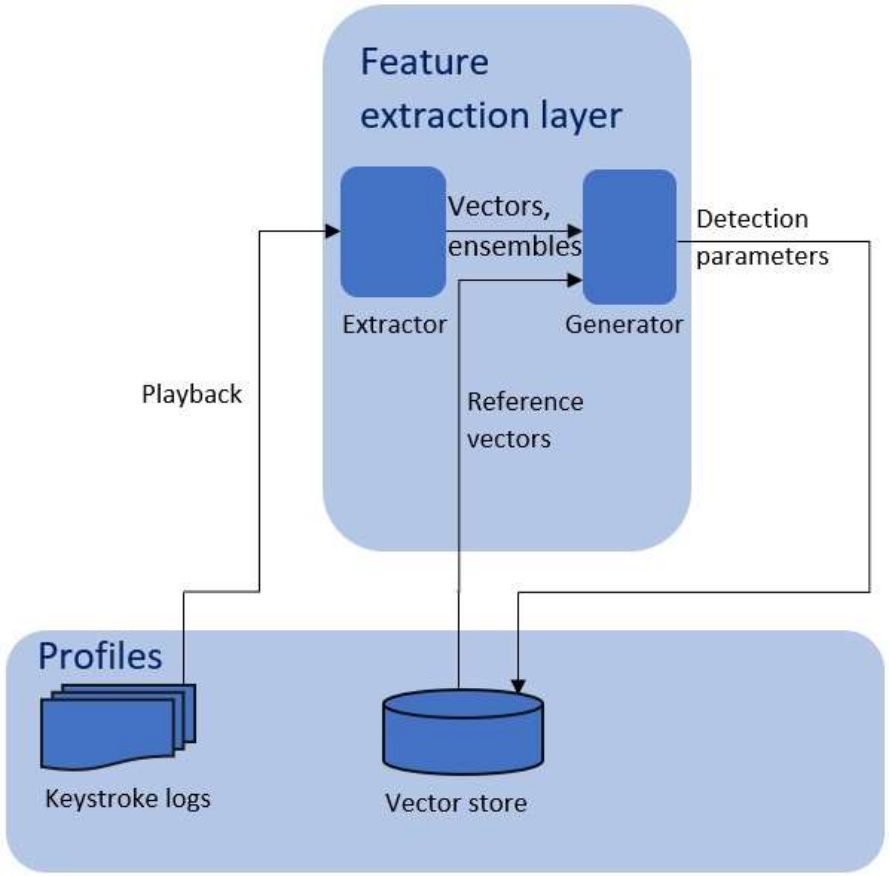


Fig. 3.4: Second stage of offline profile creation – Training data generation

Figure 3.5 shows the final step of profile creation where a classifier is trained using the training data available in the profile. The stream of anomaly indications are used in the iterative process of training the classifier.

Once the classifier is found to perform well with the testing data, it is stored in the candidate's profile for use in anomaly detection during subsequent online tests.

It will be noted that the architecture supports both profile creation and continuous authentication for the candidate. The functional independence of the 3 layers enable the offline creation of user profiles while all 3 layers come into play when continuous authentication is exercised during online tests.

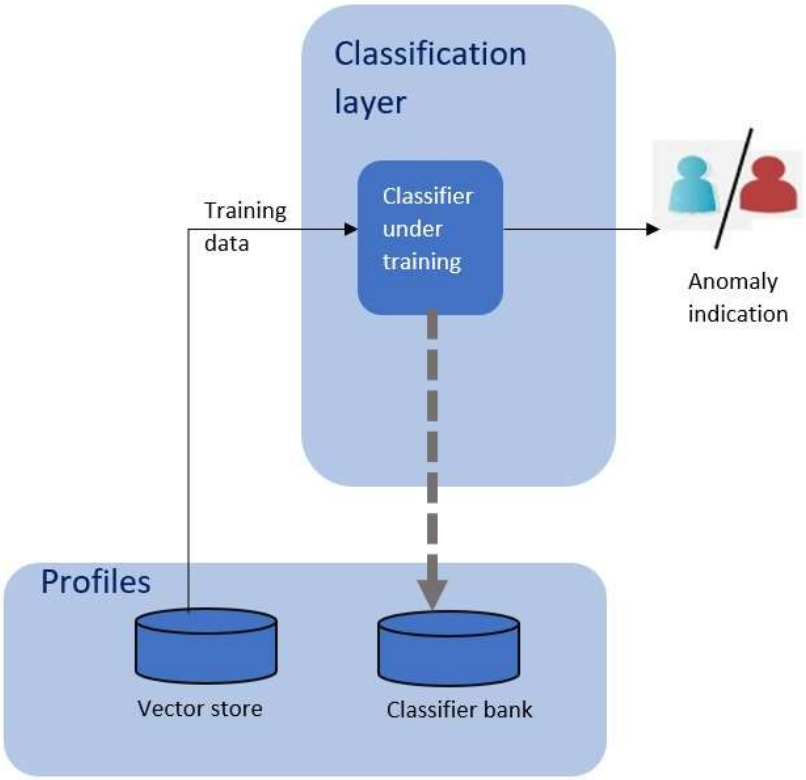


Fig. 3.5: Final stage of offline profile creation – Classifier training

3.7 Description of the features to be extracted

The definitions of 'dwell time' and 'flight times' and their computation from the key event data were presented in Chapter 2 and are reproduced here for ease of reference.

Figure 3.6. shows two consecutive key events when typing on a keyboard.

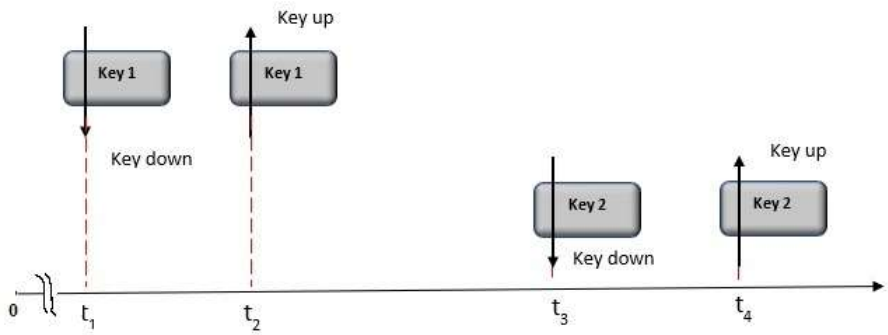


Fig. 3.6: Measurement of epochs for key events

The epochs are measured from a predefined time origin.

For key 1 and key 2 respectively, the dwell times would be

$$DT_1 = t_2 - t_1 \quad \text{..... (3.1)}$$

$$DT_2 = t_4 - t_3 \quad \text{..... (3.2)}$$

For the consecutive pressing of key 1 and key 2, we could define the flight time of type 1 as follows:

$$FT_{12}\text{type1} = t_3 - t_2 \quad \text{..... (3.3)}$$

It was proposed that the 'dwell time' and 'flight time of type 1' (which will be referred to simply as 'flight time' hereafter) be the primary biometric features to be extracted.

It was proposed that the following features be extracted from the dwell time and the flight time captured directly from the typing of the candidate during the mock test and stored.

i. Dwell time vector
ii. Hot sequence vector
iii. Quick sequence vector
iv. Far sequence vector
v. Pause durations ensemble

This extraction could be done asynchronously, for the purpose of creating candidate profiles.

3.7.1 The 'Dwell Time Vector'

The dwell time of each key stroke comprising its key up and key down events could be readily computed from the captured epochs as explained above.

Since the same key will be pressed many times during the test, a count of this number could be maintained and the 'mean dwell time' could be computed for each key. The mean dwell times of any two arbitrarily chosen keys will generally be different.

If we choose 'n_1' number of keys on the standard keyboard for observation, there will be a set of 'n_1' mean dwell times for a candidate. We could imagine an n_1-dimensional space in which these 'n_1' keys will represent the 'n_1' orthogonal directions. A candidate could be mapped to a point in this space using the candidate's n_1-tuple of mean dwell times. The mean dwell time of a particular key will be the 'abscissa' or the 'directional weight' of the point along the direction corresponding to that key.

The 'dwell time vector' will be the position vector of this point in the n_1-D space. Assuming that the tuple of dwell times is unique to a candidate, the 'Dwell Time Vector', too would be unique to that candidate.

3.7.2 The 'Hot Sequence Vector'

Hot sequences are sequences of multiple keys that have a high probability of occurrence in the text of a given language. For example, in English text, the sequence

'ing' has a high rate of occurrence and therefore is considered a hot sequence of 3 keys. Similarly, the sequences 'in', 'ng', and 'th' are hot sequences of 2 keys.

To keep the computational overheads in computing flight time to a minimum, only 2-letter sequences in the English language are proposed to be considered as hot sequences.

Consider a chosen set of *'n_2'* hot sequences of 2 letters. From the key events captured during the mock test, consecutive key events comprising the selected n_2 hot sequences could be filtered out and their flight times could be computed from the epochs of each key event. Since each hot sequence is likely to be typed more than once during the mock test, a count of such occurrences could be maintained and the 'mean flight time' of each sequence could be computed. This would yield an n_2-tuple of mean flight times of hot sequences.

Similar to the case of the dwell time vector, this n_2-tuple could be treated as being unique to the candidate. Then, through similar reasoning as done for the dwell time vector, the candidate could be mapped to a unique point in an n_2-D space defined by the n_2 hot sequences representing n_2 orthogonal directions. The 'Hot Sequence Vector' will be the n_2-D position vector of this point which too would be unique to the candidate.

3.7.3 The 'Quick Sequence Vector'

Quick sequences are 2-letter sequences that a candidate would type quickly or faster than the average speed. They are characterized by the flight time of type 1 being negative, which means the second key is pressed before the first key is released.

Unlike hot sequences, quick sequences cannot be predetermined. Each candidate may have a set of quick sequences particular to that candidate. It is even possible for a candidate to have no quick sequences.

Let the size of the set of quick sequences of a given candidate be '*n_3*'. Then, through a reasoning similar to that of the dwell time vector and the hot sequence vector, an n_3-D position vector could be defined in an n_3-D space. This would be the candidate's 'Quick Sequence Vector' and the candidate will be mapped to the point fixed by that vector.

3.7.4 The 'Far Sequence Vector'

Far sequences are two letter sequences typed using extreme fingers. For a touch typist, there is an accepted finger usage within the typing trade for typing each key on the keyboard. This usage is a quasi-standard and most people who touch type are accustomed to this finger usage. This usage is derived by positioning both hands on the keyboard and allocating keys to each finger. See Figure 3.7.

When typing a 2-key combination of a far sequence, one letter is typed using the little finger and the other letter is typed using the index finger of the same hand. As seen from Figure 3.7, each index finger is allocated 6 keys while each little finger is allocated 3 keys. Therefore, when the reverse sequences are also counted, 36 far sequences can be typed using each hand. This gives a total of 72 far sequences. These sequences will have different frequencies of occurrence in a given English text.



Fig. 3.7: Allocation of keys to each finger in each hand, for touch typing. (Source: www.learning.com)

The flight times of far sequences are likely to have a high degree of uniqueness to the person typing because the fingers are stretched to the maximum and how soon the muscles of the two fingers would respond would depend on the person to a great extent.

Similar to the case of Hot Sequences and Quick Sequences, the 'Far Sequence Vector' unique to the candidate, having n_4 dimensions could be defined where $n_4 = 72$.

The algorithm given in Figure 3.8 was proposed for the extraction of the 'dwell time vector'.

This algorithm could be generalized for the other 3 vectors as well.

3.7.5 The 'Pause Duration Ensemble'

When typing on a keyboard, people generally tend to pause between words, sentences and more specifically, after typing a white space. This hesitation is important to comprehend and reflect upon what is being typed. As described by M. Torrance and D. Galbraith [105], the human mind processes the text being typed one word at a time and this leads to the typist pausing briefly at every whitespace.

```
while there are more directional weights to be populated in the dwell_time_vector
    point to the first entry in the container
    initialize mean_dwell_time, dwell_time_total and key_occurrence_count
    while a 'key down' event of the key is not encountered OR end of container is not
      reached
        parse the container in search of a 'key down' event of that key
        if found
            increment key_occurrence _count
            save the timestamp of the event
            save the pointer of the container
         while a 'key up' event of that key is not encountered OR end of container is not
           reached
                parse the container in search of the 'key up' event of that key
                if found
                    retrieve the time stamp of the event
                    compute timestamp_difference
                else
                    time_stamp_difference = 0
                    decrement key_occurrence _count
        dwell_time_total ← dwell_time_total + timestamp_difference
        return to saved location in the container
        increment pointer in the container to point to next event
    mean_dwell_time ← dwell_time_total / key_occurrence _count
    push mean_dwell_time to the directional weight of that key in the vector
    take the next of the n1 keys in the 'dwell time vector'
```

Fig. 3.8: Algorithm to extract 'Dwell Time Vector'

If a stream of pause durations captured during a single typing session is considered, the values could be expected to have a high variance. However, if the values from the typing of two persons are scatter plotted, a considerably distinct clustering for each person is likely, as illustrated hypothetically in Figure 3.9.

In practice, the difference in the clustering may not be as conspicuous as shown in Figure 3.9 to the human eye, but it could be machine detectable.

In the stream of key events captured during the mock test, pauses could be identified by searching for white spaces. The pause duration will be the differences between the epochs of the key up event that immediately precedes the white space and the key down event immediately following the white space.

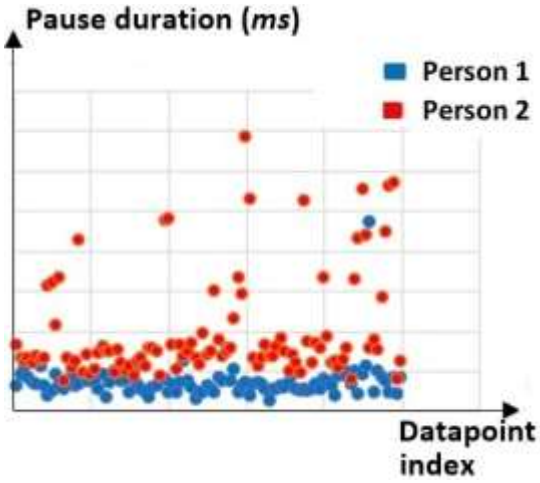


Fig. 3.9: Distinct clustering of pause durations of two persons (hypothetical, not derived from real data)

From the entire duration of the session, an ensemble of such pause durations could be extracted for a given candidate. The 'Pause Duration Ensemble' of that candidate would be the values in this ensemble arranged as a 1-dimensional array. Unlike in the cases of the other 4 vectors, there is no visualization of a multidimensional space or the mapping of the candidate to a unique point. The Pause Duration Ensemble is simply a 1-D array of scalars. Even its size will be different for different candidates.

```
Initialize array to hold the 'pause duration ensemble'
point to the first element of the array
open the container of key event data of the candidate, captured during the mock test
while a 'key down' event of a white_space_ key is not encountered OR end of container is not reached
        parse the container in search of a white_space_ key
        if found
                save the pointer of the container
                roll back pointer to the previous 'key up' event
                save the timestamp of the event
                return to saved location in container

                while a 'key down' event is not encountered OR end of container is not reached
                                parse ahead in the container in search of the next  'key down' event
                                if  found
                                        retrieve the time stamp of the event
                                        compute timestamp_difference
                                        push timestamp_difference into the array
                                        point to next element in the array
                return to saved location in the container
```

Fig. 3.10: Algorithm to extract 'Pause Duration Ensemble'

However, it is assumed that it will have a reasonably unique clustering for a given candidate.

The algorithm given in Figure 3.10 was proposed for the extraction of the pause duration ensemble.

3.7.6 Extraction of non-temporal features

In the review of related work in Chapter 2, few instances where non-temporal features were combined with keystroke dynamics to enhance the capability of the authentication system were encountered. Non-temporal features are not directly connected with time measurements of individual keys.

The following two non-temporal features were proposed to be extracted.

i. Typing speed
ii. Error rate

Two variants of typing speed could be defined as the 'raw typing speed' and the 'effective typing speed'. The raw typing speed takes every single character that is typed into account including those typed in error and those typed to make the correction. The effective typing speed on the other hand disregards the keystrokes that are concerned with correcting errors, such as the pressing of the DEL key, the BACKSPACE key, and the arrow keys used to navigate the cursor to the location of the error.

There is no formal definition for the error rate but a convenient estimate could be made as follows:

$$\text{Error rate} = \frac{Number\ of\ rubouts}{Total\ number\ of\ characters\ typed} \quad \text{..... (3.4)}$$

The rubouts are the pressing of the DEL key and the BACKSPACE key.

It is not necessary to capture additional biometrics to extract the above two non-temporal features and they are readily extractable from the keystroke data captured for the extraction of time related keystroke dynamics.

Therefore, the use of the above two non-temporal features to supplement the authentication performed using dwell time and flight time was proposed.

3.8 Creation of the Candidate's Profile

The extraction of 5 types of features from the biometrics captured while each candidate took the mock test, was outlined above. These features which are assumed to be unique to each candidate, constitute the unique profile for each candidate. The features are stored under the unique identity of each candidate. This facilitates the convenient retrieval of the features when it is required to authenticate a candidate taking a subsequent test.

3.9 Continuous Authentication during subsequent tests

When a registered candidate takes a subsequent test online, continuous authentication could be exercised using the 4 multidimensional vectors stored in the candidate's profile, in the following manner:

The keystroke dynamics of the candidate are again captured during this test and the 4 vectors are recomputed. These vectors and the vectors in the profile are compared to detect anomalies. Any anomalies between them would indicate possible fraudulent acts described in section 1.2 to be in progress.

When computing the vectors for the profile, the stream of key events captured during the entire duration of the mock test was considered. In contrast, during a subsequent online test, continuous authentication of the candidate is required. This could be achieved by computing the vectors regularly at the end of a fixed number of key events, say **M** key events.

At the end of every sequence of **M** key events, an indication could be obtained whether a fraudulent act is being committed or not by comparing the orientation of each of the 4 vectors with the corresponding vectors in the profile. In this manner there would be a stream of indications throughout the test, once for every **M** key events. Due to the statistical nature of biometrics, there is no guarantee that anomalies if any, will be detected at the end of every sequence of **M** key events. Depending on whether the test is being taken genuinely or with fraudulent acts, the detections could be true acceptances, true rejections, false acceptances, or false rejections, as in the case of any biometrics based authentication.

3.10 Method of anomaly detection

The anomaly detection method proposed in the concept could be understood by considering the use of one of the vectors. Consider the dwell time vector.

Let the dwell time vector in the candidate's profile be 'μ' and let the dwell time vector computed at the end of a certain sequence of M key events during the subsequent test be '$\widehat{\mu_s}$',

Now, 'μ' has been computed considering the mean dwell times of n_1 pre-identified keys and it is truly an n_1-dimensional vector. In contrast, '$\widehat{\mu_s}$' has been computed from a sequence of **M** key events. The keys contained in the **M** key events would depend on what the candidate has typed. Further, the **M** key events would generally contain events due to the repeated pressing of certain keys. Due to these two reasons, it is unlikely that these **M** key events would contain all n_1 keys. Therefore, the vector '$\widehat{\mu_s}$' would generally have a lesser number of dimensions.

Let the number of 'unique keys' found within the sequence of **M** key events be m. Then, in general,

$$m \leq n_1$$

The m-tuple of mean dwell times of these m keys will define an m-D subspace within the n_1-D dwell time vector space. For each sequence of **M** key events there will be an associated m-D subspace, depending on what was typed. For this sequence of **M** key events, '$\widehat{\mu_s}$' would be the position vector of a unique point within the m-D subspace.

Since the number of dimensions of the two vectors μ and $\widehat{\mu_s}$ are different and because the spatial orientation of μ is influenced by directional weights not relevant to $\widehat{\mu_s}$, it would be irrational to make any orientation comparisons between the two. Therefore, it is necessary to first establish a common basis for comparison. One way of achieving this would be the projection of the n_1-D dwell time vector 'μ' onto the m-D subspace of $\widehat{\mu_s}$.

Let this projection be μ_s. Now, both μ_s and $\widehat{\mu_s}$ are m-D vectors within the same subspace. Recalling that the directional weights of μ_s are the mean dwell times exhibited by the candidate during the mock test and that the directional weights of $\widehat{\mu_s}$ are the mean dwell times exhibited by the same candidate while typing the current sequence of **M** key events, the two vectors should ideally coincide. However, in practice $\widehat{\mu_s}$ is only a runtime estimate of μ_s and there will be an angle θ between them. We define this angle θ as the 'angle of departure'.

Figure 3.11 illustrates the formation of θ using the easily comprehended 3-D/2-D analogy for the special case where n_1=3 and m = 2 which could be generalized for larger values of n_1 and m

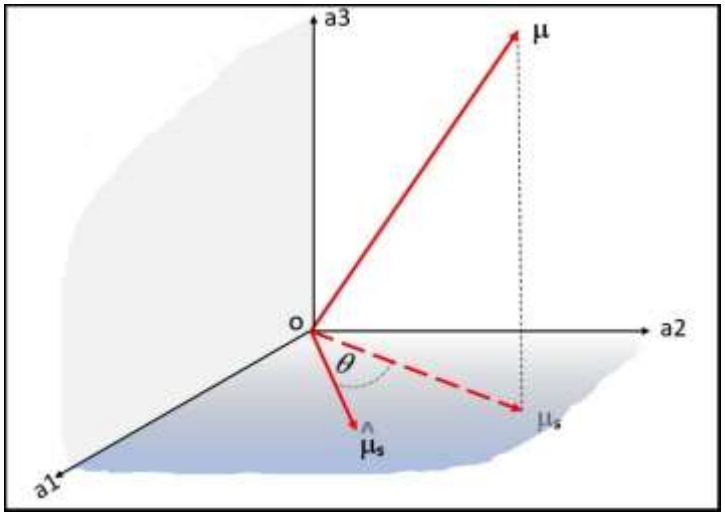


Fig. 3.11 : 3D/2D analogy for the formation of the vector pair μ_s and $\widehat{\mu_s}$ and the angle of departure θ.

θ can be computed considering the scalar product of μ_s and $\widehat{\mu_s}$ as follows:

$$\theta = \cos^{-1} \frac{\mu_S . \widehat{\mu_S}}{|\mu_S||\widehat{\mu}_S|} \qquad \ldots\ldots\ldots (3.5)$$

As the candidate continues to type, a pair of vectors μ_s and $\widehat{\mu_s}$ will be formed for every completed sequence of **M** key events, resulting in a corresponding value for θ. Hence a stream of values for θ will be generated throughout the test. Figure 3.12 illustrates this scenario.

following are noteworthy about this stream of θ values.

i. The duration of the sequence of **M** key events will generally be different from one sequence to the next.
ii. The *m*-D imaginary space will generally be different from one sequence to the next, including the number *m* and the dimensions counted by it.
iii. The vector pair μ_s and $\widehat{\mu_s}$ will have different orientations within the *m*-D space from one sequence to the next.

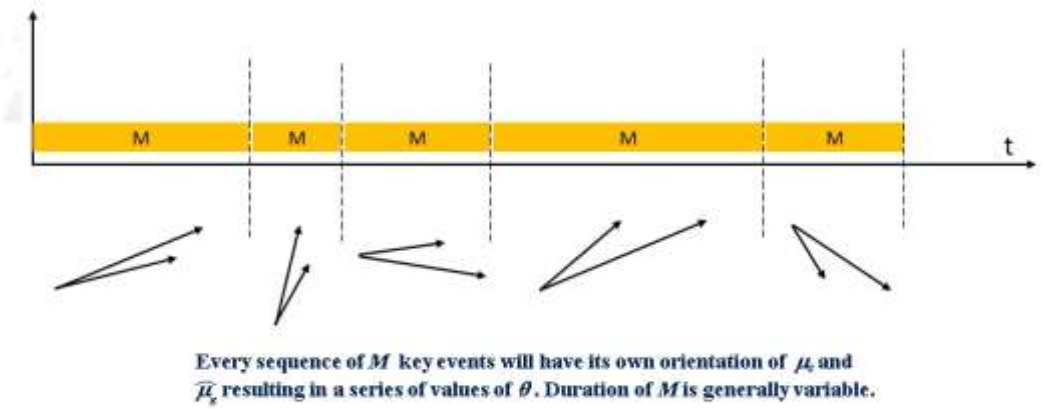


Fig. 3.12 : Generation of a stream of θ values while the candidate types.

By virtue of both μ_s and $\widehat{\mu_s}$ being derived from the biometrics of the same person, the stream of θ values would exhibit a recognizable clustering for a given candidate provided that there is no fraudulent act during the test. In contrast, if there is impersonation or other fraudulent act taking place, there is likely to be a significant deviation between μ_s and $\widehat{\mu_s}$ resulting in a different clustering because in such cases, $\widehat{\mu_s}$ is either derived from biometrics of a different person or from the biometrics of the candidate which have been altered in the process of engaging in a fraudulent act.

The difference in the clustering of the angle of departure forms the basis for anomaly detection during the online test.

A similar explanation could be provided for the other 3 vectors. Accordingly, there will be 4 independent streams of angles of departure during the test. Anomaly detection could be done independently on each stream.

Choice of **M**, the number of key events in a sequence is critical. If **M** is too large, the time between two monitoring instances will be longer and would affect the continuity of authentication. If **M** is too small, the sequence would not contain sufficient key events to estimate a vector $\widehat{\mu_s}$ that is adequately representative of the candidate. Hence, the choice should be based on a tradeoff between the frequency of monitoring and the realistic computation of the vector.

A binary classifier could be used to perform the anomaly detection at the end of each sequence of **M** key events. The classifier would be fed with a stream of scalar values of θ and it would classify the test as being 'genuine' or being 'fraudulent' at the end of each sequence of **M** key events, based on its learning. Figure 3.13 illustrates this detection scheme.

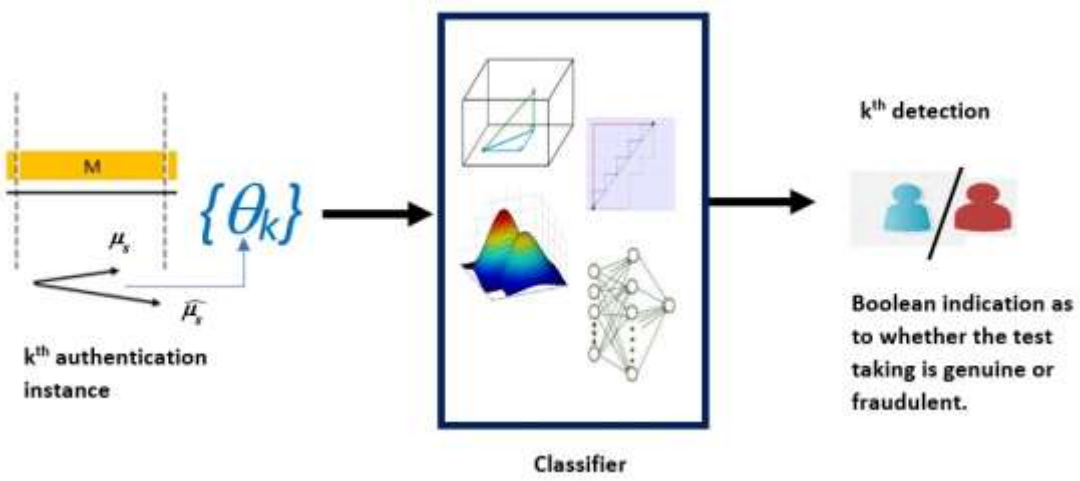


Fig. 3.13: Anomaly detection using the angle of departure

Figure 3.13 also indicates that there are many options for the implementation of the classifier. The same type of classifier may not work well for all features or for all candidates. Therefore, it is necessary to try out different models of classifiers and select the feature specific and candidate specific 'best classifiers' and include them in the candidate's profile.

It is pertinent to justify the choice of the novel feature 'angle of departure' over other features commonly used in anomaly detection.

The parameters commonly used to determine the spatial separation between two points in a multidimensional space are the 'Euclidean distance', 'Manhattan distance', and 'Mahalanobis distance'. Of these, the most common parameter is the 'Euclidian distance'. In the review of related works several studies that used this parameter were encountered.

Instead of computing the Euclidean distance between two points (or the position vector tips) in an N-dimensional space, the angle between the position vectors of the two points is computed.

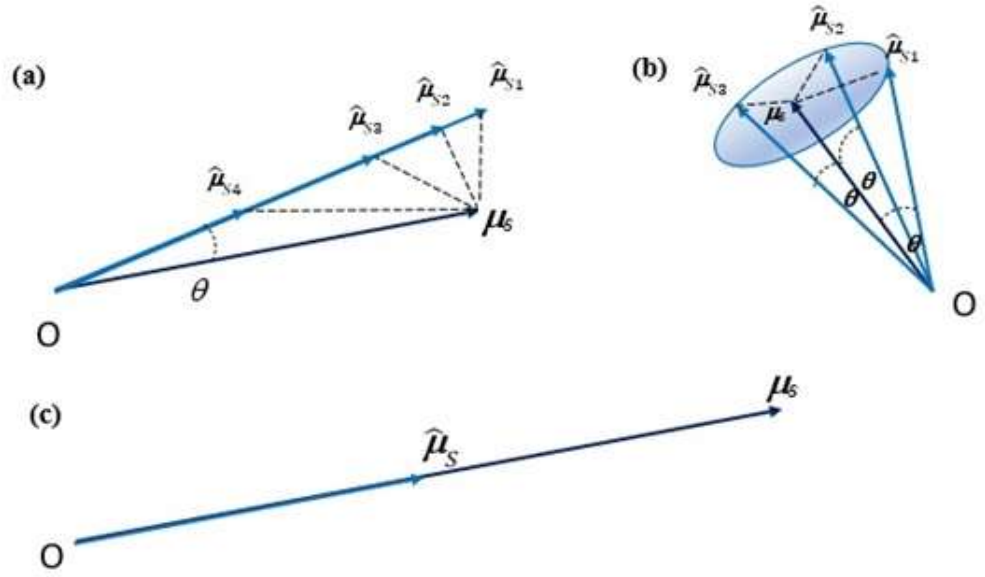


Fig. 3.14: Different scenarios of occurence for the 'Angle of Departure'

Figures 3.14(a) and 3.14(b) illustrate how the angle of departure generalizes several different spatial orientations of the two vectors into one.

Figure 3.14(a) illustrates the extreme case where several different cases of Euclidian distances yield the same angle of departure. Figure 3.14(b) illustrates the other extreme where the Euclidian distance is also constant in addition to the angle of departure but the spatial orientations of the vector $\hat{\mu}_S$ estimated at runtime are different. What is significant in both cases is that there is a single value for the angle of departure θ.

This has an important consequence on the scattering of these values. If the Euclidian distance was used as the authentication parameter in each of these cases, each would have yielded a distinct data point spread across a range of values. When the angle of departure is used instead, they all converge onto a single data point. Considering the two extreme cases illustrated in 3.14(a) and 3.14(b), it could be intuitively concluded that the parameter 'angle of departure' is in effect, a transformation of the unbounded parameter 'Euclidian distance' onto a more compressed range of values, mathematically expressed as follows:

$$\|d\| \in [0,\infty) \quad \rightarrow \quad \theta \in [0^0, 360^0)$$

This transformation has a profound effect on the learning of a machine when these parameters are used as training data. Since the requirement is for a machine to learn the clustering of the authentication parameter that is unique to a particular candidate, a parameter spread across a smaller range of values would provide more sound learning. It also has a lesser probability of occurrence of outliers.

The only instance that the angle of departure would not detect an anomaly would be when μ_s and $\hat{\mu}_s$ of generally unequal magnitudes coincide, making the angle of departure equal to zero. This is illustrated in Figure 3.14(c). This can happen if and only if the mean dwell times of each and every key within the **M** key events at hand are an equal multiple of the mean dwell time of the corresponding key in the profile, making $\hat{\mu}_s$ a scaled version of μ_s. The diminutiveness of the probability of occurrence of this eventuality could be deduced mathematically as follows.

For the two vectors $\boldsymbol{\mu_s}$ and $\hat{\boldsymbol{\mu}}_s$ to coincide in direction so as to make the angle of departure θ between them equal to zero as illustrated in Figure 3.14(c), a positive scalar constant '$\boldsymbol{k}$' should exist such that

$$\hat{\boldsymbol{\mu}}_s = \boldsymbol{k\mu_s} \quad \text{where } k > 0 \qquad \ldots\ldots (3.6)$$

For this,

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{i}} = \boldsymbol{k\mu_i}, \quad \forall \ \boldsymbol{i \in [1,N]} \qquad \ldots\ldots (3.7)$$

where '$\boldsymbol{i}$' represents one of the $\boldsymbol{N}$ dimensions (and one of the keys on the keyboard for the specific case of 'dwell time vector' being considered).

Therefore

$$P(\theta=0) = P(\hat{\boldsymbol{\mu}}_s = \boldsymbol{k\mu_s}).$$

$$= \prod_1^N P(\hat{\boldsymbol{\mu}}_{\boldsymbol{i}} = \boldsymbol{k\mu_i}) \qquad \ldots\ldots (3.8)$$

Since the keystrokes of individual keys are independent, each probability term for different values of 'i' would be very small and the probability product term in equation (3.8) above would tend to zero.

Also, higher the number of keys contained within the $\boldsymbol{M}$ key events, larger would be the number of terms in the product, further reducing the probability of the two vectors coinciding in direction. Hence, it may be concluded that the probability of the angle of departure being zero and an anomaly being missed as a consequence, is negligibly small.

Therefore the 'angle of departure' is more suited than the 'Euclidian distance' for use as an authentication parameter. It could be extracted for all 4 vectors described from 3.7.1 to 3.7.4 above.

Anomaly detection using the remaining feature, the 'Pause Duration Ensemble' has one difference. Unlike in the case of the first 4 vectors where there would be a single value of θ at the end of every sequence of **M** key events, in this case there would be an ensemble of pause durations. To have a single value, the mean of the ensemble could be taken.

Let the mean value of the ensemble of pause durations for a given sequence of **M** key events be $\boldsymbol{p}$. Accordingly, the continuous typing by the candidate will generate a stream of mean pause durations $\boldsymbol{p}$ similar to the stream of θ values in the vector based classification. The difference is that the generation of this stream does not require any parameter from the candidate's profile.

As in the case of classifiers that detect anomalies on θ, a classifier could be used for anomaly detection on $\boldsymbol{p}$ as well, as illustrated in Figure 3.15.

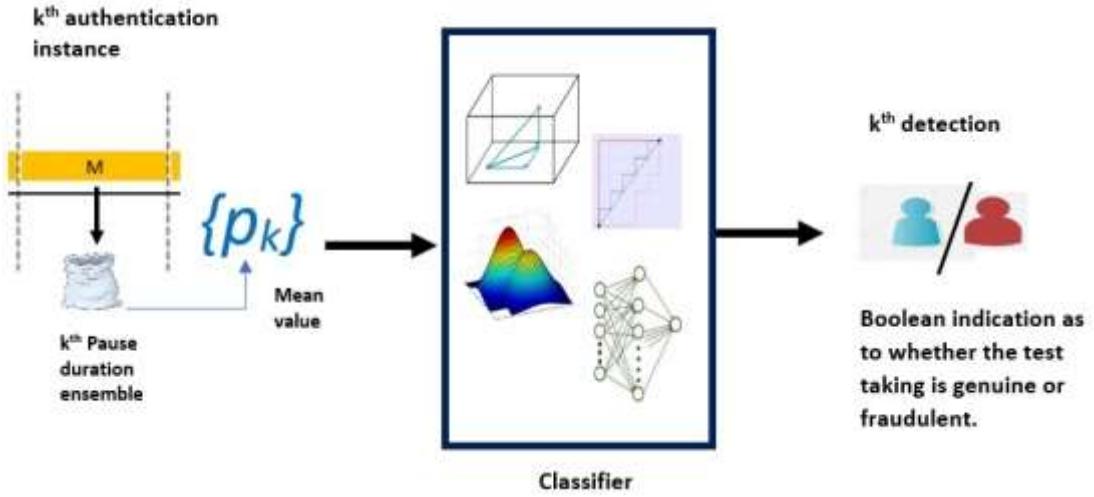


Fig. 3.15: Anomaly detection using the mean value of pause durations

In this case too, there are many options for the classifier and the candidate specific 'best classifier' should be selected and included in the candidate's profile.

3.11 Training the classifiers

For the anomaly detection schemes described above to be workable, the classifiers need to be trained beforehand. This will require training data.

It was proposed that supervised learning be used. Therefore, a labelled dataset will be required for each candidate for each feature.

In the work of Y. Deng and Y. Zhong [106] that deals with one time authentication using keystroke dynamics, the use of profile data itself for the generation of labeled training data for the training of classifiers is described. The authorized user's keystroke data will be the source of 'genuine' data while the keystroke data from the profile of any other authorized user would be a convenient source of 'anomalous' data. It was proposed that this approach be adopted. Since the profile of each candidate stores the keystroke data that were acquired during the mock test, training data could be generated through an offline 'simulated taking of the mock test' with continuous authentication being applied.

Figure 3.16 suggests an algorithm for creating training data for a binary classifier intended for the 'dwell time related features'.

Making candidate_1 and candidate_2 the same will cause the simulation of an instance of 'genuine test taking' by the candidate concerned and the resulting steam of theta values should be labelled as 'genuine'. When candidate_2 is any other candidate, the simulation will be an instance of 'fraudulent test taking' where candidate_2 poses as candidate_1. Accordingly, the resulting stream of theta values should be labelled as 'fraudulent'. In this manner, a balanced dataset with datapoints having both labels could be generated for dwell time related features.

Initialize array to hold training data points

Retrieve μ of *candidate_1*

Open the container of key event data of *candidate_2*, captured during the mock test

Point to first key event in the container

While more key events are available in the container

Retrieve *M* key events

Compute $\widehat{\mu_s}$ for the keys contained in the *M* events

Determine subspace of $\widehat{\mu_s}$

$\mu_s \leftarrow \mu$ projected to subspace of $\widehat{\mu_s}$

$\theta \leftarrow$ angle between μ_s and $\widehat{\mu_s}$

Push θ into array

Point to next element of array

Increment pointer of the container by *M*

Save array of training data points in the profile of *candidate_1*

Fig. 3.16: Algorithm for creating training data for binary classifier for dwell time related features

The method could be generalized for the other 3 vectors. For the simulation of impersonation, it is only required that the biometrics should come from a different person. The institution would have profiles of a large number of candidates and a group of such candidates could be selected for the simulated impersonation of each candidate. It should be noted that the mock test is common to all candidates which enables the simulation of impersonation in the manner described above.

The criterion for selecting other candidates as sources of anomalous data is important for the creation of a good training dataset. Rather than selecting candidates at random, they should be selected in a manner that the classifier could best learn the differences in the clustering.

The balanced dataset generated in this manner for each feature could be used to train a binary classifier of a suitable type which could be used for anomaly detection at subsequent tests.

Training the classifier for pause durations would be slightly different, In this case, there is no need to have a simulated run of the mock test to generate training data. A suitable subset of the ensemble of pause durations itself of the candidate under consideration would serve as the 'genuine' dataset while the ensemble of any other candidate would serve as a source of 'fraudulent' data.

3.12 Conformity to the specifications

As established in 2.11, the authentication system for online tests should conform to the following specifications:

a. It should authenticate the candidate continuously during the test
b. It should not require or depend on the cooperation of the user and should work in the background, invisible to the user.
c. It should not require the deployment of additional hardware or accessories at the user site and should not exert demands on resources such as connection bandwidth.

The conformity of the proposed system to the above specifications could be established as follows:

i. An anomaly detection test is applied to the data acquired during the test at the end of every **M** key events. When **M** is chosen such that, it is small enough to provide sufficiently frequent authentications and is large enough to contain sufficient key events to extract features that are truly representative of the person taking the test, continuous authentication will be realized.
ii. Since only the keystroke data that enables the computation of dwell time and flight time are captured, it can be done without the knowledge of the candidate. The candidate will not know that his/her keystrokes are being logged in the background. Therefore, the necessity to seek or depend on the candidate's cooperation does not arise.
iii. Since only the time related keystroke data are being captured, no additional sensors are required at the candidate's site. A keyboard will be used in any case to respond to the test. No other hardware or accessories will be required. The keystroke data would comprise a payload much smaller than what would be

required if remote proctoring was used and therefore the resources required for transmission would be significantly less.

Table 3.1 gives a comparison of the proposed system with other commonly used biometrics based authentication systems.

Table 3.1: Comparison of proposed system with other commonly used biometric authentication systems

	Commonly used systems	**Proposed system**
Features - Primary - Derived	Diverse, including sensor acquired features Distance measurements (e.g. Euclidian, Manhattan)	Time related vectors and ensembles Angle of departure, pause duration
Models	SVM, Random Forest, KNN, CNN, LSTM	LR, Random Forest, SVM, Naïve Bayes
Detection optimization	Equal Error Rate (EER)	Minimum FRR
Size of datasets	Comparatively large	Comparatively small
Mode of learning	Both supervised and unsupervised	Supervised
Nature of authentication	Mostly static (at time of initial access)	Continuous
Template update	Possible	Possible
Enrolment friction	High (since repeated capture of biometrics must be artificially facilitated, often inconveniencing the registrant)	Low because, within the domain of online tests, the possibility of conducting a mock test gives a unique opportunity for convenient initial data acquisition.
Deployment	Generally requires exclusive deployment of hardware, software	Possibility exists for deployment as a plugin to existing LMS

3.13 Legal implications and ethical considerations

Capturing keystroke dynamics requires the use of a keystroke logger, that is, a background application that identifies each key typed and stores it along with other related data. As mentioned in Chapter 2, in certain countries, the use of keystroke loggers is illegal or restricted. Implementing systems based on the proposed concept in such countries would entail legal and administrative initiatives for exemption.

Since the candidates are unaware that profiles of them are being created using the mock test and because the profile could lead to the identity of the candidate, it could be argued that such profile creation is unethical. To get around this issue, the institution

should adopt all required measures to ensure the following and inform the candidates during registration and before they take the mock test:

- That biometrics will be acquired during the test and that they will be used solely for profile creation.
- That, adequate measures are in place for ensuring the integrity of the biometric logs such as encryption and exclusive access for the institution.
- That, adequate measures are in place for the prevention of spoofing or tampering with the acquired biometrics, by 3rd parties.
- That the candidates' profiles will not be used for any other purpose within or outside the institution and beyond the candidate's period of affiliation with the institution.

These should be made an integral part of the registration process to which the candidate could 'opt in'.

3.14 Recapitulation

This chapter outlined the methodology to be used in this research in pursuit of the establishment of a system for the effective management of learner integrity during online assessments using the keystroke dynamics of candidates for continuous authentication. The research methodology was outlined in detail. A concept was proposed on which the system could be based.

Five different features were defined for use in anomaly detection. All five features are readily extractable from the keystroke data acquired during a mock test. It was proposed that binary classifiers be created for each feature. To train the classifiers, the use of features extracted from biometrics captured during the mock test itself was proposed. Algorithms that could be used to implement methods for each of the above were also suggested.

The legal implications and ethical considerations connected with the adoption of the proposed concept were discussed.

The next chapter will cover details of the experiments that were conducted.

CHAPTER 4

THE EXPERIMENTS

4.1 Chapter description

This chapter provides a detailed description of all the experiments that were conducted in this research. The objective of the experiments was to provide proof of the concept proposed in Chapter 3.

4.2 The experimental protocol

The experiments comprised 4 phases:

a. Data acquisition phase
b. Profile creation phase
c. Detection model development phase
d. Analysis phase

The principal activity during the data acquisition phase was the capture of biometrics from subjects for the purpose of experimentation. During the profile creation phase, the biometrics captured from the subjects were processed and unique profiles were created for each of them. This was followed by the detection model development phase where different learning machines were trained for the detection of anomalies during online tests using datasets extracted from the captured biometrics. The analysis phase comprised the simulation of various scenarios using the models and collecting the results for later interpretation. The analysis phase also involved additional experiments that did not involve the subjects. These were concerned with the nature of the keystroke data exchange and ways of inhibiting lookup from other sources during online tests.

Only the data acquisition phase had a distinct timeline. The other 3 phases were carried out with some degree of overlap with each other in order to accelerate the completion of the experiments. This is depicted in Figure 4.1

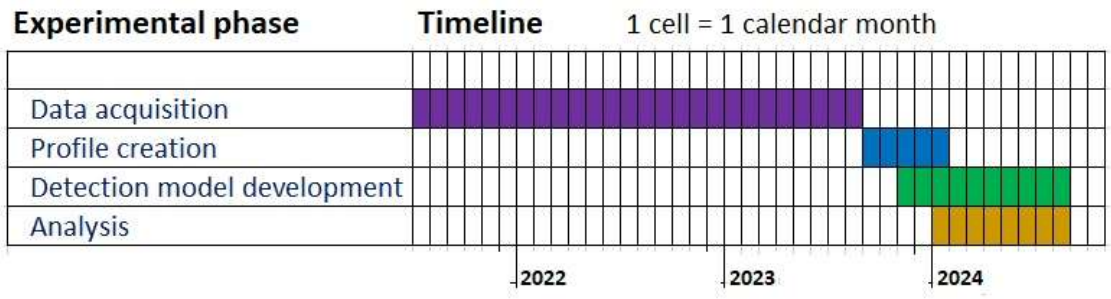


Fig. 4.1: The phases of the experiments

A key decision that had to be made at the early stages of designing the experiments was whether or not to use benchmark keystroke dynamics datasets. As mentioned in Chapter 2, the use of benchmark datasets eliminates the need to collect biometrics solely for experimentation.

When designing these experiments, the need for the acquisition of 'scenario-based biometrics' was realized. As an example, one of the fraudulent acts that the proposed anomaly detection system should be able to detect is the candidate unduly looking up material during the test. The concept proposes that the biometrics of the candidate captured while taking a test genuinely be compared with the biometrics of the same candidate captured while looking up material. For the latter, the biometrics need to be captured essentially during a scenario where the candidate is looking up while taking the test. When using a benchmark dataset, there is no knowledge as to the circumstances under which the biometrics have been collected. They can only be used in experiments where those circumstances are irrelevant. In this study, the circumstances represent a key focus and are extremely relevant. Hence, the option of using benchmark datasets was rejected.

This necessitated the inclusion of a data acquisition phase at the beginning during which biometrics were captured from volunteers, exclusively for use in the experiments.

4.3 The data acquisition phase

Since it was necessary to collect biometrics exclusively for the experiments, it was decided to obtain the services of volunteers to roleplay in scenarios that are required to be examined.

Accordingly, 4 tests were designed to simulate the following scenarios:

a. Candidate undergoing the mock test as part of the registration process.
b. Candidate taking an online test genuinely.
c. Candidate looking up while taking an online test.
d. Candidate collaborating in real time with a competent colleague while taking an online test.

4.3.1 Description of the tests

The purpose of the mock test is to capture biometrics for profile creation. In an actual implementation, the mock test needs to be conducted physically in a proctored environment after the candidate's identity has been duly verified. However, as far as the experiments are concerned, the roleplayers who were merely taking part in experiments had no incentive to impersonate or engage in other types of fraud. Therefore, the first test that simulates the mock test was also designed to be conducted

online. The other 3 tests which are online tests in any case, were also designed to be conducted online.

For the convenient access of the roleplayers, the World Wide Web was chosen as the medium of deployment. Also, since the roleplayers were not under evaluation and were merely taking part in experiments carrying no stakes for them, there was no necessity for the tests to be conducted simultaneously.

To make any person experienced in taking online tests eligible as a roleplayer, test items that covered only general knowledge concerning day to day life were included without testing in any specialized knowledge domain. The questions included objective questions that require a specific answer as well as open-ended questions where roleplayers had the choice of words when answering. Therefore both 'fixed text' and 'free text' were captured.



Fig. 4.2: A sample screen capture showing the response panel for a typical question in the online test interface

Through trial runs, the average time duration for a single test was estimated beforehand to be 30 to 45 minutes. The volunteers were informed of this time estimate to enable

their convenient participation. However, no time restriction was enforced on any test to prevent the biometrics from being influenced by time pressure.

Each test had a series of questions which were posed sequentially upon submitting the answer to each question. Figure 4.2 shows the form interface of a question in one of the tests which would serve as an indicator of the format and style of all 4 tests.

The 'SUBMIT' button was included only to give the roleplayer the feel of an actual online test. The responses submitted were not saved and a form handler was not used. The experiments did not require the evaluation of the roleplayers, so there was no necessity to collect the answers. Clicking the SUBMIT button merely caused the next question to be displayed. On the other hand, the keystroke data were captured from the very first keystroke onwards as soon as the roleplayer started typing inside the text box. Each key event was captured by a background process running on the client side and was transmitted to the server. All the key events received from the client were logged on the server in a JavaScript Object Notation (JSON) file in chronological order, in a folder dedicated to the roleplayer. Figure 4.3 shows a part of one such file.

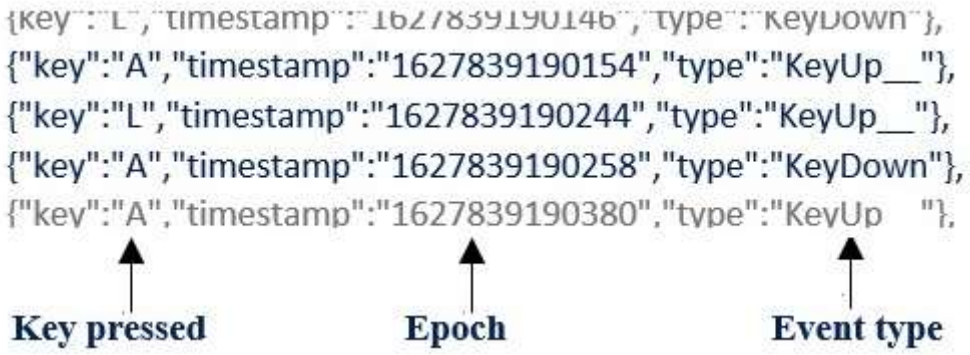


Fig. 4.3: Contents of an example file storing the key event data

The JSON file contains a comma separated series of records. The record structure is congruent with Python dictionaries. Each dictionary represents a log of a single key event and comprises 3 key-value pairs. Each value is a character string defined as follows.

"key"	:	A single alphanumeric character representing the key pressed.
"timestamp"	:	A string of 13 numeric characters representing the epoch of the key event in milliseconds, expressed as a 13 digit positive integer.
"type"	:	Character string "KeyDown" if it was a key down event. Character string "KeyUp__" if it was a key up event.

The value of the "key" item is the alphanumeric symbol displayed on the key itself, on the keyboard. It is an identifier of a 'key' on the keyboard and not necessarily an

identifier of the character generated by typing it. This means that typing in lowercase and uppercase would generate key events having the same value for the item "key". The pressing of the 'SHIFT' to generate the uppercase would meanwhile generate its own key event which too would be captured and logged. The same applies to all multiple characters generated by the pressing of a single key (e.g. '1' and '!', '7' and '&', semicolon and colon etc.).

The integer value represented by the string in the value of the "timestamp" key is the epoch of that key event in milliseconds. It is measured from the UNIX time origin of 0:0:0 Coordinated Universal Time (abbreviated in French as UTC) of 1970.01.01.

The value of the "timestamp" disregards any leap seconds[6] that may have occurred since this origin up to the instance of the epoch.

As explained in Chapter 2, the 'dwell time' and 'flight time' of keystrokes are computed by taking the difference between epochs. As a result, in a system that uses UTC synchronized timestamps, if two epochs encompassing a leap second in between are subtracted, there will be an error of ±1 second. Details of this effect may be learnt from the US NIST website [107]. Since the server that rendered the test suite for the experiments used UNIX time that is not synchronized to UTC, the requirement to perform 'leap second corrections' did not arise.

The JSON file is in effect a 'static playback' of the typing, where a replay of the typing could be constructed purely from the keystroke data.

In Chapter 3, the reason as to why the 'use of a lockdown browser' was not included as part of the concept was explained. Accordingly, the data acquisition phase of the experiments did not specifically require the roleplayers to take the tests using a lockdown browser. Taking the test using any industry standard browser was permitted.

As already mentioned, the purpose of Test[7] 01 was to simulate the mock test conducted during registration and was used to collect biometrics of the roleplayers for profile creation.

Test 02 was used to simulate the scenario of a candidate taking an online test genuinely without indulging in any fraudulent act.

Test 03 was used to simulate the scenario of a candidate looking up material to compose answers while taking an online test. The questions of this test had a flip card displayed alongside the answer text box, displaying a prompt advising the roleplayer

---

[6] A 'leap second' is a correction of ±1 second added to the UTC time scale to account for the tiny difference between the atomic clock second and the solar second. The difference causes a cumulative error which becomes significant over a long period of time. When that happens, a leap second correction is made to the UTC time scale.

[7] In this thesis, the word 'Test' is capitalized when used to refer to the tests listed in 4.3, so as to distinguish it from the general use of the word in other contexts.

to hover over the card to obtain help on answering the question. When the roleplayer hovered over the card, it flipped over to reveal an 'image with text content' that would aid the role player in formulating the answer. An image was used to prevent the transfer of text via the clipboard. Figure 4.4 indicates this lookup.

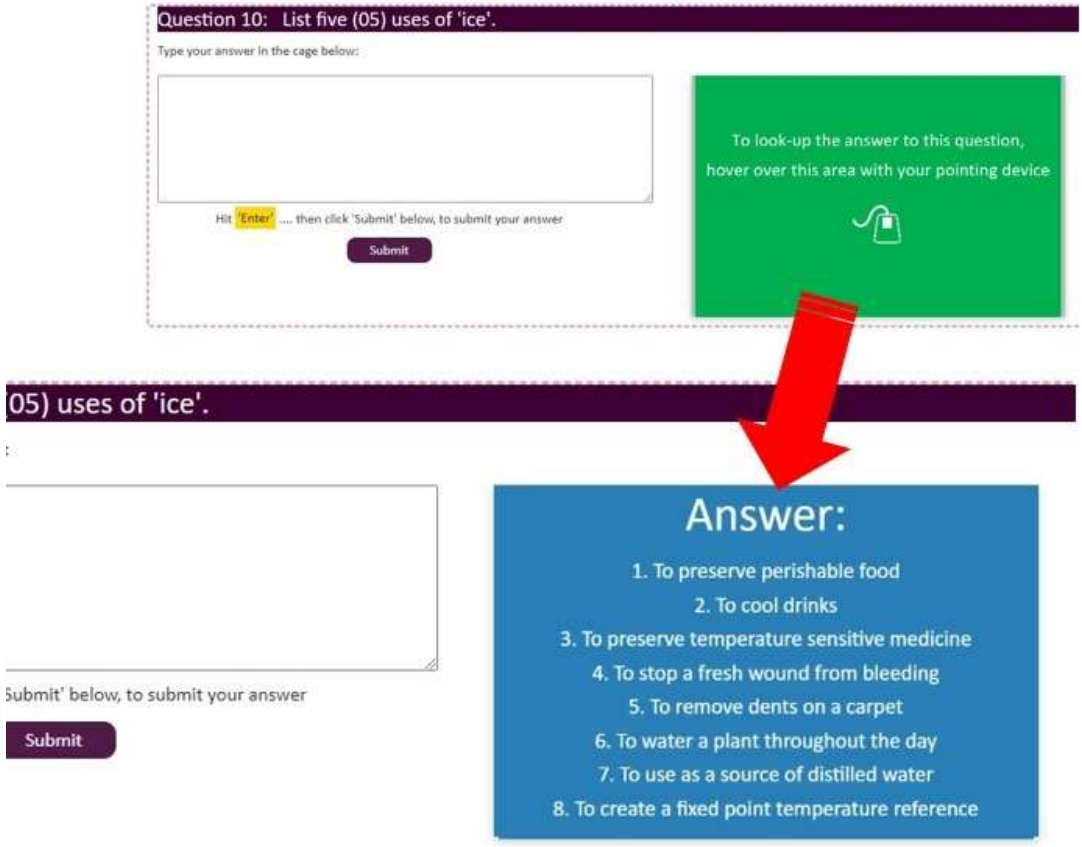


Fig. 4.4 : Simulation of lookup during an online test

The roleplayers were instructed to mandatorily lookup the flip card before answering and not to answer on their own, even if they could. This was critical to the success of the experiment. For this reason, although the test items were based on general knowledge, Test 03 used questions with answers that are less obvious compared to those in Tests 01 and 02 so that the roleplayer would be inclined to flip the card and read it.

Test 04 was used to simulate the scenario of a candidate collaborating with a colleague in real time to formulate answers while taking an online test. Each question of this test had a link to a voice clip placed alongside the answer text box. The roleplayer could play this clip by clicking on it. This was intended to simulate the scenario of a candidate listening to the advice of a competent colleague who could either be present at the location or is in contact via telecommunication. Some questions had links to two voice clips, simulating the collaboration with more than one colleague. Provision was also available for the roleplayer to pause the clip in between and to replay the clip, similar to the scenario where the colleague would offer advice in parts instead of rendering all the advice in one burst and also would repeat what he or she said at the

request of the candidate. Figure 4.5 indicates an example of a screen having two voice clips.

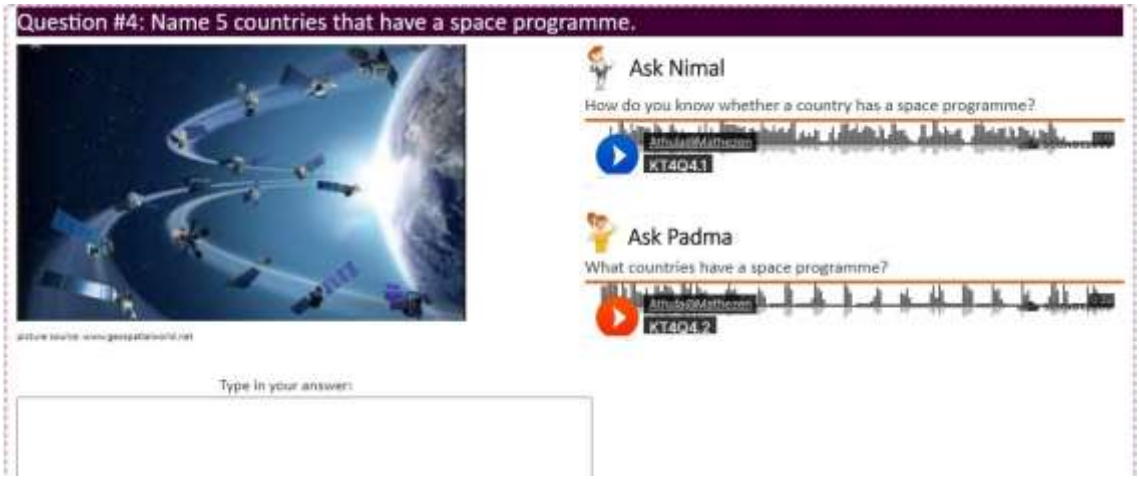


Fig. 4.5 : Simulation of real time collaboration during an online test

As in the case of Test 03, the roleplayers were instructed to mandatorily listen to the voice clips before answering and not to answer on their own, even if they could. Test 04 also used questions with answers that are less obvious compared to those in Tests 01 and 02 so that the roleplayer would be inclined to listen to the clips.

4.3.2 Selection of volunteers for roleplay

As per the terms of the ethical clearance granted for this research, the anonymity of all subjects had to be ensured. Accordingly, it was necessary to ensure that all volunteers for roleplay in the experiments remain anonymous.

To achieve this anonymity, the assistance of two trusted intermediaries was obtained. The two intermediaries were requested to enlist as many volunteers as possible for the experiment independently through their own convenience sampling. The prerequisites given to them for selecting volunteers were as follows:

a. Should be fluent in English language.
b. Should possess keyboard literacy and Internet literacy.
c. Should be experienced in taking online tests by having faced at least one online test.

Each intermediary was given a range of unique Identification Numbers (IDs) to be assigned to the volunteers enlisted by them. The intermediary assigned each volunteer an ID from the range provided. This ID itself served as the login credentials for the volunteer to login to the test suite. No other information about the volunteers were gathered in any way during the tests. The intermediaries maintained strict confidentiality on the identity of the volunteers and the assignment of the IDs to the volunteers. Through this arrangement, the identity of volunteers was effectively decoupled from the research process.

Each of the 4 tests were accessible via separate Uniform Resource Locators (URLs). The URLs were communicated to the volunteers by the intermediaries. Whenever the server flagged the researcher about the completion of a test by a volunteer, the intermediary who enlisted that volunteer was informed. Then the intermediary provided the volunteer the URL to the next test. In this manner the need for any direct communication between the researcher and the volunteers was completely eliminated.

As per the concept, biometrics must be captured without the knowledge of the candidate. However, the terms of the ethical clearance required that the subjects should be aware about what is being collected from them. Therefore, it was necessary to educate the volunteers on what would be collected. The login page for each test contained an unrestricted link to a popup page that provided all details about the tests to the volunteers. This popup briefed all volunteers about the experiments before they commence their roleplay. They were particularly informed of the following:

a. What are being collected.
b. Why those are being collected.
c. That the collection of biometrics is anonymous.
d. That the tests are not intended to evaluate the roleplayers in any manner.

The volunteers were particularly instructed to essentially use a controlled environmental setup that comprised the use of the same physical QWERTY keyboard and avoiding the use of touchscreen keyboards when taking all the tests. They were further instructed to take each test in one sitting without interruption, provide plausible responses, and to keep the test contents confidential.

Although the volunteers were informed that biometrics will be collected, it did not affect the non-obtrusive nature of the biometrics collection process. The capture of the keystroke data as described in 4.3.1 was not conspicuous to the roleplayers. From the briefing, they knew that their keystroke dynamics were acquired but they were unaware as to how it was done. Also, no additional accessories needed to be deployed at the roleplayer's site. This made the Tests of the experiments fully conform to the specification that the collection of biometrics should be invisible to the candidate.

No clarifications had been sought by any volunteer from the intermediaries which indicated that the initial briefing and guidance given online were effective and adequate. There were also no explicit intimations from any volunteer about withdrawing from the experiments.

The data acquisition phase was kept active from 2021.07.18 to 2023.08.31.

When it was decided to conclude the data acquisition phase on 2023.09.01, the completion of tests by the volunteers stood as follows:

Table 4.1: Completion of roleplays by the volunteers

Test #	01	02	03	04
Number of volunteers who had completed the roleplays	59	53	37	29

4.4 The profile building phase

Creation of a unique profile for each roleplayer comprised the following activities:

i. Examination of the biometrics for usability.
ii. Extraction of the 'dwell time vector'.
iii. Extraction of the 'hot sequence vector'.
iv. Extraction of the 'quick sequence vector'.
v. Extraction of the 'far sequence vector'.
vi. Extraction of the 'pause duration ensemble'.

Although the biometrics of 59 roleplayers had been collected through Test 01, the biometrics of 6 of them were found to be corrupted and unusable, leaving the biometrics of only 53 roleplayers available for profile creation.

4.4.1 Extraction of the 'Dwell Time Vector'

The concept presented in Chapter 3 required n_1 numbers of keys on the keyboard to be selected to form the dwell time vector. The keys had to be those used frequently when typing English text.

Accordingly, the following 51 keys were selected for observing the dwell time, thus making $n_1 = 51$:

All keys of the alphabetical characters	26 keys
All keys of the numeric characters	10 keys
The keys of the special characters in the following set:	
{SPACE, PERIOD, COMMA, [,], /, \, =, -, semicolon, reverse_accent, TAB, CR, BACKSPACE, SHIFT}	15 keys
Total number of keys =	51 keys

In the first instance, the dwell time vector was extracted through a script that used the algorithm given in Figure 3.4.

To remove outliers, the script used the following method:

- Another vector with identical dimensions as the dwell time vector was populated with the 'standard deviation' of dwell times of each key by parsing the JSON file again and using the mean dwell times in the vector.
- The dwell time vector was repopulated by identifying outliers in the tuple of dwell times of each key with the aid of the standard deviation vector, applying the threshold of $\mu \pm 3\sigma$, and replacing them with the mean.

4.4.2 Extraction of the 'Flight Time Related Features'

The 'hot sequence vector', 'quick sequence vector', 'far sequence vector', and the 'pause duration ensemble', all of which are related to the flight times of 2-key sequences were computed in a single process that parsed the JSON file.

As described in Chapter 3, hot sequences are multiple key sequences having a high probability of occurrence within English text. Hot sequences involving only two consecutive key strokes were considered for the experiments.

The article by B. Keating titled "Letter Frequencies in the English Language" [108] gives details about the distribution of sequences of letters in typical English Language text. Figure 4.6 shows the relative frequency distribution of the top 20 sequences of 2 letters found through the scanning of 2,383,373,483 2-letter sequences contained in 9,481 English works as mentioned in this article. This has 17 sequences that are common to the set of 25 most frequent 2-letter sequences in the 'Buffalo datasets'. Details about these widely used datasets are given in the work of Huang et al. [109].

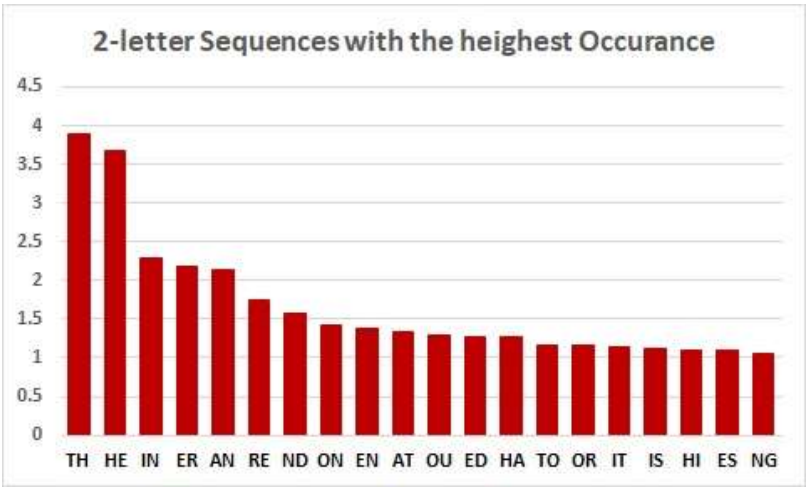


Fig. 4.6: Relative frequencies of the highest occurring 20 sequences of two letters in English text (Vertical axis shows percentage from 2,383,373,483 2-letter sequences analyzed).

For the experiments, in addition to these 20 sequences, 5 more sequences were added after examining the JSON logs resulting from roleplayers taking Test 01. Accordingly, there were 25 hot sequences as follows, making $n_2 = 25$:

{ 'IN', 'NG', 'IE', 'ED', 'TH', 'HE', 'IT', 'EI', 'ER', 'AN', 'RE', 'ND', 'ON', 'EN', 'AT', 'OU', 'HA', 'TO', 'OR', 'IS', 'HI', 'ES', PERIOD +SPACE, COMMA+SPACE, PERIOD +CR}

Computing the flight times (di-graphs) is more complex than computing dwell time. Unlike for dwell time where only the two events of single keystrokes need to be considered, flight times must take several possibilities into account.

There are 3 possible cases for the timing of the key events in a 2-key sequence.

i. The two key presses being completely exclusive (i.e. the second key is pressed after the first key is released).
ii. The two key presses overlapping in time (i.e. the first key is released after the second key is pressed and before the second key is released).
iii. The events of the second key encompassed within the events of the first key (i.e. the second key is pressed and released before the first key is released).

Occurrence of cases 'ii' and 'iii' can be detected on the JSON file. If only the key presses described in 'i' were made, there will only be consecutive pairs of key down and key up events of single keys.

In addition, the case of a single key being held down continuously also needs to be considered. This usually happens in the case of the modifier keys like SHIFT, ALT, and CTRL which need to be held depressed while other keys are typed. Any other key too could be held depressed inadvertently.

When a key is pressed on a computer keyboard, a key down event for that key will occur instantly. If the key is held depressed and no other key is pressed, there will be a fixed delay before the second key down event for that key is generated. This will be followed by a continuous stream of key down events of that key at a fixed rate, until the key is released or another key is pressed. The fixed delay and the rate of repeated key down events are governed by the keyboard settings 'repeat delay' and 'repeat rate' in the computer. Both are user configurable. When a key is held down with no other keys pressed, there will be records of two consecutive key down events of that key, in the JSON file. They will have epochs with a difference equal to the repeat delay. Thereafter there will be a series of key down events of that key. The difference in epochs of these events will be constant and equal to the reciprocal of the repeat rate.

When parsing the JSON file in search of the key down event of the second key of a 2-key sequence, it is necessary to consider each of the above cases separately. This makes the flight time computations much more complex than dwell time computations. Therefore the basic algorithm of Figure 3.8 had to be suitably enhanced to deal with the above complexities. The detailed algorithm is given in Appendix – A.

Although only flight times of type 1 were proposed to be considered, the script was written to extract the flight times of all 4 types so that it would be more general and

useful for future work. The flight times of each of the 25 hot sequences were initially written into a 3-D array having 4 slices, with a slice allocated for each of the 4 types of flight times. Each slice had 25 rows, one allocated for each hot sequence. The cells in each row were used to store the flight time of each occurrence of that hot sequence. Figure 4.7 illustrates the structure of this 3-D array.

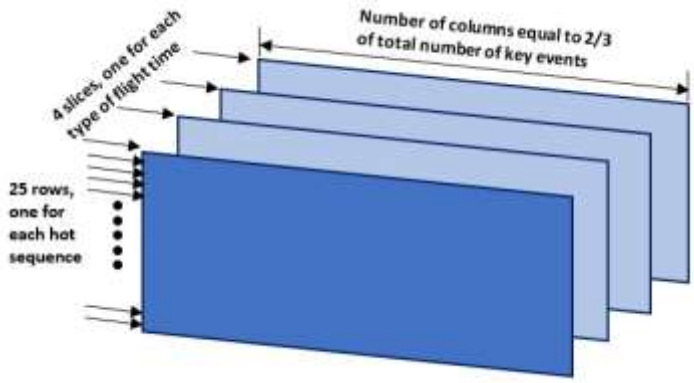


Fig. 4.7: Structure of the 3-D array used to store hot sequences

Since the number of occurrences of a hot sequence cannot be defined beforehand, a number of columns equal to two thirds of the total number of events in the JSON file was allocated, which could be lavish but would ensure zero runtime errors in case any hot sequences for any candidate had high occurrence.

Thereafter, the mean flight time and standard deviation of each hot sequence were computed and with the aid of those, outliers in each row were detected using a threshold of $\mu \pm 3\sigma$. Mean flight times were recomputed after eliminating the outliers.

Next, a 2-D array having the structure shown in Figure 4.8 was populated using the data in the 3-D array.

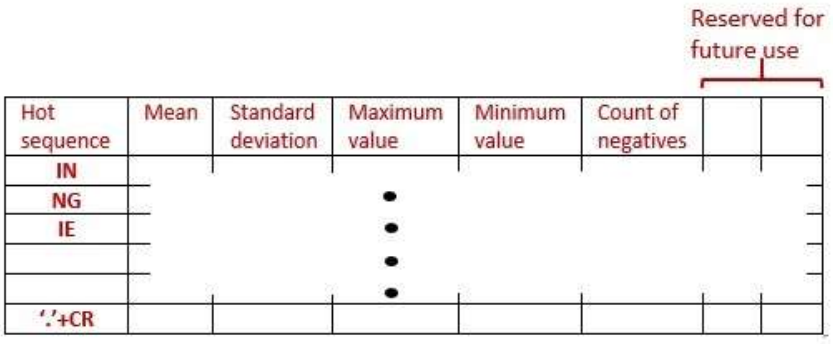


Fig. 4.8: Structure of 2-D array for storing the parameters of 'hot sequences'

When populating this matrix, only the first slice of the 3-D flight times matrix was considered. This amounted to taking only the flight times of type 1 as proposed in the concept.

It will be noted that the second column of this 2-D matrix allocated to hold the mean flight time with outliers eliminated is in fact the 'hot sequence vector' that is of interest. The other columns were also populated with parameters anticipated as being useful for analysis.

Quick sequences are 2-key sequences having negative flight times of type 1. As explained in Chapter 3, they cannot be predefined commonly for all candidates, unlike the hot sequences.

Quick sequences were detected by parsing the JSON file looking for 2-key sequences which give negative flight times of type 1. They were processed in a manner similar to how the hot sequences were processed. Only the flight time of type 1 was used so only a 2-D matrix was necessary. A data frame with the structure shown in Figure 4.9 was created for the quick sequences.

Quick sequence	Number of occurrences within JSON file	Aggregate flight time	Hot sequence (Y/n) ?

Fig. 4.9: Structure of the data frame for storing 'Quick Sequences'

Since the quick sequences are not predefined, the rows were allocated dynamically to each quick sequences detected, during runtime. The second column stored the count of occurrences of that quick sequence 'if the flight time was negative' while the third column was used to store the aggregate flight time of all occurrences of that sequence. It will be noted that the second and third columns could be used to compute the mean flight time of each quick sequence and that would yield the quick sequence vector of that roleplayer. The last column stores a flag whether the quick sequence is one of the hot sequences as well. This was in anticipation of possible optimization of processing during anomaly detection since the hot sequence data are separately on record.

The far sequences were processed in a manner identical to that of hot sequences. As explained in Chapter 3, far sequences are those typed using the index finger and little finger of each hand. There are 72 far sequences as follows:

{ QR, QF, QV, QT, QG, QB, AR, AF, AV, AT, AG, AB, ZR, ZF, ZV, ZT, ZG, ZB,
RQ, FQ, VQ, TQ, GQ, BQ, RA, FA, VA, TA, GA, BA, RZ, FZ, VZ, TZ, GZ, BZ,
PY, PH, PN, PU, PJ, PM, ;Y, ;H, ;N, ;U, ;J, ;M, /Y, /H, /N, /U, /J, /M,
YP, HP, NP, UP, JP, MP, Y;, H;, N;, U;, J;. M;, Y/, H/, N/, U/, J/, M/ }

The computed flight times were stored in a data frame having a structure similar to that shown in Figure 4.9 where the first column stored the 'far sequences' and the last column was not used.

Since the pause duration ensemble is not multidimensional and is simply an ensemble of scalar values, it has no predefined size. The JSON file was parsed from beginning to end looking for occurrences of key events leading to whitespaces namely, the SPACEBAR, the TAB key, and the ENTER key. Pauses are generally made after typing these characters.

It should be clarified that the duration of the pause is not the dwell time of the whitespace key. The pause begins as soon as the key preceding the whitespace key is released. As stated in the previously cited work of Torrance and Galbraith [105], some typists may immediately press the whitespace key while others may press it during the pause. Therefore, taking the key up event of the preceding key is more accurate. Also, a typist may release the whitespace key while continuing to pause. The pause actually ends when the key following the whitespace key is pressed.

Accordingly, whenever a key down event of any of the above whitespace keys was encountered, the duration of the pause was computed by subtracting the epoch of the key up event of the immediately preceding key from the epoch of the key down event of the immediately following key. When searching for the key down event of the next key, the case of the whitespace key being held down was taken into account similar to what was done when detecting hot sequences.

All the pause durations thus computed were pushed into a 1-D array. After the elimination of outliers, this array comprised the 'pause duration ensemble'.

At various stages of extracting the above features, the biometrics of 15 more roleplayers were found to be unusable. The most common reason was that the JSON file was small, implying that the capture of keystrokes has not happened properly. Accordingly, profiles were created successfully for the remaining 38 roleplayers.

4.4.3 Extraction of non-temporal features

The following features were also extracted from the biometrics captured during Test 01, in anticipation of exploiting the possibility of using 'non-temporal' features to supplement the time based features, as explained in Chapter 3.

i. Raw typing speed
ii. Effective typing speed
iii. Typing error rate

The above were computed using the following definitions:

$$\text{Raw typing speed} = \frac{Total\ number\ of\ key\ events}{Total\ duration\ of\ capture} \quad \text{…….. (4.1)}$$

$$\text{Effective typing speed} = \frac{Total\ number\ of\ key\ events - count\ of\ rubouts}{Total\ duration\ of\ capture} \quad \text{..(4.2)}$$

$$\text{Typing error rate} = \frac{Total\ number\ of\ rubouts}{Total\ number\ of\ key\ events}\ \% \quad \text{…….. (4.3)}$$

The above definitions are not 100% accurate as per the classical definitions used in the typing trade as mentioned in the work of Kumar et al. [67] and should not be taken literally. Due to the possibility to have repeated key down events when a key is held down, the number of key events cannot be directly related to the number of characters typed. The metrics defined above are intended for use in authentication only.

The non-temporal features were also added to the profile of each roleplayer. The 'error correction mode' being another non-temporal feature encountered in the review of related work, was not used in this study.

4.5 The detection model development phase

As outlined in the ITU-T Technology Watch Report [31], there are two modes in which a biometrics-based authentication system could operate. One mode is the 'verification mode', where the system checks whether a person claiming to be a registered user is actually the registered user, by comparing a biometric sample provided at the time of access against the user's biometrics on record in the profile. The other mode, called the 'Identification Mode' attempts to identify the person requesting access as one of the users registered with the system by comparing the requester's biometric sample successively against the biometrics stored in the profiles of all registered users.

The concept presented in Chapter 3 is only concerned with determining whether the user is actually the registered user or an imposter. Therefore what is applicable is the 'verification mode'. It is also extended to the scenario where the person is in fact the registered user but the test taking is fraudulent. Therefore, what is necessary is for the system to possess some entity which has learned the peculiarities of the biometrics of the registered user and is capable of verifying whether a biometric sample captured at a later stage also belongs to the same person.

To train such entity for each feature using supervised learning, balanced training datasets are required. How such datasets were created for the 'dwell time related features' is described as an example.

4.5.1 Model development for dwell time related features

Consider the generation of data points to train a classifier to detect anomalies in the 'angle of departure of the dwell time vectors'. The dwell time vector is already available in the profile of that roleplayer. The JSON file created during Test 01 containing the key events acquired while this roleplayer took Test 01 is also available in the profile. Since the JSON file represents a static playback of the session, a replay of the taking of Test 01 could be made using the JSON file.

The concept proposed that the JSON file be parsed in batches of **M** key events. It was also stated that **M** should be selected with care, since the choice was critical to the success of the experiment. Trial runs were conducted using the JSON file of Test 01 of a random sample of roleplayers. Successive approximation was used to converge on a suitable value of **M** by trying out different values of **M** and observing the authentication cycle time. The value of 40 key events ultimately yielded the best compromise. Accordingly, **M** was chosen to be 40.

It should be noted that authentication is not applicable to the 'mock test' and a playback 'with authentication' is made purely for the generation of training datapoints.

4.5.1.1 Generation of training data for 'genuine test taking'

```
player ← command line argument

Open JSON file of test 01 of player as dictionary array key_events[]
Retrieve vector mu_dwell[] of player
Initialize mu_dwell_hat[]
Initialize blank array autotheta[] to store theta

Point to the first dictionary element in key_events[]
Initialize the index of array autotheta[]
while the end of key_events[] is not reached:
        m = 0
        while m < 40 :
                Parse the array key_events[] populating mu_dwell_hat[] along the way
                m = m + 1
        Determine subspace of mu_dwell_hat[]
        Project mu_dwell[] into the subspace
        theta ← angle (mu_dwell_hat[],projected mu_dwell[])
        Push theta to array autotheta[]
        Label theta as 'genuine'
        Advance the index of autotheta[] by 1
        Advance the index of key_events[] by 40

Save array autotheta[] in folder of player
Close all open files
```

Fig. 4.10: Algorithm to generate datapoints for genuine test taking

The algorithm given in Figure 4.10 was used to generate the stream of angles of departure, θ for genuine test taking. The array of θ values generated was named '*autotheta[]*' to highlight that θ is the angle between two vectors pertaining to a single person.

4.5.1.2 Generation of training data for 'fraudulent test taking'

As explained in Chapter 3, the easiest way to simulate fraudulent test taking was to use the biometrics of a different roleplayer. In that case the dwell time vector *mu_dwell[]* will be of the roleplayer under consideration while the JSON file will be of another roleplayer.

Consequently, *mu_dwell_hat[]* will relate to the other roleplayer. As a result, the angles of departure will have a different clustering. This is what the classifier must learn to discriminate for anomaly detection.

The algorithm given in Figure 4.11 which is a modified version of that shown in Figure 4.10 was used to generate the stream of angles of departure θ, for fraudulent test taking. This array of θ values was named '*crosstheta[]*' to highlight that θ is the angle between two vectors pertaining to different persons.

```
player ← command line argument 1
other_player ← command line argument 2

Open JSON file of test 01 of other_player as dictionary array key_events[]
Retrieve vector mu_dwell[] of player
Initialize mu_dwell_hat[]
Initialize blank array crosstheta[] to store theta

Point to the first dictionary element in key_events[]
Initialize the index of array crosstheta[]
while the end of key_events[] is not reached:
        m = 0
        while m < 40 :
                parse the array key_events[] populating mu_dwell_hat[] along the way
                m = m + 1
        Determine subspace of mu_dwell_hat[]
        Project mu_dwell[] into the subspace
        theta ← angle (mu_dwell_hat[],projected mu_dwell[])
        Push theta to array crosstheta[]
        Label theta as 'fraudulent'
        Advance the index of crosstheta[] by 1
        Advance the index of key_events[] by 40

Save array crosstheta[] in folder of player
Close all open files
```

Fig. 4.11: Algorithm to generate datapoints for fraudulent test taking

4.5.1.3 Building the composite dataset

To create the composite training dataset for each roleplayer, 100 data points labelled as 'genuine' and 100 datapoints labelled as 'fraudulent' were used, making a balanced set of 200 data points. The data points labelled as 'genuine' were picked through random sampling from the array *autotheta[]* of the roleplayer under consideration. The 100 datapoints labelled as 'fraudulent' were picked through random sampling from the arrays *crosstheta[]* of 5 other roleplayers. 20 points were picked from each of them to make a total of 100. Accordingly, the dataset had 100 points labelled as 'genuine' and 100 points labelled as 'fraudulent', thus making a balanced dataset of 200 points. This is important because the effective training of a binary classifier requires an approximately equal number of datapoints having each label.

4.5.1.4 Selection of roleplayers for simulation of 'fraudulent test taking'

As explained in Chapter 3, the roleplayers who are sources of data to be labelled as 'fraudulent' should be selected in a manner that the dataset provides the classifier the best learning about the differences in the clustering.

A vague indicator of the clustering of a set of theta values would be its centroid. The centroid represents the central tendency of the data. If the mean values of two sets of theta are compared it would generally be a reasonable indication of the difference between the central tendencies of the two sets.

Since there were 38 valid profiles created, for each roleplayer there were 37 other roleplayers to choose from. Accordingly, the following procedure was used to select the 5 roleplayers as best sources of data to be labelled as 'fraudulent'.

i. The array *autotheta[]* was generated for the roleplayer under consideration, using a script based on the algorithm given in Figure 4.10.
ii. Taking each of the other 37 roleplayers in turn as the 'other player', 37 *crosstheta[]* arrays were generated in respect of the roleplayer under consideration.
iii. Mean values of all arrays were computed.
iv. The 5 roleplayers having the 'smallest' difference between the mean of the *crosstheta[]* array and the mean of the *autotheta[]* array were selected as the best sources of data to be labelled as 'fraudulent'.

When theta values having a smaller difference in clustering are chosen, the classifier will learn to discriminate between smaller differences in clustering between genuine and anomalous datapoints and is likely to be a better classifier. Smaller differences represent the worse cases which are more difficult to detect. If theta values having a more significant difference in mean values were chosen, there is the possibility for the classifier having learned from such data to miss smaller differences in clustering.

4.5.1.5 Choosing a classifier model

As mentioned in Chapter 3, a classifier model needs to be selected for each roleplayer separately because the performance of the classifier would depend on the peculiarities of each roleplayer's datasets. Different classifiers work on different principles, operate on different types of data, and entail different processing overheads.

When selecting a classifier model, the classical theories of machine learning were relied upon. Two sources that were consulted for this purpose are the web article by MathWorks titled "Choosing the Best Machine Learning Classification Model and Avoiding Overfitting" [110] and the book by M. Awad and R. Khanna [111]. As learnt from these sources, it is customary to consider the following factors among other things when deciding on a classifier model:

- Volume of data available for training
- Type of data
- Storage
- Processing

The volume of data in these experiments is relatively small. Hence, it was deemed suitable to choose classifiers that could learn from a relatively small dataset. Accordingly, the following 4 types of binary classifiers were chosen to be used in the experiments:

i. Naïve Bayes (NB) classifier.
ii. Logistic regression (LR) classifier.
iii. Support vector machine (SVM) classifier.
iv. Random forest (RF) classifier.

The following variants of training datasets were used to train the classifiers:

i. The dataset comprising the θ values themselves (the basic dataset).
ii. The dataset with outliers replaced with the mean value of the 'source' of θ.
iii. The dataset normalized using the mean value of θ of the 'genuine test take'.
iv. The combination of 'ii' and 'iii'.

It will be noted that the last 3 types of datasets listed above are linear transformations of the basic dataset. Accordingly, with 4 types of classifiers and 4 variants of training data derived from the dwell time related feature, it was possible to train 16 classifiers for each roleplayer.

This was for the dwell time related feature alone. Classifiers are required for each of the other features as well.

4.5.2 Model development for the remaining features

Apart from the 'dwell time vector', there are 4 other features namely, the hot sequence vector, the quick sequence vector, the far sequence vector, and the array of pause times, available in the roleplayers' profiles, all based on the flight times.

For each of these features, the same process used for dwell time based features comprising the generation of training data for genuine test taking, selecting 05 roleplayers as best sources of anomalous data, generation of training data for fraudulent test taking, building the composite training dataset, and training the 4 types of classifiers was attempted. While the process was similar, there were few peculiarities observed.

Compared to the dwell time vector and the pause duration ensemble, it was found that the population of the other vectors from a sequence of 40 key events was sparse. For example, for the hot sequence vector, only 3 or 4 hot sequences were captured within most of the 40 key event sequences. This is to be expected, because within a given piece of text, the occurrences of specific 2-key sequences will be far less than the occurrences of individual keys. In fact, it is even possible for a 40 key event sequence not to have any hot sequences. Such occurrences too were visible during the experiments.

One way to increase the occurrences of hot sequences within a sequence of **M** key events would be to choose a higher value of **M**. The probability of a particular 2-key sequence being present within a sequence of **M** events would increase with **M**. However, increasing **M** would have consequences.

Since the classifier for the dwell time based feature is already working well with **M** = 40, key event sequences of integer multiples of 40 could be used for authentication using the remaining features so that synchronized detections by all features occur. However, this would result in longer time gaps between such synchronized authentications and would violate the condition of continuity of the authentication process, which is critical.

This phenomenon was even more acute in the case of the ‘quick sequences’ and the ‘far sequences. It was observed that in the case of the majority of roleplayers, consecutive batches of 40 key events passed without a single ‘quick sequence’ or ‘far sequence’ being detected, resulting in null vectors, which in turn made the angle of departure ambiguous.

Accordingly, it was decided not to use the features of quick sequences and far sequences and to use only the hot sequences and the pause durations for the experiments. One type of 2-key sequences is sufficient to establish proof of the concept. Therefore, out of the remaining features, training data sets were created only for the ‘hot sequences’ and ‘pause durations’. Each dataset had the 4 variants described in 4.5.1.5 and each of them were used to train the 4 types of classifiers. Together with the 16 classifiers for the dwell time based features there were a total of 48 classifiers in each roleplayer’s profile.

4.5.3 Selecting the best classifier

For each feature extracted from the biometrics, there would be one classifier that performs best in anomaly detection. For each roleplayer, out of the 16 types of

classifiers for each feature, it was necessary to identify the best classifier, giving each profile 3 best classifiers.

The criterion to be used in selecting the best classifier had to be established considering the peculiarities associated with authentication during online tests. In all other types of access restricted systems, it is customary to adopt a 'zero tolerance' policy. Whenever even the slightest doubt about the authenticity of a user is detected at any stage of an online session, a re-authentication would be triggered immediately. If that fails, it would immediately trigger the denial of service. If it was a false rejection, that would be rectified by engaging the user and authenticating the user outside the system. In such systems, although a false rejection would inconvenience the user, a user would not mind the temporary denial of service done in the user's own interest and would be willing to re-establish authenticity independent of the system to regain access to the system.

However, any attempt to re-authenticate a candidate who is taking the test genuinely during the test would unduly distract the candidate from the test and place him at a disadvantage due to time pressure and the mental turbulence caused by the unexpected interruption. Denial of access due to a false rejection of a genuine candidate during an online test would be even worse because there will not be another opportunity for that candidate to take the same test. Therefore, "not rejecting genuine candidates" should take priority over the "detection of imposters" and the candidate should always have the benefit of the doubt. Plainly put, low false rejection rate is preferred over high true rejection rate when it comes to authentication in online tests.

Therefore, the criterion to select the best classifier was that the classifier should produce the lowest rate of false rejections of the roleplayer. For each roleplayer, there will be 3 such 'best classifiers', one for each feature.

To identify the best classifiers, a run of each roleplayer taking the Test 01 was simulated and was monitored using each classifier in turn. The classifier that exhibited the lowest false rejection rate (FRR) was identified as the best classifier for that feature. This criterion could be mathematically expressed as follows:

$$FRR(best_classifier) = MIN(FRR(classifier_j)), \text{ where } j = 1,2,3, \ldots 16 \quad \ldots\ldots (4.4)$$

The 3 best classifiers identified for each roleplayer were recorded in a common database separate from the profiles.

4.5.4 Consistency with the concept

It is important to note that only the biometrics captured during Test 01 were used during the entire process of profile creation including the training of classifiers. Although the biometrics from all 4 tests were available at the beginning of the profile creation phase, no biometrics whatsoever from Tests 02, 03, or 04 were used for anything connected with profile building. Considering that Test 01 simulated the mock test which is carried out during registration in actual practice, an institution that uses a

system based on the concept should also be able to create profiles for candidates using the biometrics from the mock test alone before any online tests are conducted. Therefore, the manner in which profiles were created during the experiments is fully consistent with the concept presented in Chapter 3.

4.6 The analysis phase

During this phase, simulations of various scenarios were carried out using the biometrics captured from Tests 02, 03, and 04. The classifiers that were developed as described above were employed in anomaly detection in the simulated scenarios. Their performance was assessed using both traditional as well as time related metrics.

4.6.1 Repeatability of experiments

When exercising continuous authentication during an actual online test, the candidate's keystroke dynamics should be captured and compared against those in the profile, while the online test is in progress.

When the taking of Tests 02, 03, and 04 during the experiments are considered, it will be noted that the biometrics exhibited by each roleplayer during those tests are the same biometrics that would have been used for continuous authentication, had these tests been actual online tests.

As a further elaboration of this, consider the JSON file of a particular roleplayer **P** from Test 02. This file contains a static playback of the actual test taking with records of all key events that occurred while **P** was typing during Test 02, listed in chronological order. The time instance of occurrence of all key events are precisely known. The timestamps recorded in them would be the same timestamps that the authentication system would have captured if continuous authentication was actually employed while roleplayer **P** took Test 02 online. This authentication process could therefore be simulated offline by parsing the JSON file or more specifically, by replaying the taking of the test. Therefore, it follows that offline analysis of the biometrics captured during the tests is equivalent to actual continuous authentication during the test and is adequate for experimental purposes.

This provided for the unlimited simulation of various scenarios and for experiments to be repeated as many times as required. For instance, the scenario of genuine test taking by a candidate could be simulated by parsing the keystroke dynamics exhibited by a roleplayer during Test 02 and comparing them against those in the profile of the same roleplayer. If a scenario of impersonation needs to be simulated, the keystroke logs of any other roleplayer when taking Test 02 could be parsed and compared against those in the profile of the first roleplayer.

Similarly, parsing of the keystroke logs of the taking of Test 03 by any roleplayer would be a simulation of a scenario of looking up while parsing the keystroke logs of Test 04 would simulate a scenario of collaboration.

4.6.2 Simulation of impersonation

There were 38 roleplayers for whom profiles were created. For each roleplayer, there is the possibility to simulate 37 impersonations by taking each of the other roleplayers in turn as the imposter.

Figure 4.12 shows the algorithm used to simulate continuous authentication amidst impersonation using the angle of departure of the 'dwell time vector':

```
player_1 ← command line argument 1
player_2 ← command line argument 2

Retrieve dwell_time vector of player_1 as mu_dwell[]
Retrieve best classifier of player_1 for dwell time features as model()
Open JSON file of test 02 of player_2 as dictionary array key_events[]

genuine ← TRUE
point to first dictionary element of key_events[]
while end of key_events[] is not encountered:
    create blank matrix mu_dwell_hat[] with structure similar to
            mu_dwell[]
    read M consecutive key events from key_events[]
    compute mu_dwell_hat[] of the N keys within the M events
    determine subspace of mu_dwell_hat[]
    project mu_dwell[] to subspace of mu_dwell_hat[]
    theta ←  angle between mu_dwell_hat[] and projected
            mu_dwell[]
    genuine ← model.predict(theta)
    advance pointer of key_events[] by M dictionary elements

close all files
```

Fig. 4.12 : Algorithm for simulation of continuous authentication amidst impersonation.

It will be noted that this algorithm too is "generic" like the one given in Figure 3.16 in that it can simulate authentication during genuine test taking as well as impersonated test taking

A single run of the while loop represents a single round of authentication. A detection results at the end of every execution of the while loop giving a stream of detections

which terminates when the end of the JSON file is reached. This is continuous authentication in all respects and is consistent with what is proposed in section 3.9.

The initial setting of flag *genuine* to TRUE is consistent with the policy where genuine test taking is considered as the default while impersonation is considered as the exception the system is attempting to detect.

Simulation of continuous authentication using the hot sequence vector is similar and can be understood using the same algorithm. In this case, the objects fetched from the profile of *player_1* would be the hot sequences vector and the classifier trained on features extracted from hot sequences.

Simulation of continuous authentication using the pause durations is also based on the algorithm in Figure 4.12, except that only the classifier is fetched from the profile of *player_1*. It is not necessary to fetch the pause duration ensemble itself unlike in the case of the other two features. By virtue of having been trained on training data that includes the pause durations captured while *player_1* took the mock test, it represents *player_1*. Since there is no computation of an angle of departure between two vectors, no other object is required from the profile.

As far as authentication at subsequent tests is concerned, there is no necessity to retain the pause duration ensemble in the profile after the classifiers are trained. However, in an actual implementation, it is necessary to retain this ensemble in the profile for the following reasons:

i. It is required for the generation of training data when new classifiers are trained.
ii. It is a potential source of anomalous data points to train the classifiers of new candidates.

During a simulated run of authenticated test taking, the 3 independent classifiers can be made to operate in tandem, on different authentication parameters computed from each sequence of **M** key events. The OR combination of the 3 detections are taken.

An important difference between the computation of features for inclusion in the profiles and the computation of features during authentication is that during the latter, outlier elimination is not performed. Outliers themselves could be the result of a fraudulent act. Therefore they should be retained as vital clues about fraudulent acts.

It is also important that the data fed to the classifier during authentication matches the variant of the dataset that was used to train that classifier. For example, if the 'best classifier' of a candidate for a certain feature is the random forest classifier trained using the normalized dataset, then during subsequent tests, the authentication data derived from that feature must also be normalized before feeding them to the classifier. More generally, the authentication data must use the same linear transformation used for the data from which the classifier learned. The algorithm in Figure 4.12 shows only the particular case where the basic dataset itself is directly used, for the sake of simplicity.

To facilitate this, the database of best machines also holds a record as to which variant of training data had been used to train that classifier.

For each of the 38 roleplayers, a run of impersonated taking of Test 02 by each of the other 37 roleplayers were simulated in turn and the detections were recorded. Altogether there were 38*37 simulations of impersonation. From the recorded detections, the performance metrics such as FAR, T2FD, and FAWI were computed.

4.6.3 Simulation of looking up and real time collaboration

Where the scenarios of looking up and real time collaboration are concerned, there is only a single roleplayer involved.

Figure 4.13 shows the algorithm used to simulate continuous authentication amidst looking up. The angle of departure of the 'dwell time vector' is used as the example.

```
player ← command line argument 1

Retrieve dwell_time vector of player as mu_dwell[]
Retrieve best classifier of player for dwell time features as model()
Open JSON file of test 03 of player as dictionary array key_events[]

genuine ← TRUE
point to first dictionary element of key_events[]
while end of key_events[] is not encountered:
        create blank matrix mu_dwell_hat[] with structure similar to mu_dwell[]
        read M consecutive key events from key_events[]
        compute mu_dwell_hat[] of the N keys within the M events
        determine subspace of mu_dwell_hat[]
        project mu_dwell[] to subspace of mu_dwell_hat[]
        theta ← angle between mu_dwell_hat[] and projected mu_dwell[]
        genuine ← model.predict(theta)
        advance pointer of key_events[] by M dictionary elements

close all files
```

Fig. 4.13 : Algorithm for simulation of continuous authentication amidst lookup.

The classification using the other 2 features can also be understood based on this algorithm.

The simulation of real time collaboration was similar except that the biometrics of Test 04 were used.

Not all roleplayers who had completed Test 02 had completed Tests 03 and 04. Therefore, there were only 34 roleplayers with usable biometrics from Test 03 and only 26 roleplayers with usable biometrics from Test 04. Accordingly, only those numbers of simulations were possible.

Detections made during each simulation were recorded. From the recorded detections, the performance metrics were computed.

4.6.4 Use of non-temporal features

The following non-temporal features explained in section 4.4.3, extracted from biometrics captured during Test 01 were already available in each profile.

i. Raw typing speed
ii. Effective typing speed
iii. Typing error rate

Accordingly, the above features were also extracted and recorded during the simulation of continuous authentication during Tests 02, 03 and 04.

Detailed algorithms used in the above methods are included in Appendix – A.

4.7 Supplementary experiments

Although not directly linked to the proof of the concept, two supplementary experiments were carried out to make additional observations that were envisaged as being useful for the establishment of the system. They were,

i. Protocol analysis of the client-server communication during the capture of keystroke data.
ii. Examination of clipboard transfers during online test taking.

4.7.1 Protocol analysis of the client server communication during the acquisition of keystroke data

This experiment was motivated by the observation made in the work of A. Wengelin and V. Johansson [102] found during the review of related work that keystroke logging generates 'massive amounts of data'. Since one aim of adopting keystroke biometrics based authentication during online tests as opposed to remote proctoring was to reduce the communication overheads, the volume of data that keystroke logging would generate merits investigation.

For this purpose, sample tests were simulated on a browser. The tests were rendered from a local host running on the same machine running the browser. A software protocol analyzer installed on the machine was used to sniff the data transfer between the browser and the local host via the 'local loopback channel'. Since person dependence was not relevant in these experiments and only a functionality was being examined, no population was used. The candidate's role was played by the researcher himself.

4.7.1.1 Examining the nature of the data transfer

Question 1 of Test 01 was used as the sample test for this purpose.

With data capture in the protocol analyzer turned on, the character string **'1234ABCD'** followed by the carriage return was typed into the answer box. The 'Submit' button was clicked and the data capture was turned off on the protocol analyzer.

Figure 4.14 is a screen capture of the browser screen rendering the test, just before clicking the 'Submit' button.

Figure 4.15 shows a screen capture of the protocol analyzer window with data captured from the communication between the browser and the local host.

The data captured on the protocol analyzer were saved for later analysis.

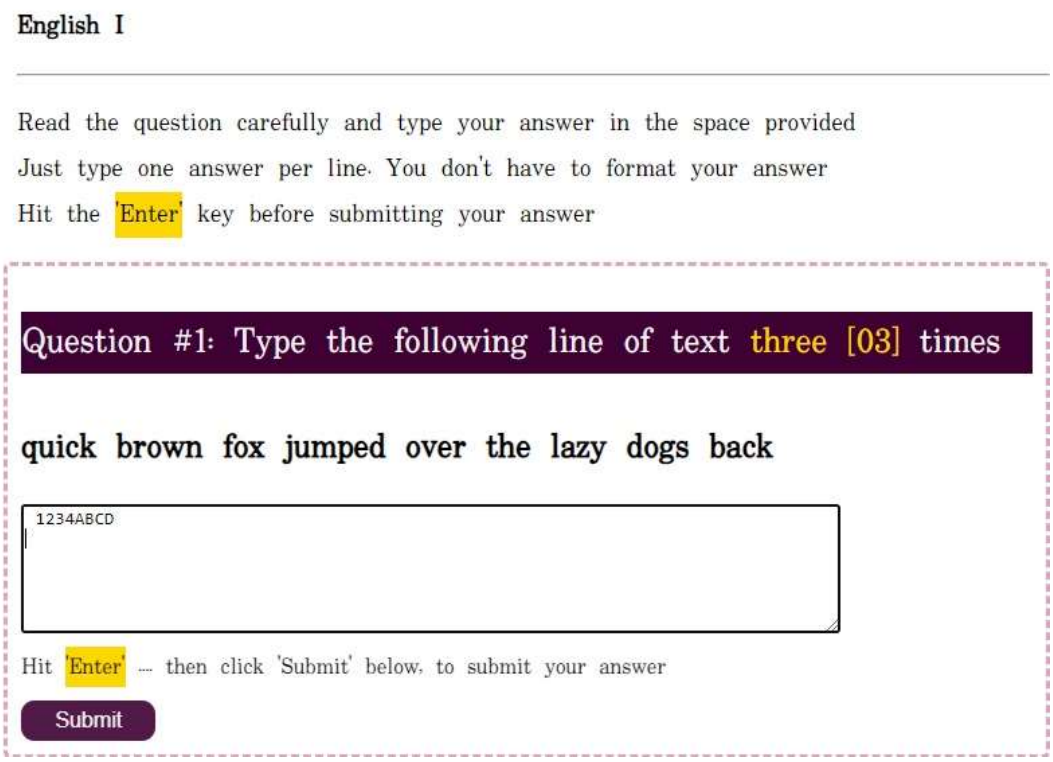


Fig. 4.14: The test character sequence typed into the answer box of question 1 of Test 01, just before submission.

4.7.1.2 Examining the effect of the volume of text typed

The above experiment used a short string of characters to examine the nature of the data transfer when the keystroke data were transmitted to the back end. To check the effect of larger volumes of text typed, the above experiment was repeated using a slightly modified apparatus.

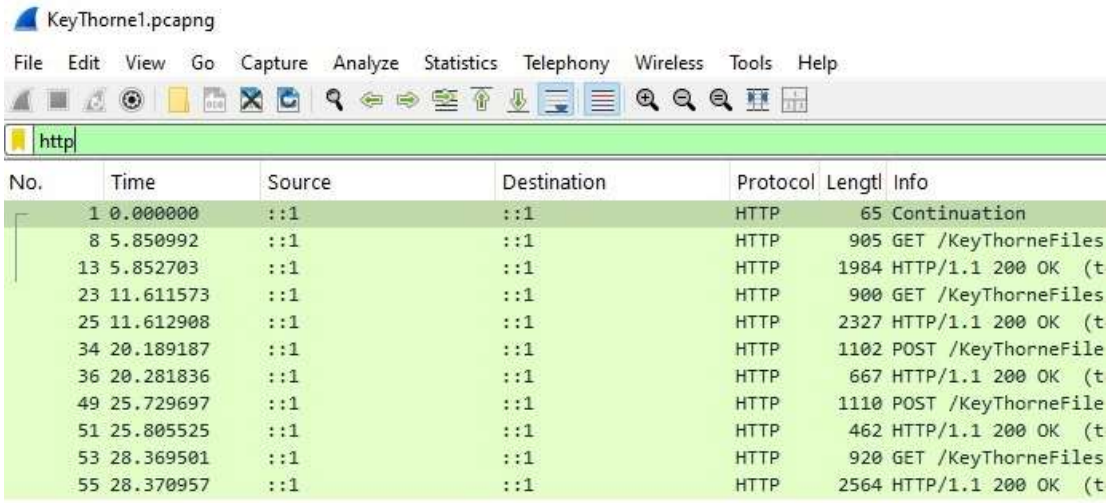


Fig. 4.15: Protocol Analyzer screen with filter 'HTTP' selected, showing the data exchange while the test sequence was typed.

The test suite was modified so that question 01 was repeatedly rendered upon each submission. Instead of the character string **'1234ABCD',** the 10-character sequence **'1234567890'** was used for the ease of keeping track of the number of keys typed. The sequence was repeatedly typed into the response panel and submitted. The number of characters was increased in logarithmic steps up to 10,000 characters. Meanwhile the communication was monitored on the protocol analyzer and all data captured on the protocol analyzer were saved for later analysis.

4.7.1.3 Testing of a lean keystroke logging system

During the analysis phase, examining the results of the protocol analysis of the previous experiment showed that the transmission of keystroke data by the keystroke logger used in the experiments makes inefficient use of the Hypertext Transfer Protocol (HTTP) payload. This motivated the development and testing of lean keystroke logging.

To understand the development of this lean keystroke logger, consider the following example dictionary entry in a JSON keystroke log file of the format that was used in the experiments.

{"key":"Q","timestamp":"1697397467326","type":"KeyDown"}

The three key-value pairs in this dictionary entry indicate that the key was 'Q', that the timestamp was 1697397467326 milliseconds, and that the event was a 'key down'. This entry consists of a total of 56 alphanumeric characters and are encoded in the American Standard Code for Information Interchange (ASCII) for transmission.

As far as keystroke logging relevant to this research is concerned, it is only necessary for the pressed key to be uniquely identified. There is no need for 'case sensitive' key

logging. However, when encoding the keys in ASCII to match the JSON format used at the back end, a single byte per character is nevertheless required.

To identify each key disregarding its case, a 1-byte code is not necessary. There are only 57 distinct keys used for text generation on a general purpose QWERTY keyboard including the modifier keys. The code depicted in Table 4.2 which is a 6-bit code largely based on the location of keys on the keyboard was devised to code these 57 keys, leaving an excess of 7 codes.

Table 4.2: 6-bit code to encode the distinct keys on a QWERTY keyboard

Key	Code	Key	Code	Key	Code
0	0	I	19	C	38
1	1	O	20	V	39
2	2	P	21	B	40
3	3	[	22	N	41
4	4	]	23	M	42
5	5	A	24	Comma	43
6	6	S	25	Period	44
7	7	D	26	/	45
8	8	F	27	Space	46
9	9	G	28	Backspace	47
-	10	H	29	Tab	48
=	11	J	30	Ctrl	49
Q	12	K	31	Alt	50
W	13	L	32	West	51
E	14	;	33	North	52
R	15	’	34	South	53
T	16	CR	35	East	54
Y	17	Z	36	Reverse accent	55
U	18	X	37	\	56

The timestamp (e.g. 1697397467326 in the example) expressed in milliseconds is a 13-digit decimal integer. Only 44 bits which rounds off to 6 bytes, are required to represent the largest 13-digit decimal integer, which is less than half the number of bytes required to code the timestamp as a string of ASCII characters. This number was further reduced by exploiting the range limits of the timestamp that would be encountered in practice during an online test, as follows.

Hypothetically, the worst case keystroke biometric that would be acquired during an online test would be the flight time of a 2-key sequence, with the first key typed as soon as the test commences and the second key typed just before the test ends. That would yield just two keystrokes giving a single flight time approximately equal to the total duration of the test. Therefore, the maximum flight time encountered in a test would be 3 hours or 10,800,000 milliseconds, being the typical maximum duration of a test. This extreme value can be represented by an 8-digit decimal integer.

A difference of 10,800,000 milliseconds would result from the difference between only the 9 least significant digits of the two timestamps. Therefore, it is sufficient to use a

truncated timestamp having only the 9 least significant digits of the 13-digit timestamp, which could be encoded by a 30-bit binary number.

Since the remaining keystroke data item which defines whether the event was a key press or a key release is in effect a Boolean variable, its coding required only a single bit.

Accordingly, it was possible to encode all the data related to a single keystroke using 37 bits (6+30+1). This was rounded off to 40 bits, being a whole number of bytes, since the HTTP payload accepts only bytes. Figure 4.16 illustrates the 5-byte data structure that was used to encode the keystroke data of a single key event.

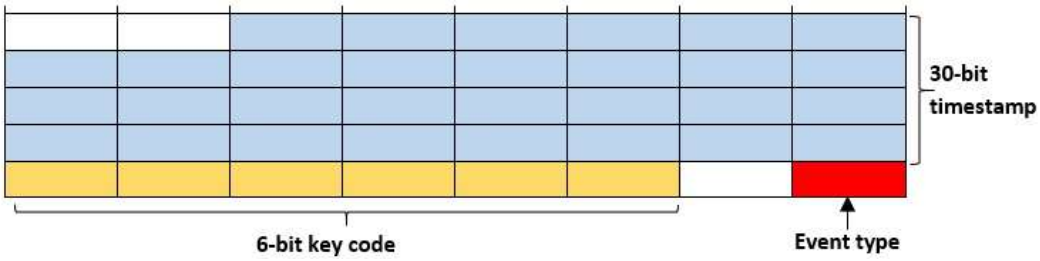


Fig. 4.16: 5-byte data structure used to encode a single key event

The following modifications were made to the ECMAScript used for keystroke logging in the online test suite, to make it lean.

i. Instead of identifying the typed keys using their ASCII code, the 6-bit code depicted in Table 4.2 was used and was initially represented by an integer.

ii. The type of the key event was identified using an integer, as '1' for the 'KeyDown' and as '0' for the 'KeyUp__'.

iii. The timestamp was truncated by taking only the least significant 9 digits and was converted to a 30-bit integer.

iv. The 3 integers representing the identity of the key, the event type, and the truncated timestamp were encoded onto the data structure shown in Figure 4.16 as 5 consecutive bytes.

v. The 5 bytes were transmitted to the receiving Hypertext Preprocessor (PHP) script at the back end as a 5-byte stream of raw data.

At the back end, The 5 bytes were logged in a 'byte file' instead of a JSON file.

The algorithm used in the modified ECMAScript for encoding a single key event is shown in Figure 4.17.

The modified keystroke logger was tested similar to the manner in which the conventional keystroke logger was tested, using the same test sequence '1234ABCD'.

The test described in 4.7.1.2 was repeated for the modified keystroke logger to ascertain the consistency of the lean keystroke logging method when a larger number of keys are typed, The same input sequence was used.

```
get key, timestamp, event_type from keystroke

N1 = N2 = N3 = 0
b[] ← Initialized byte array of 5 bytes

N1 = int(timestamp) mod 10^9
N2 =  6-bit_code_table_lookup(key)
if (event_type is 'KeyDown')
        N3 = 1
b[0] = N1 mod 256
b[1] = ((N1-  b[0]) mod 256^2)/256
b[2] = ((N1 – b[0] – b[1]*256) mod 256^3)/256^2
b[3] = ((N1 – b[0] – b[1]*256 – b[2]*256^2) mod 256^4)/256^3
b[4] = N2 * 2^2 + N3

return(b[])
```

Fig. 4.17: Algorithm for lean coding of a single key event

4.7.2 Examination of clipboard transfers during online test taking

As realized during the review of related work in Chapter 2, with the advent of Generative Artificial Intelligence (GenAI) and the widespread use of LLM among the student community, candidates with the will to cheat during online tests have gained additional options for cheating. It is possible for the candidate to consult an online LLM chatbot to generate plausible text based answers for a question and then transfer the generated content into the text box of the online test via the clipboard for submission. Unless the test administration has a way of detecting this clipboard transfer, there is no way to determine whether the submitted response was genuinely typed by the candidate out of his/her own thinking or whether it was copied. As mentioned in Chapter 2, this would be a special case of the fraudulent practice of 'lookup'.

Using an LLM as the source of lookup is more convenient and more efficient than looking up other common sources because less effort is required on the part of the cheater. At the minimum, the test question itself could be posed to the LLM. Moreover, the responses thus generated could be conveniently transferred to the response panel of the test using the clipboard. The cheating candidate reaps a twofold advantage over those who are taking the test genuinely, firstly due to the plagiarizing and secondly due to better time management.

A system aimed at maintaining learner integrity during online tests must essentially detect this practice. This provided the motivation for this experiment.

When using an online LLM tool, the candidate could invoke the tool on another tab on the browser used for taking the test or even on another browser. As long as the tool is invoked on the same device used for taking the test, it is possible for the candidate to

use the clipboard to perform the fraudulent text transfer. The clipboard is a tool provided by the operating system and by default, it works across applications.

To determine the effect of clipboard transfers on keystroke logging, a special version of 'Test 01' was developed where the same question is continuously rendered upon each submission so that various lookup scenarios could be easily tested out as many times as required.

Since the test suite for the experiments captures only the keystroke data and does not log the answers per se, a form handler was included in the special version, to capture the answers along with the keystrokes, as in an actual implementation. The form handler captures the text in the text box upon each pressing of the 'Submit' button and appends the text onto a text file residing on the server. An additional feature was included to log the timestamp as well alongside the submitted answer to enable the convenient reconciliation between the contents of the JSON file and the contents of the text file.

Since person dependence was not relevant in this experiment as well, no population was used and the researcher himself played the role of the candidate.

The test was rendered from a local host and invoked on a browser running on the same device. A separate tab was opened on the same browser to generate the text to be copied to the clipboard. The search bar itself of that tab was used to type the text to be copied and this simulated the copying of text from the response panel of an LLM chatbot. It was reckoned that for test purposes, all that was necessary was some source of text for copying which need not particularly be the output of an LLM chatbot.

The character sequence 'Text via clipboard' was typed into the search bar of the other tab and was copied onto the clipboard. The character sequence 'Text via keyboard' was used as the text directly typed into the answer box of the test. This keyboard input was applied and submitted first because the keystroke logging must have at least one keystroke typed for the JSON file to be created. For the subsequent posing of the question, the following actions were carried out in the given sequence.

i. The clipboard contents were pasted onto the answer text box by bringing up the context menu of the browser by right-clicking and then clicking on the 'paste' option. Next, the 'submit' button was clicked.
ii. The clipboard contents were pasted onto the answer text box using the keyboard shortcut 'CTRL + V', and the 'submit' button was clicked.

The above steps were replicated after switching the source of copied text to each of the following to verify the effect of different sources for copied text:

i. A text box in a web page opened on a different browser running on the same device.
ii. A text box of an online application running on the same device.
iii. A text editor running on the same device.

Both JSON logs and the answer text files were saved for analysis.

4.8 Choice of techniques and methods

In these experiments the methods and techniques used for feature extraction were statistical. This can be understood using the dwell time vector which was used throughout as the example feature vector. Its components are the mean dwell time of each key. Outlier elimination was done using the standard deviation. Both are statistical parameters.

Alternative techniques are finding increased use within this knowledge domain. Examples are machine learning applications. As an example relevant to this study, instead of extracting the mean dwell time of each key, it would have been possible to make a machine read the keystroke data directly and acquire understanding of the typing patterns. Patterns learnt of the registered candidate could be used for authentication during online tests where the keystroke data must exhibit patterns consistent with those learnt from the profile. Since there are many machine learning models, many options are available out of which the best one could be selected per each candidate.

The main reason that favored a statistical approach rather than machine learning approaches was the concerns surrounding the small size of datasets for both profile creation and for anomaly detection. Computational overhead was also a concern given the uninterruptible and inexorable nature of online tests. Operation of a learned machine during authentication is intuitively more resource intensive compared to statistical techniques.

Accordingly, machine learning approaches were not used in the extraction of the primary features. However, they were extensively used as the mode of classification on secondary features namely, the angle of departure and pause durations.

4.9 Resources used in the experiments

The entire test suite comprising the 04 tests used to capture biometrics as described in section 4.3.1 was developed using PHP.

To capture the roleplayers' keystrokes in the background, several keystroke loggers that were available in the public domain were initially tried out. However, these loggers failed to execute because they were being detected as 'malware' by the protection systems of the device used for the trial runs. Since this could happen even in an actual system based on the proposed concept and because it is not possible to require the candidates to turn off the protection systems on their devices, a solution was necessary to capture the roleplayers' keystrokes in the background and to transmit

them to the back end without being detected as malware. After studying several open source libraries used in similar applications, a keystroke logger of the 'form grabbing type' as described by G. Tubin [112] was specifically written using ECMAScript, for use in the experiments. This script executed well without being detected as malware.

The test suite was deployed on a cloud server[8] hosted by 'Amazon Web Services™'.

All the feature extractions, profile creation, and simulation of all scenarios described above were carried out on a DELL™ Inspiron Intel® core i7-7500U 2.7 GHz system using Ubuntu 22.04.3 LTS as its operating system and on an Ubuntu 22.04.3 virtual box playing on a VMWare workstation running on an HP™ 11th Gen Intel® Core i7-1165G7 2.80 GHz system using Microsoft™ Windows 10 professional version as its operating system.

All the methods used during the phases of profile creation, model development, and analysis were implemented using Python 3.8 scripts. Some early work connected with processing the biometric files used programs written in 'C' but these were later superseded by equivalent Python scripts. Linux shell scripts were used to automate the simulation of scenarios for all roleplayers as batches.

All profile data comprising the feature vectors, the datasets, matrices holding intermediate data, and analysis results were saved using both Comma-Separated Values (.CSV) as well as .NPY formats. Classifier models were saved using the .PK1 format.

LibreOffice Calc community edition was used for certain analysis performed on the .CSV files and for the generation of all data visualizations.

The WireShark Network Protocol Analyzer Version 4.4.0 (v4.4.0-0-g009a163470b5) was used for packet capture in the supplementary experiments. Apache server was used as the local host.

The web suite that was used for gathering of biometrics and the source code of all methods comprising Python scripts and the Linux shell scripts used in the experiments are deposited in the repository of 'additional material' accompanying this thesis.

The experimental apparatus could be replicated from the contents of the repository. 'ReadMe' files provide necessary instructions for setting up. The candidate profiles containing the features extracted from the biometrics, the classifiers, and the analytics are also included in the repository for the replication of the experiments at any level. The raw biometrics are not included. As per the conditions of the administrative and ethical clearance granted for this research, the biometrics of the roleplayers captured during the experiments shall remain the property of the university.

---

[8] Elastic IP address of the site was 52.52.162.59. The site was torn down in September 2023 after the data acquisition phase was terminated.

4.10 Recapitulation

In this chapter, all the experiments that were conducted during this research were described in detail along with the detailed explanation of the underlying principles. The experiments comprised 04 phases.

During the data acquisition phase, the keystroke dynamics of roleplayers who volunteered to be subjects in this experiment were captured. In the profile creation phase, the biometrics captured during Test 01 of the data acquisition phase were used to extract features and to create unique profiles for the roleplayers. During the detection model development phase training data were generated and classifier models were developed for each roleplayer. In the analysis phase, various scenarios of genuine test taking as well as fraudulent test taking were simulated offline and the results generated were recorded.

two additional experiments were conducted to supplement the results of the 04 main experiments.

The next chapter will deal with the interpretation of the results.

CHAPTER 5

INTERPRETATION OF RESULTS

5.1 Chapter description

This chapter provides a detailed interpretation of the results that were yielded by all the experiments that were described in the previous chapter.

5.2 Interpretation of the validity of the profiles created

5.2.1 Validity of the experimental subjects

Table 5.1 shows the completion of test taking by the enlisted roleplayers at the time the data acquisition phase was terminated and the usability of the biometrics as realized during the subsequent phases.

Table 5.1: Test completion by the roleplayers and usability of biometrics

	Test 01	Test 02	Test 03	Test 04
Number of roleplayers who completed tests	59	53	37	29
Number of roleplayers with usable biometrics	53	38	34	26

The lowest number of subjects for experimentation was for Test 04, which is 26. In the review of related work given in Chapter 2, experiments where 25 subjects had been used were encountered. Therefore, the size of the population used for each test could be regarded as being sufficiently representative and is acceptable.

Further, the roleplayers have been adequately briefed on the collection of their biometrics and their anonymity has been maintained. Therefore, the use of experimental subjects fully conforms to the terms of the ethical clearance granted for the research.

5.2.2 Validity of the mock test

There were 38 roleplayers with valid biometrics from Test 01.

Figure 5.1 shows the frequency distribution of these 38 roleplayers in respect of the size of the JSON file created for logging their keystroke data during Test 01. As seen from the figure, the distribution approximates a normal distribution with the mean, median, and modal classes coinciding. The distribution has a slight negative skew.

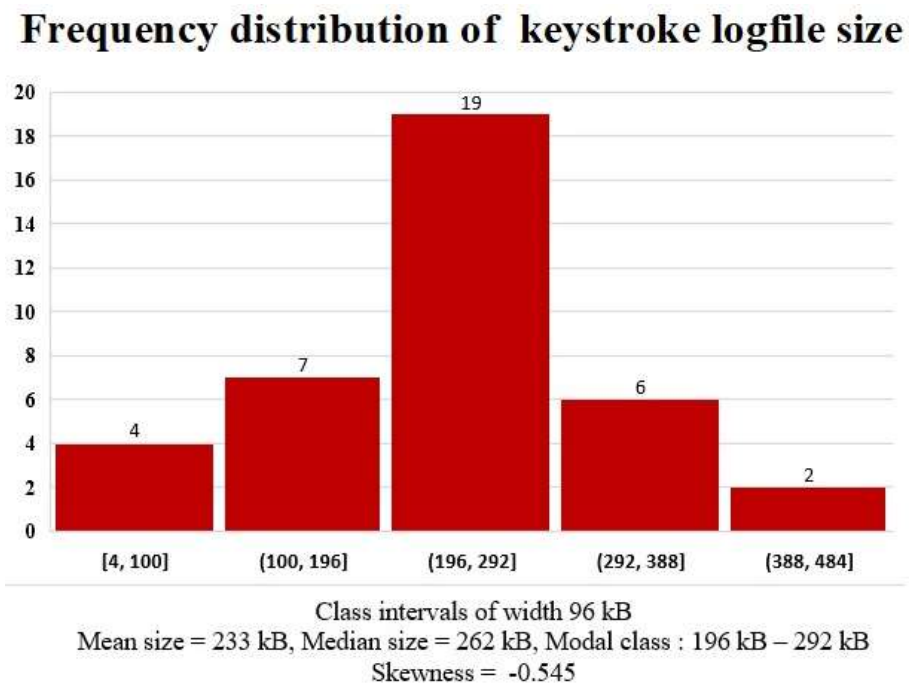


Fig. 5.1: Frequency distribution of the size of keystroke logfile across the population of 38 roleplayers

Considering the fact that the answers to questions in Test 01 required both fixed text (requiring a specific answer) and free text (where the roleplayer has the choice of words), it is evident that the above statistical distribution of the size of the JSON file results from the answers requiring free text. In a practical scenario, the length of free text answers are likely to be normally distributed and continuous authentication favors free text. Therefore, the biometrics that had been captured are consistent with that requirement.

5.2.3 Validity of the features extracted

It will be recalled that the concept suggested the extraction of 5 distinct features.

It was realized at the experimentation stage that the extraction of the 'quick sequence vector' and the 'far sequence vector' was not effective for the majority of roleplayers. Therefore, only the 3 remaining features namely the 'dwell time vector', 'hot sequence vector' and 'pause duration ensemble' were extracted.

The dwell time vector has been extracted successfully in respect of all 38 roleplayers with a constant size of 1275 bytes while the hot sequence vector has been extracted successfully with a fixed size of 5 kB for all 38 roleplayers. The file size of the two vectors is fixed because the vectors have a fixed structure. The ensembles of flight times of pauses which do not have a fixed structure have also been extracted successfully in respect of all 38 roleplayers. Figure 5.2 shows the frequency distribution of the 38 roleplayers in respect of the size of the file holding their pause duration ensembles.

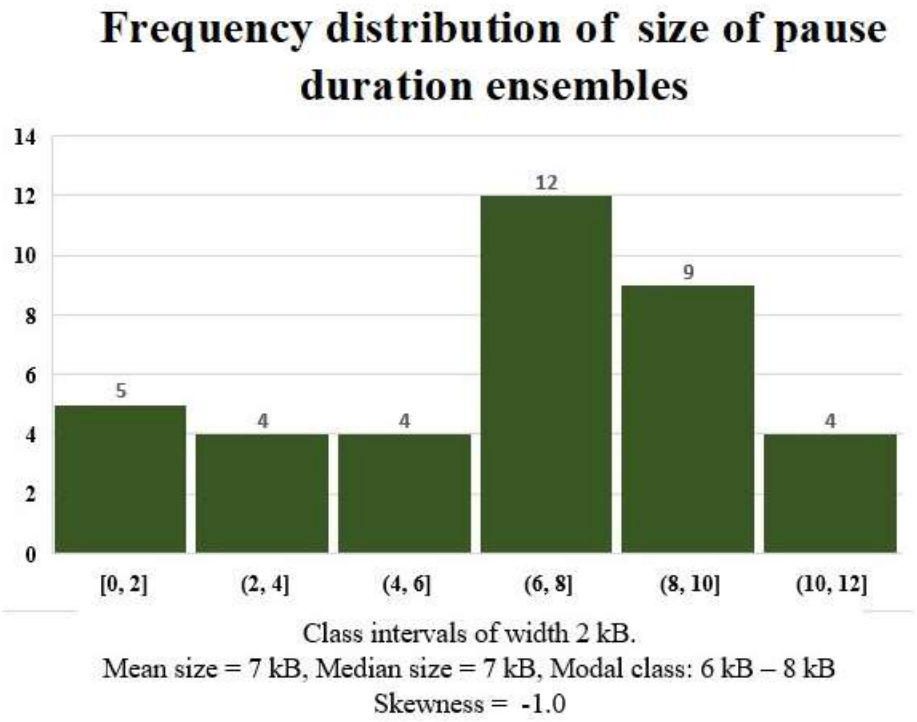


Fig. 5.2: Frequency distribution of the size of 'Pause Duration Ensemble' file across the population of 38 roleplayers

As seen from Figure 5.2, the size of the pause duration ensembles are quasi-normally distributed with the mean, median, and modal classes coinciding. It has a negative skew of -2.0. Since the occurrence of pauses during typing is extremely person dependent and would also depend on the text that are typed, the sizes of the pause duration ensembles exhibit this statistical distribution. The variations are evidently due to the pauses that occur during the typing of free text. Roleplayers who have opted to provide elaborate answers to the open ended questions in Test 01 have exhibited a higher number of pauses, resulting in larger sized ensembles.

Accordingly, all 3 features that were extracted could be regarded as being valid.

5.2.4 Validity of the training datasets

As described in Chapter 4, balanced datasets were created in respect of each of the 3 features using the simulations of Test 01.

Figure 5.3 shows the scatter plots of the training datasets based on the 'angle of departure between dwell time vectors' found in the profiles of 4 randomly selected roleplayers.

Figure 5.4 shows the scatter plots of the training datasets based on the 'angle of departure between hot sequence flight time vectors' in the profiles of 2 randomly selected roleplayers.

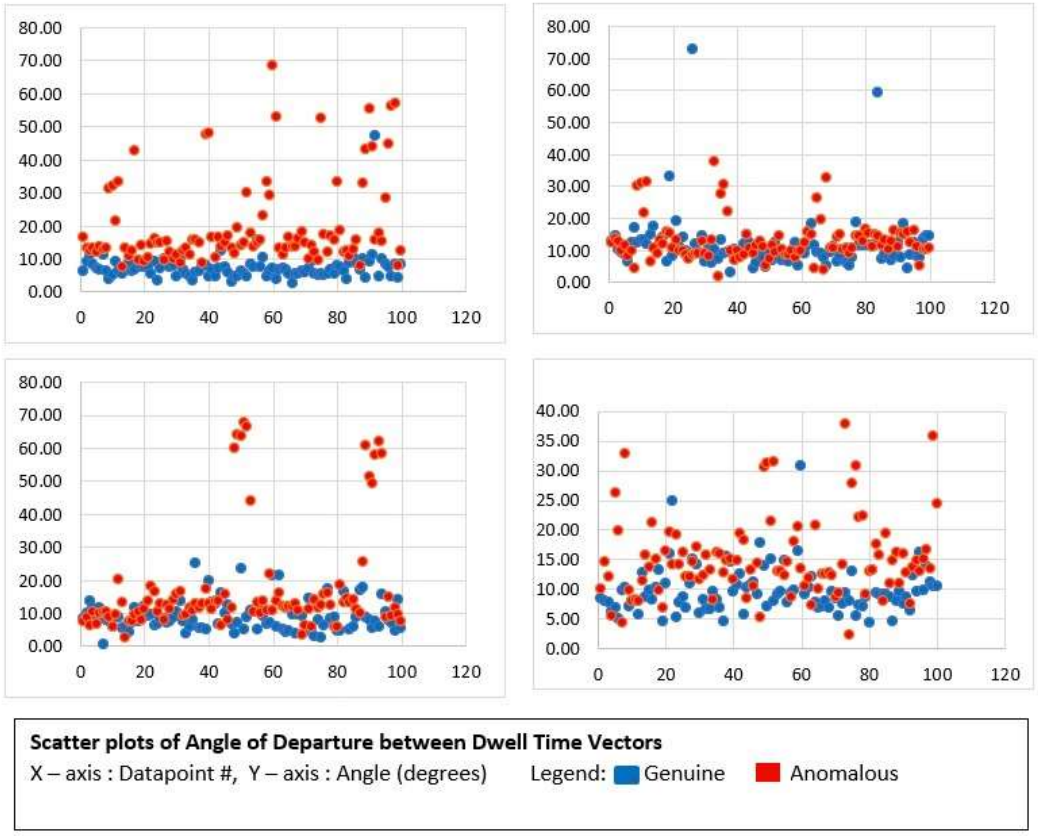


Fig. 5.3: Scatter plots of training datasets based on 'Angle of Departure' between Dwell Time Vectors.
(Source: Profiles of 4 randomly selected roleplayers)

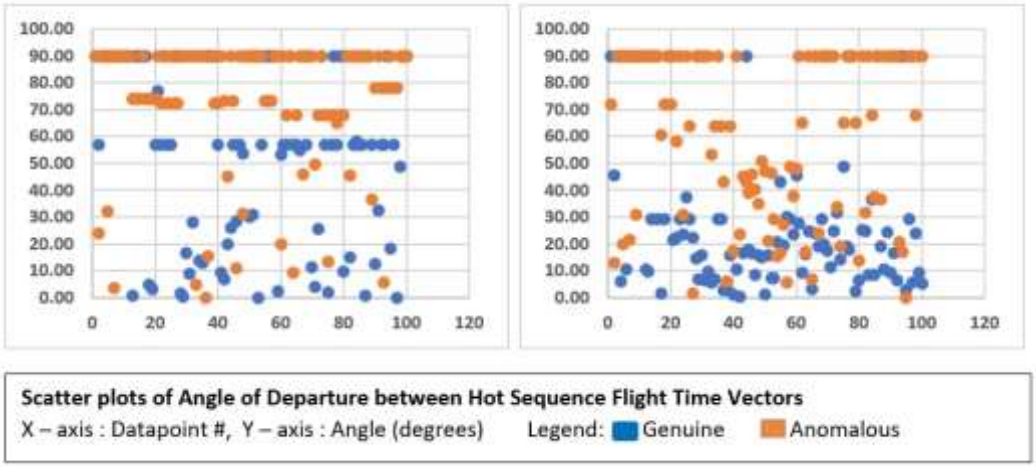


Fig. 5.4: Scatter plots of training datasets based on 'Angle of Departure' between Hot Sequence Flight Time Vectors.
(Source: Profiles of 2 randomly selected roleplayers)

Figure 5.5 shows the scatter plots of the training datasets based on the 'ensembles of pause durations' found in the profiles of 4 randomly selected roleplayers

.

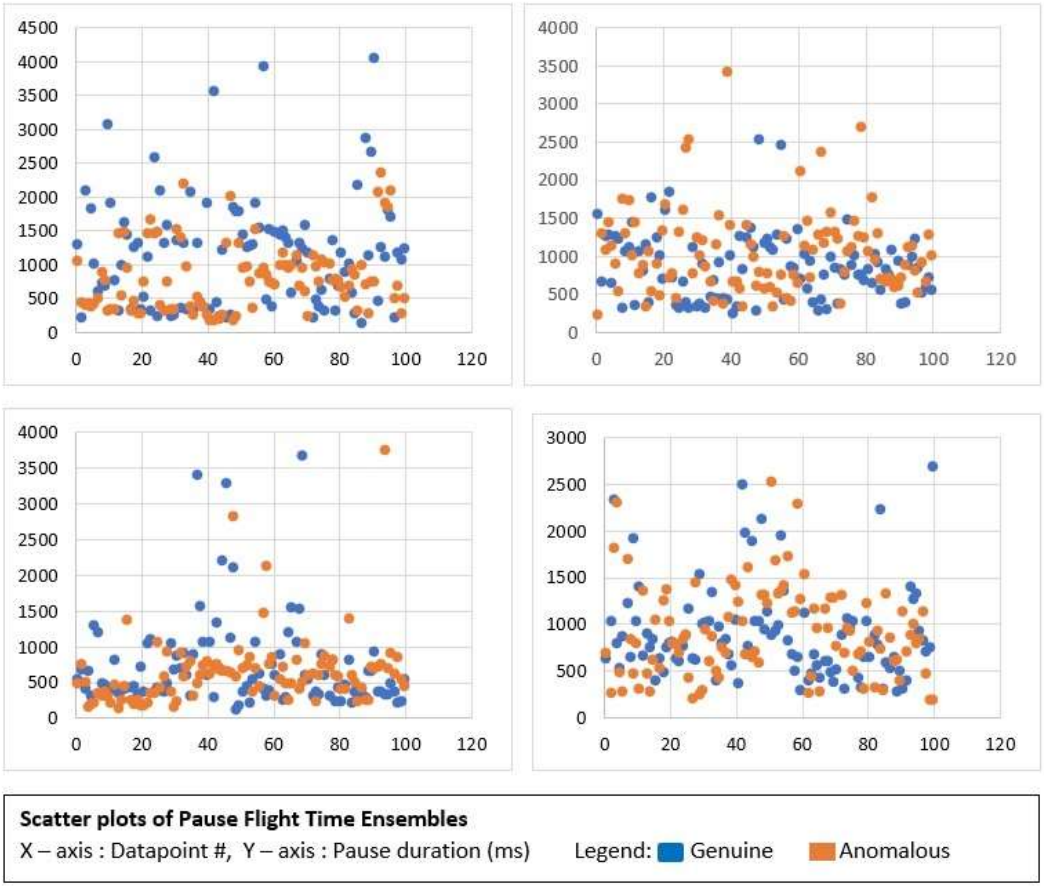


Fig. 5.5: Scatter plots of training datasets based on 'Pause Duration Ensembles'. (Source: Profiles of 4 randomly selected roleplayers)

Since the other three variants of the datasets were formed using linear transformations of the basic data, they would inherit the validity of the original dataset and there is no need to examine them separately.

The effectiveness of the classifiers trained using these datasets is yet to be seen.

5.2.5 Validity of selection of the best classifiers

4 types of binary classifiers were used as explained in Chapter 4.

Each of the 3 features had 4 variants of training data. A classifier of each of the 4 types was trained using each variant of training data. Accordingly, there were 16 distinct classifiers for each feature as listed in Table 5.2.

Table 5.2: Formation of classifiers for a single feature

Classifier #	Type	Training data variant
1	LR	Basic dataset
2	LR	Basic dataset with outliers replaced
3	LR	Basic dataset normalized
4	LR	Basic dataset normalized and outliers replaced
5	SVM	Basic dataset
6	SVM	Basic dataset with outliers replaced
7	SVM	Basic dataset normalized
8	SVM	Basic dataset normalized and outliers replaced
9	RF	Basic dataset
10	RF	Basic dataset with outliers replaced
11	RF	Basic dataset normalized
12	RF	Basic dataset normalized and outliers replaced
13	NB	Basic dataset
14	NB	Basic dataset with outliers replaced
15	NB	Basic dataset normalized
16	NB	Basic dataset normalized and outliers replaced

For each feature, the best classifier was selected by simulating the taking of Test 01 monitored by each of the 16 classifiers in turn, and identifying the classifier that is 'least discriminating' towards the roleplayer. The rationale for selecting the least discriminating classifier was explained in Chapter 4. Table 5.3 shows the number of candidates for whom each classifier has been identified as the best.

As seen from this table, classifier No.9, the 'Random Forest binary classifier' trained on the basic dataset is the modal best classifier for the feature 'angle of departure between dwell time vectors'. Classifier No.13, the 'Naive Bayes binary classifier' trained on the basic dataset comes next. 6 classifiers have been selected as the best in respect of few roleplayers with the remaining 8 classifiers not been selected as the best for any roleplayer.

Classifier No.4, the 'Logistic regression binary classifier' trained on the basic dataset normalized and outliers replaced is the modal best classifier for the feature 'pause durations', becoming the best classifier for 10 roleplayers. Both classifiers No.1 and No.2, being 'Logistic regression classifiers' trained on the basic dataset and outlier replaced basic dataset respectively, share a score of 6. There are only 4 classifiers which have not emerged as best in respect of any roleplayer while the remaining 9 classifiers emerge as best in respect of 3 roleplayers at most.

Table 5.3: Number of roleplayers for whom each classifier has been identified as the best

Classifier #	Feature		
	Angle of departure for dwell time	Flight time of pause durations	Angle of departure for flight time of hot sequences
1	0	6	0
2	2	6	0
3	0	3	0
4	3	10	0
5	1	2	0
6	2	2	0
7	0	0	0
8	0	2	0
9	15	0	0
10	0	1	0
11	2	1	0
12	0	0	0
13	12	3	38
14	0	1	0
15	1	0	0
16	0	1	0

It is also noted that classifier No.7, the 'Support Vector Machine binary classifier' trained on the normalized basic dataset has not emerged as a best classifier in respect of any roleplayer for any of the 3 features. The above observations are not sufficiently conclusive to reject any classifier as unsuitable. What is noteworthy is that all classifiers have some probability of emerging as a best classifier when the population is large.

Classifiers trained on each of all 4 variants of data have emerged as the best machine in respect of at least one feature. This shows that using variants of the dataset in addition to the basic dataset has been a good strategy.

Therefore, it may be concluded that testing out several types of classifiers trained on variants of datasets and selecting the best one out of them is a sound strategy.

5.2.6 Overall validity of profiles created

Having considered the validity of the mock test, the feature extraction, and the selection of best classifier, it may be concluded that the profile creations done in the experiments are valid and that they are consistent with the profile creation envisioned in the concept. The effectiveness of the training datasets remains to be ascertained.

5.3 Interpretation of results of the simulation of genuine test taking

In the experiments, genuine test taking was simulated by each roleplayer taking Test 02 being monitored using the roleplayer's best classifier.

5.3.1 Accuracy of genuine detections

Figure 5.6 depicts the distribution of the False Rejection Rate (FRR), across the population of 38 roleplayers.

As seen from the figure, all 38 roleplayers exhibit an FRR less than 20%, 36 of them exhibit an FRR less than 15%, 33 of them exhibit an FRR less than 10%, and 29 of them exhibit an FRR less than 5%, giving a distribution highly skewed to the right.

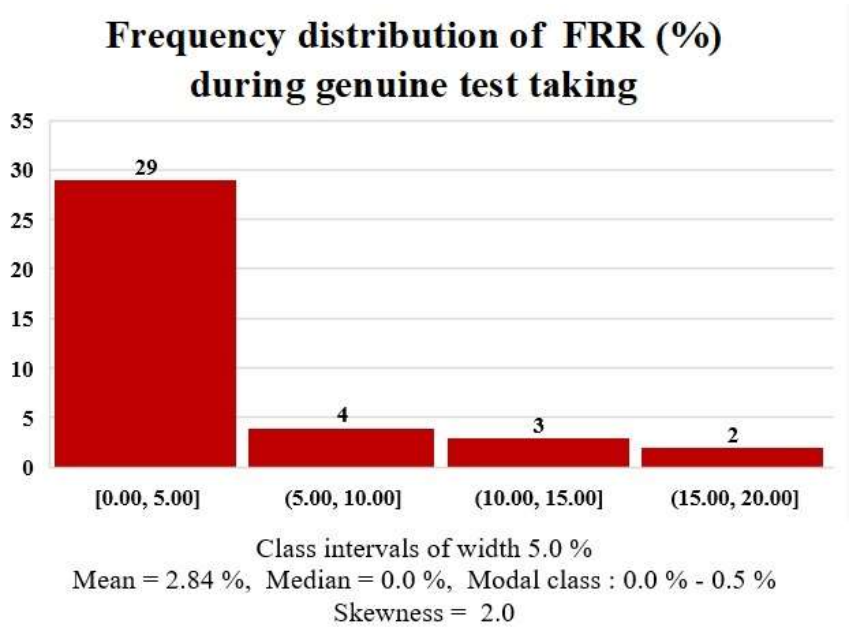


Fig. 5.6: Frequency distribution of FRR (%) across the population of roleplayers

5.3.2 False Rejection Worst Interval (FRWI) for genuine test taking

Figure 5.7 depicts the distribution of the False Rejection Worst Interval (FRWI), the longest duration a genuine candidate remained detected erroneously as an imposter or a fraudulent test taker, across the population of 38 roleplayers.

As seen from the figure, not more than 10 roleplayers had to remain classified as fraudulent for more than 1 minute. There are 7 more roleplayers who remained classed erroneously as fraudulent for 3 minutes or less. For the 3 remaining roleplayers, the FRWI is at most 7 minutes.

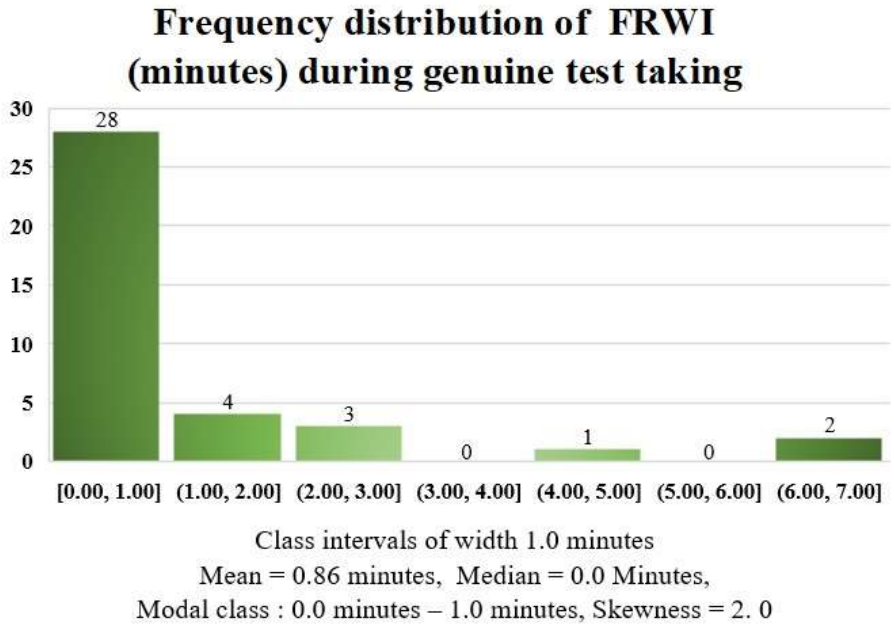


Fig. 5.7: Frequency distribution of the FRWI across the population of roleplayers

5.3.3 Validity of non-temporal features

The non-temporal features were also extracted during the experiment with the possibility of exploiting them to enhance the proposed authentication system.

As seen from the comparison between non-temporal features extracted from Tests 01 and 02, of 10 randomly selected roleplayers given in Table 5.4, there is a significant difference between the non-temporal features of the same roleplayer.

When typing in response to questions during a test, there is an attempt by the candidate to be more accurate than at other times. The candidate may look over what has been typed and make amendments before submitting the answer. The typing may also be more cautious so as to avoid mistakes in the first instance. The inconsistency in the non-temporal features could be the net result of all these factors. Therefore, non-temporal

features do not appear to be suitable for use as conclusive metrics for anomaly detection during online tests.

Table 5.4: Comparison of non-temporal features extracted from Tests 01 and 02 (Source: 10 randomly selected roleplayers)

Non-temporal features from Test 01			Non-temporal features from Test 02		
Raw typing speed (cpm)	Effective typing speed (cpm)	Typing error rate (%)	Raw typing speed (cpm)	Effective typing speed (cpm)	Typing error rate (%)
234.76	225.96	3.75	118.34	113.04	4.48
106.43	102.22	3.95	97.62	89.67	8.15
22.52	16.89	25.00	66.54	64.98	2.35
121.98	120.13	1.52	60.73	58.57	3.56
140.43	135.71	3.36	87.52	84.73	3.19
131.44	125.70	4.37	81.27	78.68	3.19
56.19	54.40	3.18	107.56	102.18	5.00
5.20	5.19	0.03	132.97	132.97	0.00
92.57	87.76	5.19	107.78	102.02	5.34
161.62	152.67	5.54	101.48	95.99	5.41

5.3.4 Overall interpretation of results of the simulation of genuine test taking

When the FRR and the FRWI are considered jointly, very low FRR means that the rejection of genuine test takers is extremely infrequent and that in the rare occasions of such detection occurring, it persists only for a very short time due to subsequent continuous authentications detecting the test taking again as being genuine.

This indicates that the continuous authentication system used in the experiments is extremely accurate as far as non-rejection of genuine test taking is concerned.

5.4 Interpretation of results of the simulation of impersonation

The impersonation of each roleplayer was simulated by taking all other roleplayers in turn as imposters while Test 02 was run. Accordingly, there were 1406 (38*37) simulations of impersonation.

With the results of 37 simulated impersonations for each of the 38 roleplayers in hand, it is pertinent to examine the observed performance metrics in two ways. The first would be to examine the best performance. The best performance in respect of different metrics for a given roleplayer may generally be in respect of different imposters. Nevertheless, the best performance would be a direct indication of the system's capability.

Table 5.5: Best performance metrics of simulated impersonations by 37 imposters

	1	2	3	4	5	6	7	8	9	10	11	12	13
T2FD (min)	0.029	0.092	0.104	0.092	0.029	0.029	0.029	0.029	0.029	0.029	0.092	0.029	0.029
Accuracy (%)	98.95	96.15	84.00	79.81	95.65	92.77	0.00	95.79	98.08	95.19	78.00	94.74	88.42
FAWI (min)	0.096	0.416	0.576	0.767	0.438	0.350	0.000	0.311	0.138	0.333	0.563	0.350	0.350

	14	15	16	17	18	19	20	21	22	23	24	25	26
T2FD (ms)	1756	5533	5533	1756	3933	5533	1756	1756	1756	1756	5533	5533	1756
Accuracy	80.00	98.13	98.85	96.84	98.00	99.00	89.66	87.37	84.34	99.04	98.11	98.08	95.18
FAWI(ms)	21895	3459	8111	13680	6928	5628	60170	20974	47102	3337	4547	7932	14334

	27	28	29	30	31	32	33	34	35	36	37	38
T2FD (min)	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029
Accuracy (%)	99.02	95.79	92.63	91.58	95.79	97.59	98.95	98.13	95.79	0.00	98.13	0.00
FAWI (min)	0.096	0.169	0.240	0.274	0.291	0.077	0.096	0.107	0.235	0.000	0.138	0.000

T2FD – Time to First Detection , FAWI – False Acceptance Worst Interval

Table 5.6: Average values of performance metrics of simulated impersonations by 37 imposters

	1	2	3	4	5	6	7	8	9	10	11	12	13
TotalChks	90	89	88	90	90	89	90	90	91	88	90	90	90
RubOuts	246	253	243	252	255	250	257	256	257	235	253	250	255
TotalImpDets	72	59	54	35	61	68	90	70	65	68	25	40	32
T2FD (min)	0.288	0.394	0.497	0.881	0.349	0.328	0.227	0.263	0.526	0.347	1.223	0.621	1.368
Accuracy %	78.89	62.34	60.34	36.67	67.05	73.54	97.62	76.32	69.12	74.58	27.15	43.76	35.14
FAWI(min)	1.452	3.626	3.414	7.013	2.551	2.096	0.000	1.784	3.370	2.201	8.781	4.732	5.734
Session(min)	34.726	34.251	34.228	34.887	34.788	34.389	34.333	34.157	34.770	34.099	35.173	35.149	32.166

	14	15	16	17	18	19	20	21	22	23	24	25	26
TotalChks	90	91	90	89	90	90	90	89	90	90	91	91	90
RubOuts	252	259	259	246	250	255	251	252	248	250	250	257	254
TotalImpDets	18	57	65	61	66	69	50	37	45	85	76	64	45
T2FD (min)	1.426	0.398	0.423	0.284	0.432	0.497	1.198	0.546	1.662	0.241	0.309	0.533	0.594
Accuracy %	20.66	60.63	70.43	67.95	71.34	74.05	54.60	42.50	47.59	92.64	81.77	67.87	47.97
FAWI(min)	15.807	3.723	2.158	2.598	2.003	2.495	3.587	5.171	5.437	0.859	2.198	3.047	4.189
Session(min)	34.163	35.042	34.477	34.566	34.491	34.964	35.106	34.892	34.362	34.786	34.968	34.841	34.662

	27	28	29	30	31	32	33	34	35	36	37	38
TotalChks	90	91	90	91	90	90	90	90	90	90	91	92
RubOuts	259	257	253	256	246	248	246	253	255	253	254	258
TotalImpDets	77	74	47	56	64	77	72	72	74	90	76	92
T2FD (min)	0.238	0.232	0.546	0.565	0.295	0.262	0.288	0.250	0.301	0.229	0.317	0.222
Accuracy %	83.25	79.81	50.95	60.85	70.53	84.62	78.89	77.06	80.56	97.62	81.19	97.62
FAWI(min)	1.402	1.368	3.445	3.597	2.514	1.023	1.452	1.443	2.451	0.000	1.509	0.000
Session(min)	34.998	35.265	34.488	35.212	34.726	34.675	34.726	34.307	35.030	35.006	34.987	35.214

TotalImpDets – Total Imposter Detections

Table 5.5 indicates the best values of three (03) selected performance metrics in respect of each roleplayer picked from the 37 simulations of impersonation.

The second would be to examine the average performance where the performance of each metric is averaged over the 37 imposters. The average values would differ from the best values and may also be colored to various extents by outlier behavior.

Table 5.6 indicates the average values of the performance metrics for each roleplayer. The numbering of the roleplayer in both tables is purely for distinction and the numbers have no correlation with their user ID used in the experiments.

It could be noted that the average duration of the test sessions and the average of the number of checks carried out during the sessions display a high degree of invariance. This could be examined further by calculating the $\mu \pm \sigma$ limits of these two metrics.

Table 5.7 indicates the variance of the 'average values' of the session duration and the number of checks during each session.

Table 5.7: The variance of average test session parameters across the population of 38 roleplayers, when impersonated by 37 roleplayers

	Mean	**Standard deviation**	**$\mu \pm \sigma$ Confidence interval**
Session duration (minutes)	34.66	0.53	[34.13, 35.19]
Total number of checks carried out during the session	90	0.72 (≈ 1)	[89,91]

The number of checks carried out during a session refers to the number of 40 key event sequences captured during the session. In a simulated run, both session duration and number of checks will be governed by the biometrics of the roleplayer playing the role of 'imposter'. For a given roleplayer, when the other roleplayers are taken in turn as imposters and the taking of Test 02 is simulated, each simulation will use a different set of biometrics. The average values shown in Table 5.6 are the average values of each metric computed in each of the imposter simulations. What Table 5.7 indicates is the distribution of the 'mean values' of two of these metrics across the population of 38 roleplayers.

The session durations averaging around 35 minutes and the number of checks carried out during a session also averaging to 90, both with an extremely low variance suggests that the time taken to complete Test 02 and the number of characters typed in response to the test during the test session has largely been independent of the roleplayers. Considering the fact that roleplayers had been selected by two intermediaries independently through convenience sampling, the figures in Table 5.7 show that the roleplayer selection has yielded a statistically sound population.

5.4.1 Accuracy of imposter detection

Figure 5.8 depicts the distribution of the minimum False Acceptance Rate (FAR) observed for each roleplayer when impersonated by each of the other 37 roleplayers in turn. They represent the best performance of imposter detection.

It is seen that 24 roleplayers have exhibited an FAR less than 5.96% while being impersonated by at least one (01) imposter. Also when the 3 outliers are disregarded, the mean FAR of the best performance is 6.18% which corresponds to a detection accuracy of 93.82%. This in fact, is the capability of the system when detecting impersonation.

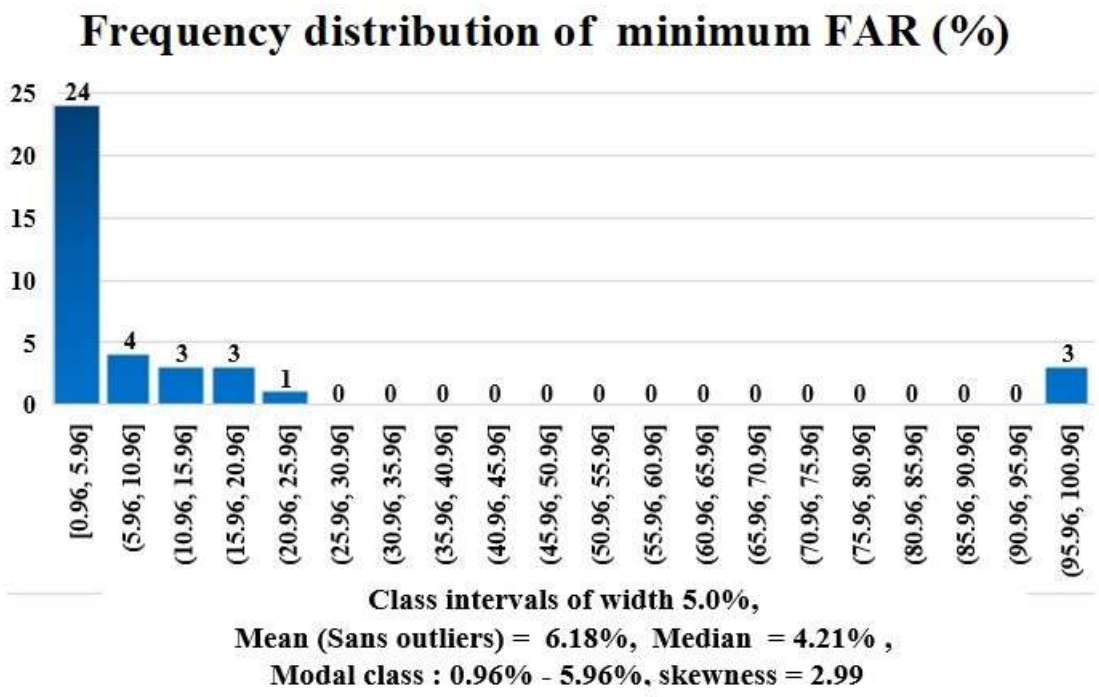


Fig. 5.8: Frequency distribution of Minimum FAR (best performance) across the population of roleplayers

Figure 5.9 depicts how the average FAR has been distributed across the population of 38 roleplayers, when each of them were impersonated in turn by the remaining 37 roleplayers. It is seen that the average FAR exhibits a normal distribution around a mean value of 33.0% with a positive skew. There are 23 roleplayers whose average imposter detection accuracy is greater than 68.0 % while there are 28 roleplayers with detection accuracy greater than 58.0 %.

This shows that the average imposter detection capability is less than the best performance, yet appreciably high. There is considerable variation in the detection capability when impersonated by different players.

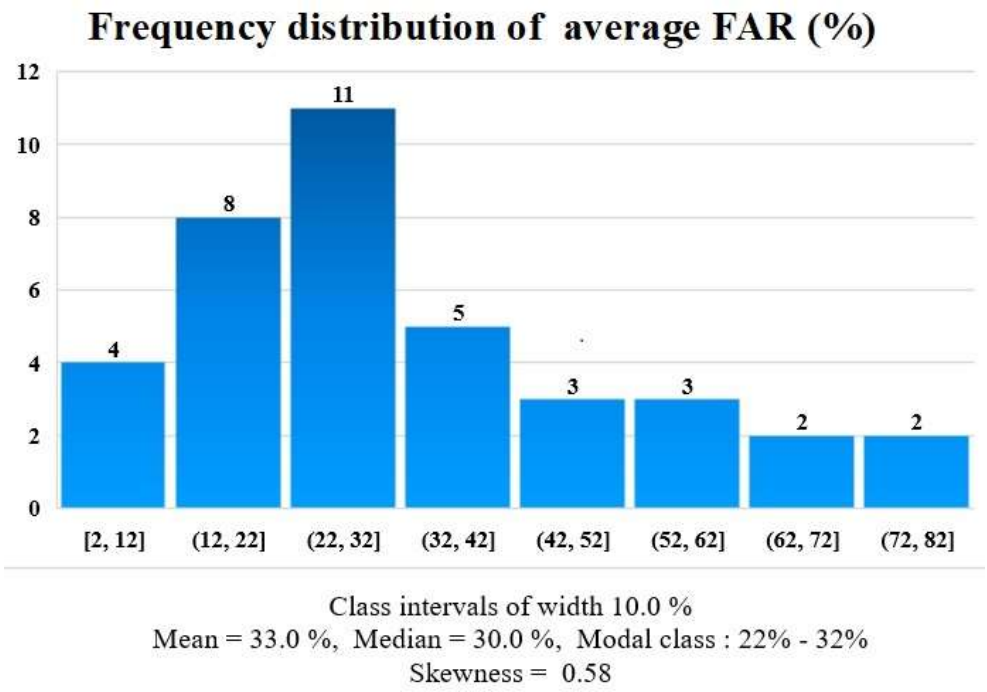


Fig. 5.9: Frequency distribution of average FAR across the population of roleplayers

Therefore it may be deduced from these figures that the system has good imposter detection capability which is imposter dependent for reasons not evident at this stage.

5.4.2 False Acceptance Worst Interval (FAWI) during impersonation

Figure 5.10 depicts the distribution of average 'False Acceptance Worst Interval' displayed for each of the 38 roleplayers when impersonated in turn by the other 37 roleplayers.

The average FAWI exhibits a normal distribution around a mean value of 3 minutes with a high positive skew. On the average, in the case of 30 roleplayers, imposters cannot evade detection for longer than 4 minutes. Imposters cannot evade detection for longer than 6 minutes in the case of 35 roleplayers.

There are 3 outliers, the most right flung case being an FAWI of 16 minutes and the other two being 10 minutes and 8 minutes.

It is seen from Table 5.6 that the FAWI is zero in the case of 3 roleplayers. Since the FAWI is essentially a positive quantity ($FAWI_j \geq 0 \quad \forall\ j$), the average FAWI can be zero if and only if all individual FAWI values themselves are zero. This means that in the cases of these 3 roleplayers, impersonation by all other players have yielded a null FAWI. As per the algorithm used for the simulation of taking Test 02, FAWI = 0 can happen only in the case of two eventualities.

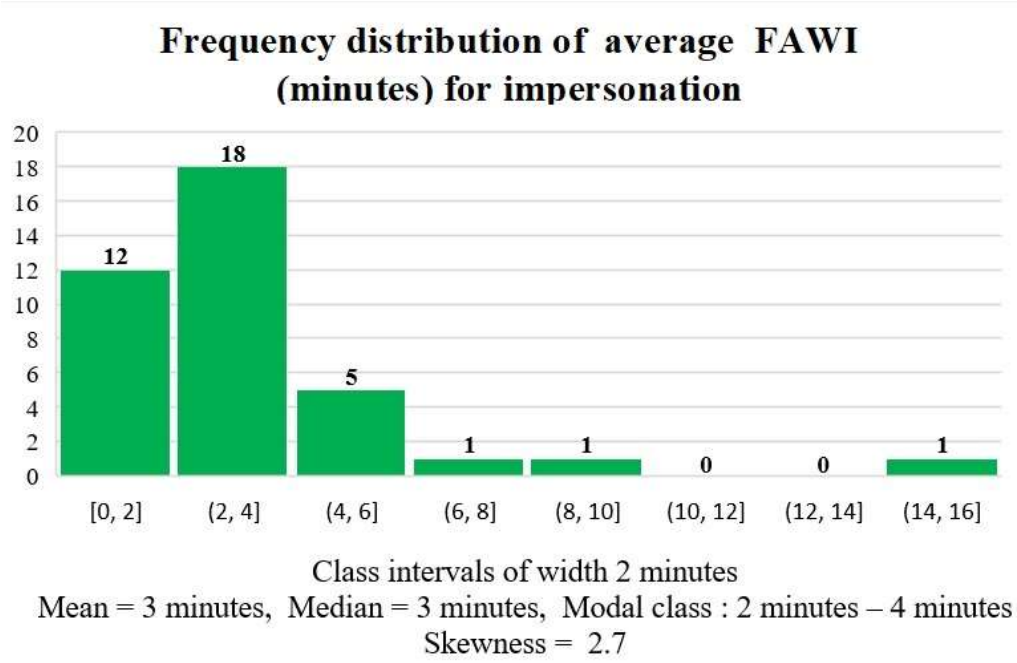


Fig. 5.10: Frequency distribution of average worst case False Acceptance Interval (in minutes) when roleplayers impersonate each other

i. The imposter been detected throughout the session at all checks from beginning to end, in which case false acceptance does not arise.

ii. There has not been a single occurrence of two or more consecutive imposter detections. Whenever an impersonation has been missed (falsely accepted as genuine), it has been redetected at the immediately following check, in which case the program records the False Acceptance Interval as zero.

In the first eventuality, the detection accuracy should essentially be 100%. In the second, it should be 50% for the worst case where detections alternate. Table 5.6 indicates that in the cases of all 3 roleplayers whose average FAWI is zero, the average detection accuracy is 97.62% which points to consistent imposter detections with a few isolated misses in between.

5.4.3 Time to First Imposter Detection (T2FD) during impersonation

Figure 5.11 depicts the distribution of average time for the first imposter detection since the beginning of the test session. It is seen that the distribution is off-normal with neither the mean nor the median being included in the modal class. However, the distribution is extremely skewed to the right, implying that low values of the time to first detection have a high occurrence.

.

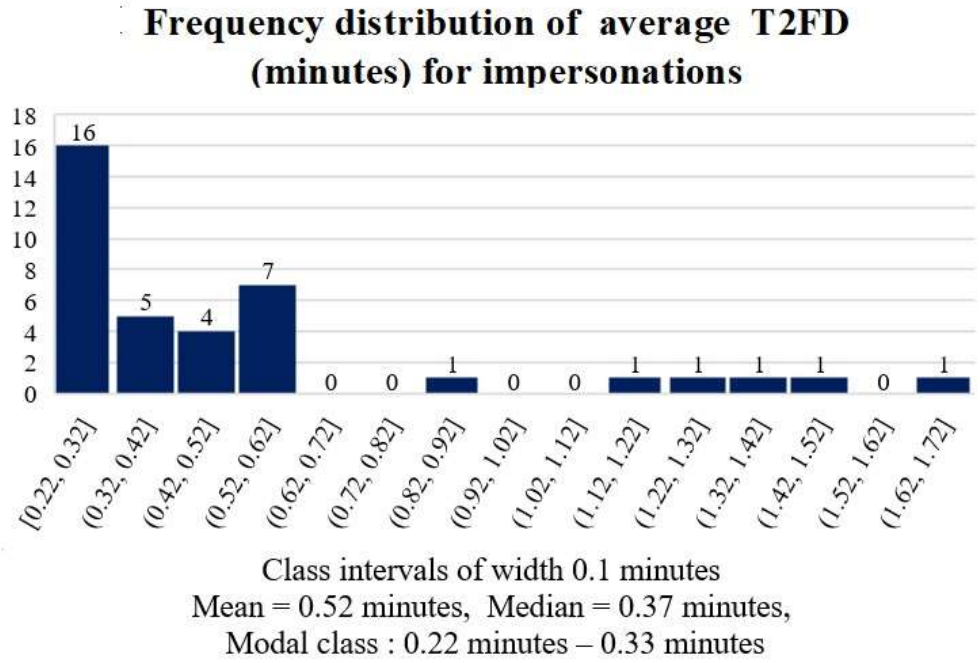


Fig. 5.11: Frequency distribution of T2FD (minutes) when roleplayers impersonate each other

When the 5 outliers are disregarded, it is seen that not a single imposter has evaded detection since the beginning of the test session for longer than 1 minute while 32 imposters have been detected before the lapse of approximately 37 seconds. These figures too indicate very effective imposter detection capability.

5.4.4 Unusability of non-temporal metrics in imposter detection

Since it was realized during the interpretation of results of the simulations of genuine test taking that the non-temporal metrics of even the same roleplayer are poorly correlated between Test 01 and Test 02, the correlation (if any) between non-temporal metrics of two roleplayers cannot be confidently reckoned as an indication of impersonation. Therefore, no attempt is made to interpret the non-temporal metrics during simulated impersonations.

5.4.5 Overall interpretation of results of the simulation of impersonated test taking

When the very low average values FAWI and the time to the first detection are considered together with the extremely low values of minimum FAR and the appreciably low average values of FAR, it is seen that the majority of impersonations had been detected promptly, consistently, and with a high accuracy. The best performances was found to be extremely high. Accordingly, the detection capability

of the anomaly detection system used in the experiments is sound and dependable as far as impersonation is concerned.

5.5 Interpretation of results of the simulation of lookup

There were 34 roleplayers with valid biometrics who had completed Test 03 that dealt with the particular fraudulent practice of 'lookup'.

5.5.1 Accuracy of detecting lookup

The performance metrics of taking of Test 03 by the 34 roleplayers are shown in Table 5.8. In this case too, the number given to each roleplayer is for distinction only and has no relation to the user ID of the roleplayer used in the experiments.

Figure 5.12 depicts the distribution of FAR across the 34 roleplayers.

Keeping in mind that the taking of Test 03 by each of the 34 roleplayers is a known case of fraudulent test taking where the roleplayer is answering while looking up, it follows that every instance of the roleplayer being detected as 'genuine' would amount to a false acceptance. Accordingly, for the authentication system to be deemed as effective, the FAR should be low. It will be noted that in the case of 8 roleplayers, the FAR has been 100% meaning that the authentication system has not worked at all in respect of them. Therefore, FAWI in respect of them should be the session duration itself while the T2FD is irrelevant.

Table 5.8: Performance metrics of test taking by 34 roleplayers while looking up

	FAR (%)	T2FD (minutes)	FAWI (minutes)		FAR (%)	T2FD (minutes)	FAWI (minutes)
1	60.00	0.22	4.12	**18**	12.00	0.11	0.36
2	39.00	0.10	1.88	**19**	52.00	0.19	0.64
3	84.00	0.26	10.23	**20**	59.00	0.39	1.09
4	47.00	0.56	3.66	**21**	14.00	0.10	1.24
5	37.00	0.13	2.13	**22**	23.00	0.13	1.96
6	0.00	0.87	0.00	**23**	43.00	0.79	0.80
7	24.00	0.24	1.29	**24**	76.00	0.52	5.12
8	80.00	0.56	6.41	**25**	53.00	0.79	2.60
9	33.00	0.13	3.14	**26**	36.00	0.19	13.50
10	88.00	1.18	5.86	**27**	28.00	0.43	0.85
11	63.00	0.17	1.34	**28**	0.00	3.15	0.00
12	38.00	2.73	11.88	**29**	100.00	-	36.84
13	83.00	1.31	11.60	**30**	100.00	-	63.77

	FAR (%)	T2FD (minutes)	FAWI (minutes)		FAR (%)	T2FD (minutes)	FAWI (minutes)
14	33.00	0.24	2.53	**31**	100.00	-	44.01
15	100.00	-	31.03	**32**	100.00	-	19.82
16	100.00	-	125.72	**33**	100.00	-	11.08
17	47.00	0.06	1.83	**34**	100.00	-	1.23

While the authentication system failing in respect of 8 roleplayers raises some questions about its reliability, the FAR being zero in respect of one roleplayer is noteworthy. Several other roleplayers also have recorded low FAR figures.

Careful examination of the frequency distribution shows that there are two central tendencies visible, suggesting that there are possibly two independent phenomena at work simultaneously. One phenomenon appears to interfere with the anomaly detection negatively, causing the FAR to rise. The other phenomenon appears to cause the anomaly detection accuracy approach a quasi-normal distribution of relatively low FAR values.

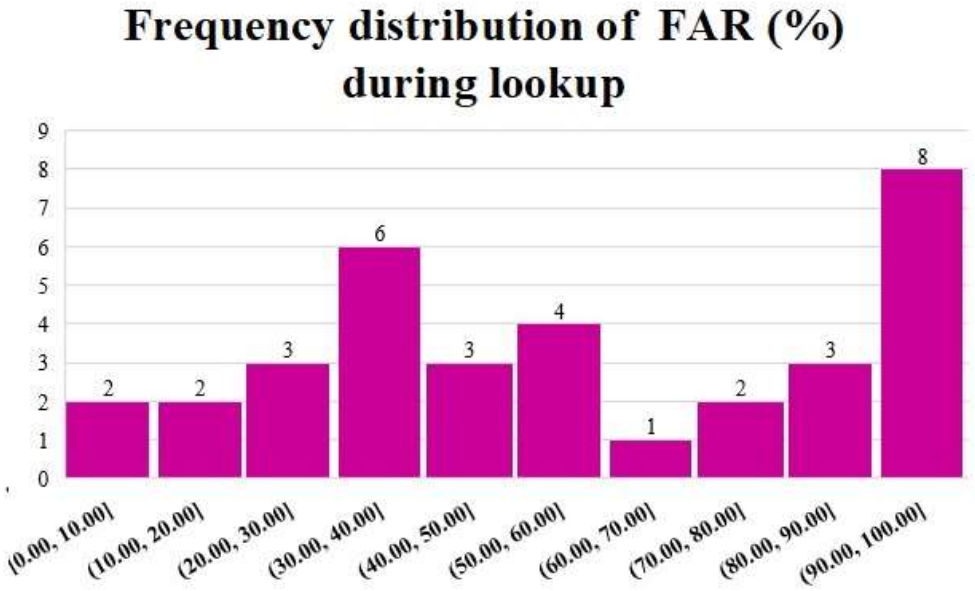


Fig. 5.12: Frequency Distribution of FAR across 34 roleplayers when they take online test while looking up.

There are 14 roleplayers with FAR greater than 60% which corresponds to a detection accuracy less than 40% which includes the 8 roleplayers whose FAR is 100%. They are the roleplayers whose test taking appears to be predominantly affected by the first phenomenon. The balance 14 roleplayers are those whose test taking appears to be influenced more by the second phenomenon which happens to be the phenomenon of

interest. Therefore, it is worthwhile examining the time related metrics of the roleplayers omitting the 14 players whose FAR is greater than 60%.

5.5.2 Time related metrics during lookup

Figure 5.13 depicts the frequency distribution of the FAWI of the set of 20 roleplayers whose FAR is 60% or less while Figure 5.14 depicts the frequency distribution of their T2FD.

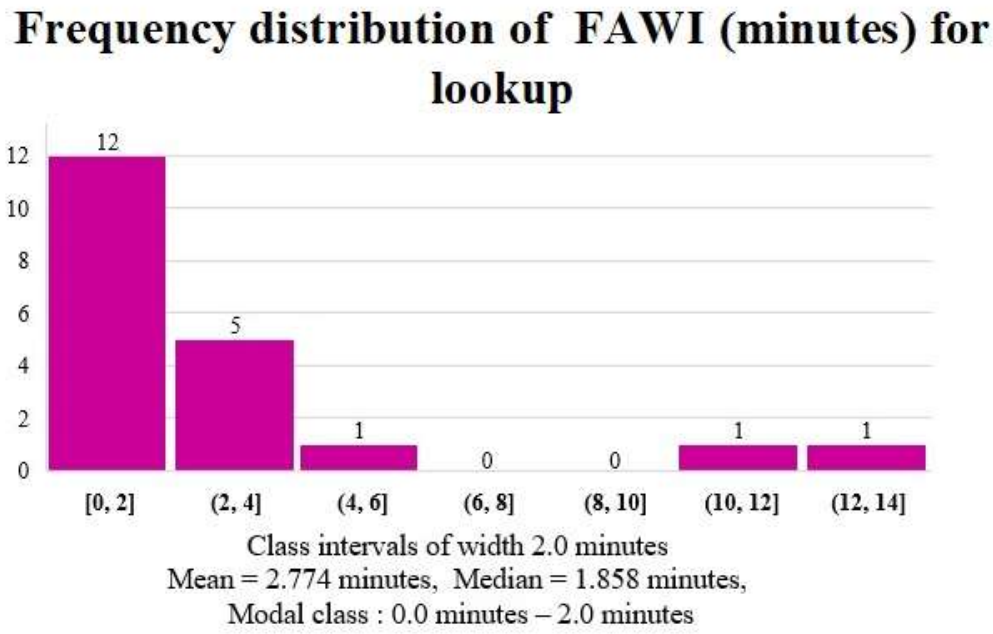


Fig. 5.13: Frequency Distribution of FAWI while looking up, across 20 roleplayers whose test taking exhibit an anomaly detection accuracy of more than 40%

It will be observed that in both distributions, the modal class is the leftmost, favoring the smallest possible values for the metric.

The roleplayer having an FAWI in the interval [10,12] and T2FD in the interval [2.558,3.058] is the same. However, the most right-flung roleplayer in the FAWI distribution in the interval [12.0,14.0] is in the modal class in the T2FD distribution. Also, the most right flung roleplayer in the interval [3.058,3.558] in the T2FD distribution happens to be one of the roleplayers with 100% FAR where the FAWI is in fact the session duration.

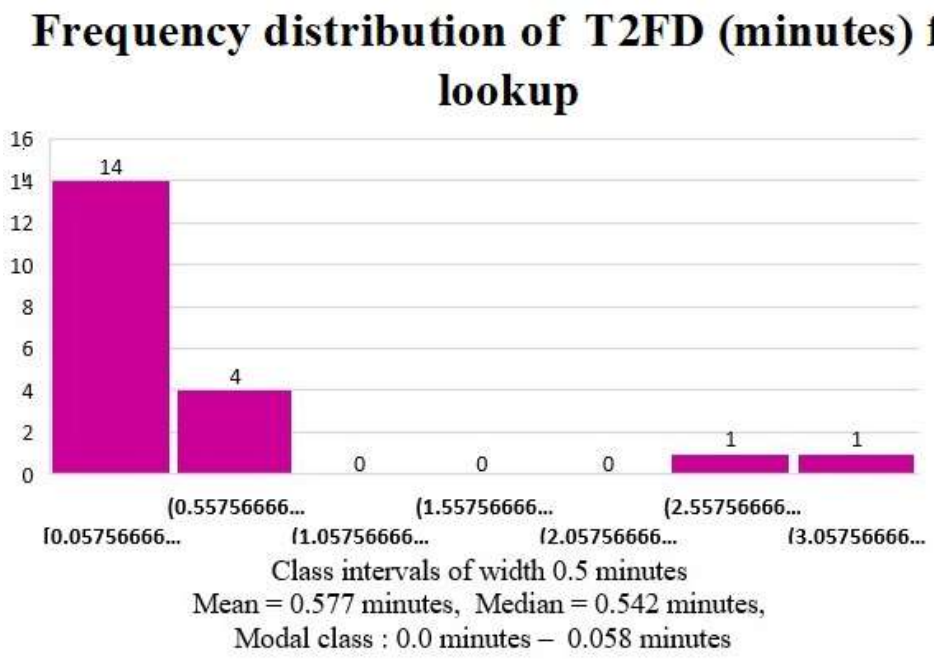


Fig. 5.14: Frequency Distribution of T2FD while looking up, across 20 roleplayers whose test taking exhibit an anomaly detection accuracy of more than 40%

The classifiers do not appear to have learnt the existence of the two phenomena or how to disregard the phenomenon that causes some of the roleplayers to be flung to the right.

5.5.3 Overall interpretation of results of simulation of test taking while looking up

The accuracy of anomaly detection being at least 50% in respect of only 47% of the population indicates that detection of the fraudulent act of lookup by the authentication system is less effective than the detection of impersonation.

5.6 Interpretation of results of the simulation of collaboration

There were 26 roleplayers with valid biometrics who had completed Test 04 which dealt with the particular fraudulent practice of 'collaboration'.

5.6.1 Accuracy of detecting collaboration

Table 5.9 shows the performance metrics of taking of Test 04 by the 26 roleplayers. In this case too, the number assigned to each roleplayer is for distinction only and has no relation whatsoever to the candidate's user ID in the experiments.

As in the case of Test 03, each test taking is a known case of fraudulent test taking where the roleplayer is answering while listening to voice clips. Therefore, every instance of the roleplayer being detected as 'genuine' would amount to a false acceptance. Accordingly, for the authentication system to be deemed as effective, the FAR should be low. In this case, there are only 2 roleplayers with FAR of 100% or the authentication system having failed. There are 3 roleplayers with 100% detection accuracy.

Table 5.9: Performance metrics of test taking by 26 roleplayers while collaborating

	FAR (%)	T2FD (minutes)	FAWI (minutes)		FAR (%)	T2FD (minutes)	FAWI (minutes)
1	48.00	8.99	14.69	**14**	71.00	3.51	4.98
2	27.00	1.43	1.98	**15**	62.00	0.13	3.81
3	69.00	2.72	2.82	**16**	0.00	0.15	0.00
4	20.00	0.66	1.03	**17**	37.00	0.11	2.02
5	0.00	0.28	0.00	**18**	24.00	0.16	0.71
6	78.00	2.37	2.31	**19**	4.00	0.39	0.22
7	53.00	0.43	0.95	**20**	7.00	0.13	1.24
8	54.00	3.89	6.73	**21**	34.00	0.46	3.23
9	100.00	-	18.33	**22**	77.00	13.39	11.15
10	70.00	0.15	4.62	**23**	12.00	0.10	2.83
11	77.00	0.19	15.36	**24**	0.00	0.39	0.00
12	2.00	0.21	0.12	**25**	16.00	3.36	0.32
13	100.00	-	26.16	**26**	0.00	0.57	0.00

Figure 5.15 shows how the False Acceptance Rate (FAR) is distributed among the population of 26 roleplayers.

As in the case of simulated lookup, in this case too, two phenomena at play are visible, one reinforcing a right skewed distribution with low FAR (or high accuracy) and the other reinforcing a high FAR (low accuracy) that is adversely affecting the authentication system. It is appropriate to examine the time related metrics with these two outliers omitted.

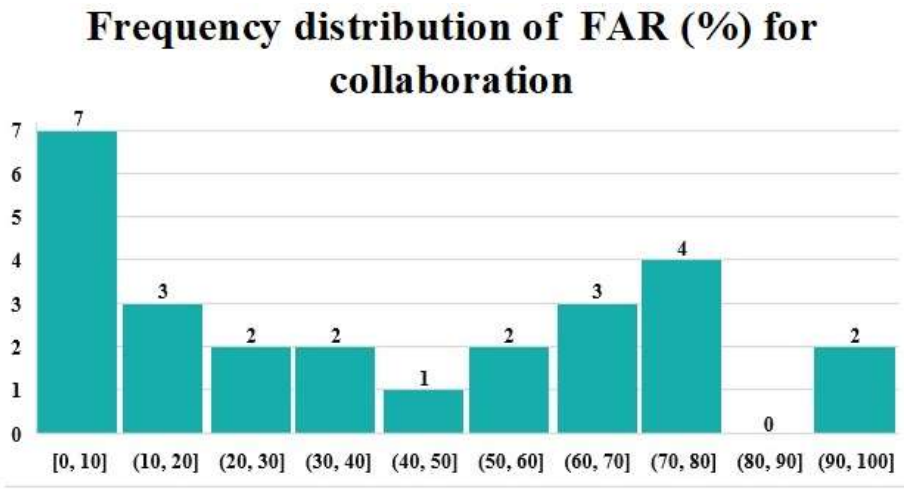


Fig. 5.15: Frequency distribution of FAR across 26 roleplayers taking test while collaborating

5.6.2 Time related metrics during collaboration

Figure 5.16 shows the frequency distribution of the T2FD across the 24 roleplayers remaining after omitting the 2 outliers while Figure 5.17 shows the frequency distribution of their FAWI.

.

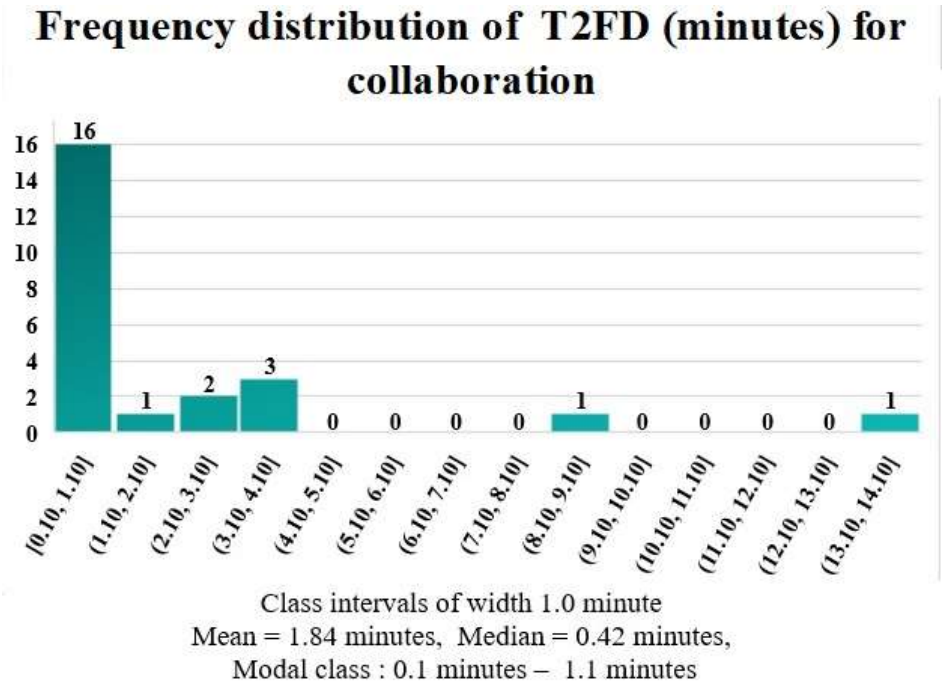


Fig. 5.16: Frequency Distribution of T2FD across 24 roleplayers whose test taking exhibits a non-zero anomaly detection accuracy

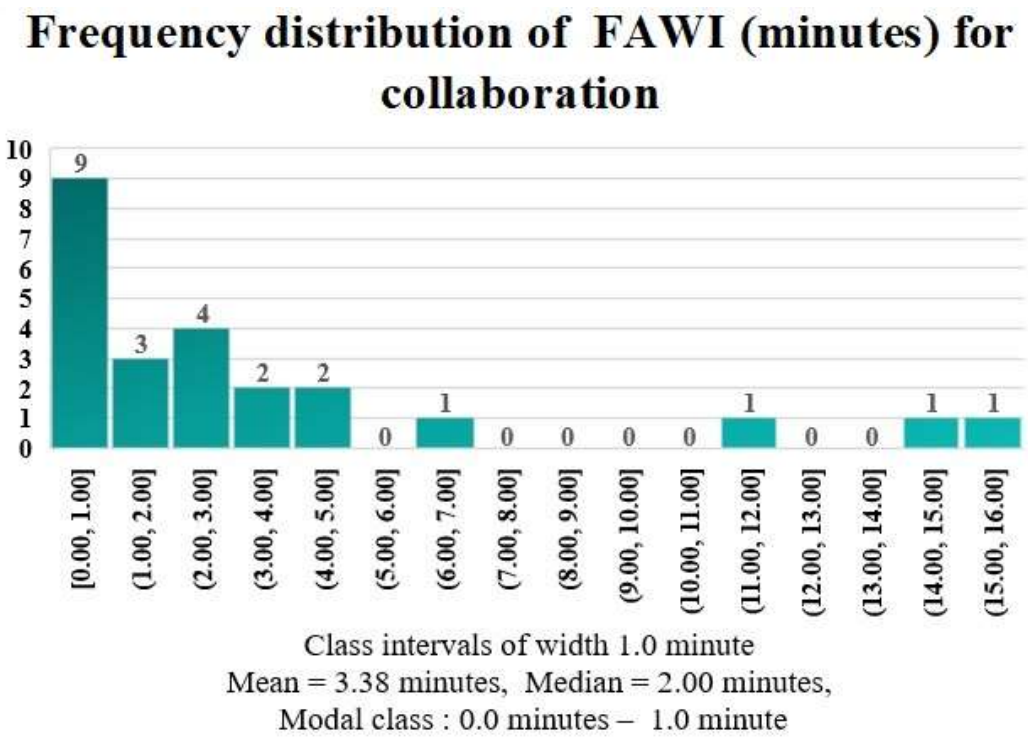


Fig. 5.17: Frequency Distribution of FAWI across 24 roleplayers whose test taking exhibits a non-zero anomaly detection accuracy

It will be observed that the two distributions are consistent, with 22 roleplayers exhibiting a distribution with an extremely high positive skew. In both distributions, the modal class is the leftmost, favoring the smallest possible values for the metric. The roleplayer having an FAWI in the interval [11.0,12.0] and T2FD in the interval [13.1,14.1] is the same while the roleplayer having an FAWI in the interval [14.0,15.0] and T2FD in the interval [8.1,9.1] is the same. However, the most right-flung roleplayer in the interval [15.0,16.0] in the FAWI distribution is in the modal class in the T2FD distribution. The two metrics need not necessarily correlate as they are independent. A similar distribution was observed in the case of time related metrics of Test 03 too.

In general, a distribution skewed more to the right for both T2FD and FAWI would indicate a more effective anomaly detection.

5.6.3 Overall interpretation of results of simulation of test taking while collaborating

The accuracy of anomaly detection being at least 50% in respect of approximately 58% of the population indicates that detection of the fraudulent act of collaboration by the authentication system is less effective than the detection of impersonation but more effective than the detection of lookup.

5.7 Comparison of performance when detecting different fraudulent acts

Table 5.10 gives a comparison of the average performance of the classifiers when detecting the acts of impersonation, collaboration, and lookup.

A marginally superior performance is exhibited only in respect of the FAWI amidst lookup. When compared to the performance of imposter detection, it is seen that all other metrics have degraded to different extents. This reinforces the observations made in 5.5.3 and 5.6.3 that the best performance is in respect of imposter detection.

Table 5.10: Comparison of detection performance for different fraudulent acts

	FAR (%)	T2FD (min)	FAWI (min)
Impersonation	33.0	0.52	3.00
Lookup	48.3	1.84	2.77
Collaboration	40.1	0.58	3.38

A marginally superior performance is exhibited only in respect of the FAWI amidst lookup. When compared to the performance of imposter detection, it is seen that all other metrics have degraded to different extents. This reinforces the observations made in 5.5.3 and 5.6.3 that the best performance is in respect of imposter detection.

The table also indicates that the accuracy of detection of collaboration is more than the accuracy of detection of lookup.

5.8 Interpretation of results of the Protocol Analysis experiments

The aim of these experiments was to examine the nature of the transfer of keystroke data from the candidate's device to the server. An alternative keystroke logger that used a significantly smaller keystroke record was also tested.

Figure 5.18 shows a screen capture of the JSON file that was created on the local host when the test sequence '1234ABCD' was used.

The contents of this JSON file is a splendid illustration of the mechanism used in the experiments to capture keystroke data of roleplayers when taking online tests. The key events generated when typing the test sequence '1234ABCD' are logged exactly in that sequence. Since the typing was done without haste with each key been released before the next key was pressed, alternating events had been logged one after the other with no overlap of key presses. The entry pertaining to the key down event of key

“\u0010” represents the pressing of the ‘SHIFT’ key to generate the uppercase letters that follow.

The contents of the JSON file also confirms the order in which the test sequence ‘1234ABCD’ had been transmitted to the server.

Figure 5.19 is a screen capture of the protocol analyzer window with the protocol filter set to HTTP, showing the traffic captured on the local loopback while the test sequence ‘1234ABCD’ was typed. Part of the capture is blown up for clarity.

HTTP data frame number 8 shows the browser executing the GET method to fetch the file ‘index.php’ which is the home page of Test 01. Frame 13 is the response of the server. Frame 23 shows the browser executing the GET method again to fetch the file ‘Q1.php’ which renders question 01. That is responded to by the server in frame 25.

The transmission of keystroke data captured when typing the test sequence in response to question 01 starts from frame 34 and continues through frame 49. In frame 34, the browser executes the POST method to upload the payload containing the first batch of keystroke data resulting from the typing of the test sequence. Once the frame is acknowledged by the server through frame 36, another POST method is executed through frame 49 to upload a payload containing the remaining keystroke data.

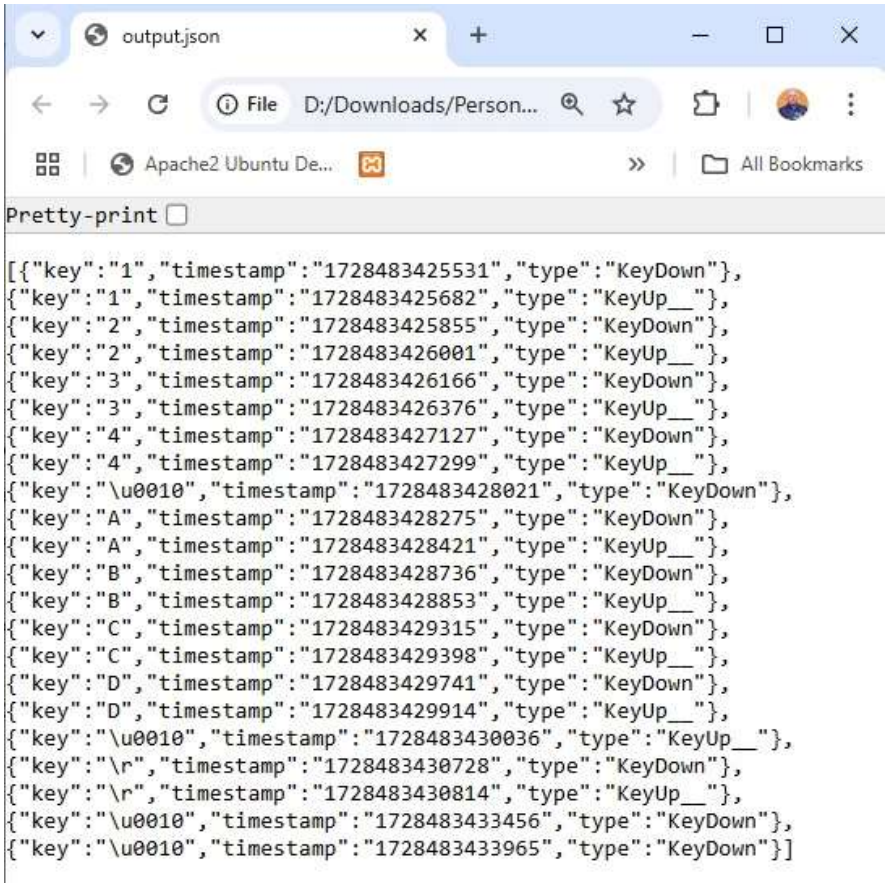


Fig. 5.18: Contents of the JSON file holding the keystroke events transmitted from the browser client.

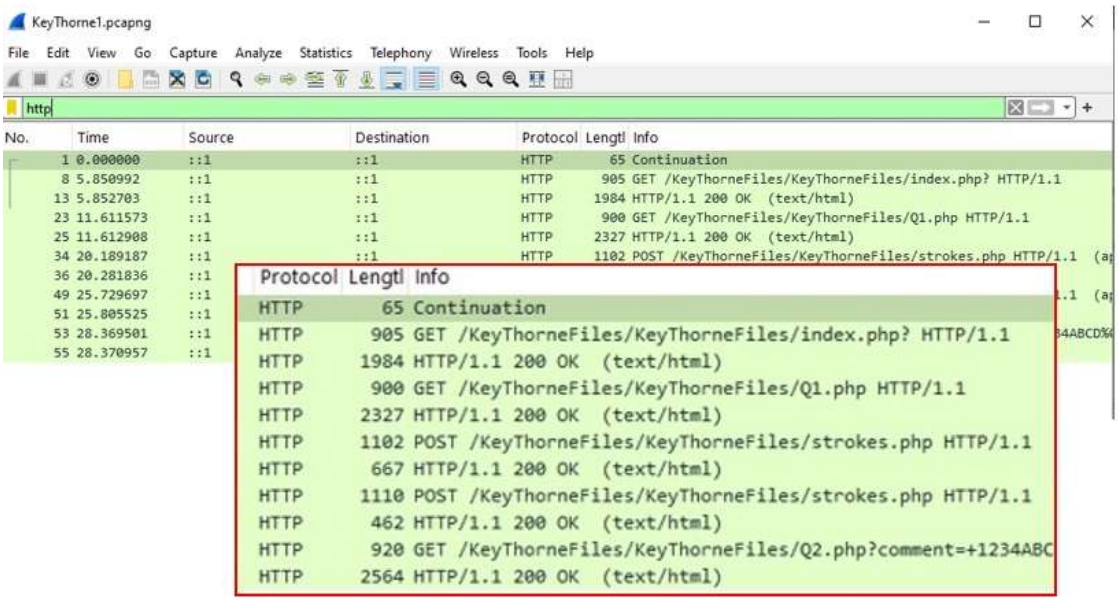


Fig. 5.19: Protocol analyzer screen showing the loop back traffic while the test sequence was typed.

Clicking the SUBMIT button triggers the execution of GET method to fetch the next question of Test 01.

The transmission of keystroke data captured when typing the test sequence in response to question 01 starts from frame 34 and continues through frame 49. In frame 34, the browser executes the POST method to upload the payload containing the first batch of keystroke data resulting from the typing of the test sequence. Once the frame is acknowledged by the server through frame 36, another POST method is executed through frame 49 to upload a payload containing the remaining keystroke data. Clicking the SUBMIT button triggers the execution of GET method to fetch the next question of Test 01.

As explained in Chapter 4, no form handler was associated with the SUBMIT button. As a result, there is no execution of a POST method following the explicit submission as typically found in form submission web applications. It is seen that the keystroke data were uploaded in two batches with the first batch being uploaded as soon as the HTTP payload size was exhausted, to be followed by the uploading of the second batch.

This confirms that the keystroke data are in fact being captured in real time and are being uploaded to the server at the earliest possible instance, requiring no explicit submission by the candidate. This is consistent with the concept which envisaged a real time logging of keystroke data.

Figure 5.20 is a screen capture of the protocol analyzer window showing the contents of HTTP frame 34, the frame that carried the first batch of keystroke data.

Comparison of the contents of the JSON file shown in Figure 5.18 and the ASCII dump on the right side of the analyzer screen shown in Figure 5.20 clearly indicates a one-to-one correlation between the two. The correlations may be observed in the lines of the hex dump starting at address 0050h and 0070h.

It will be noted that the string of bytes ’31 37 32 38 34 38 33 34 32 35 35 33 31’ are in fact the ASCII codes of each digit in the character string representing the timestamp of 1728483425531’ milliseconds of the key down event of key ‘1’.

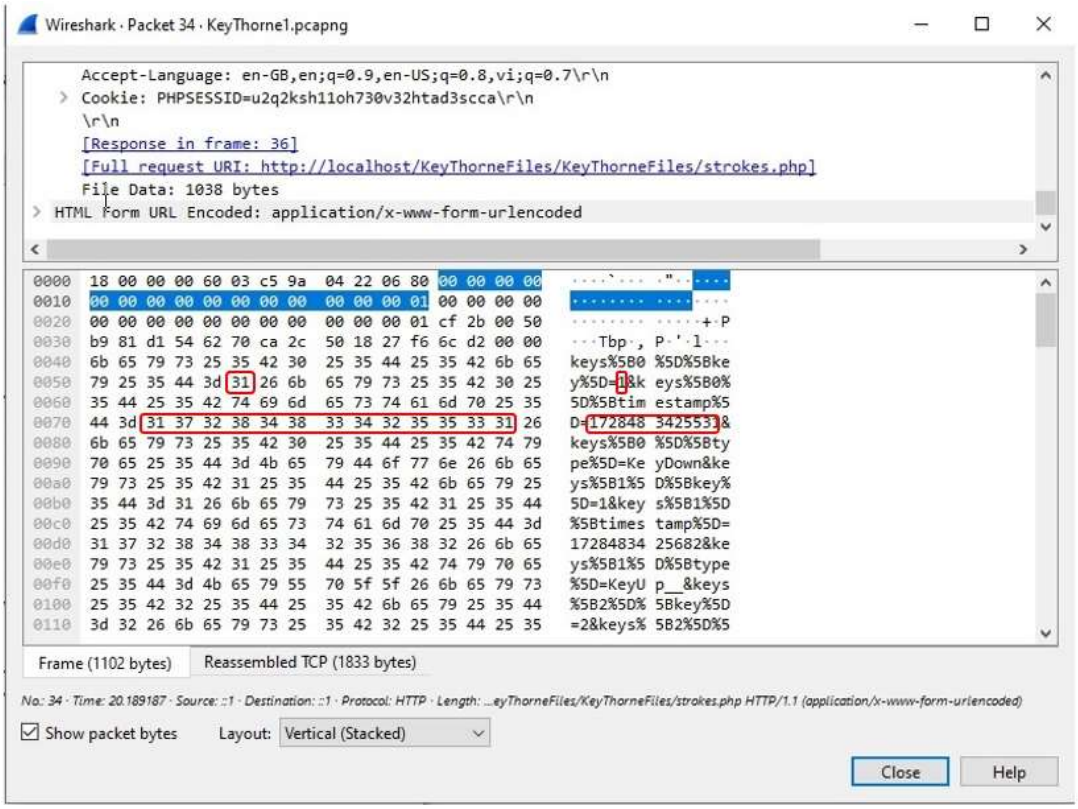


Fig. 5.20: Anatomy of the first HTTP datagram that carried the keystroke data of the test sequence to the server

This shows that the key value pairs containing the keystroke data of each key typed are encoded in ASCII as a string of characters before being uploaded to the server.

It could also be deduced from Figure 5.20 that the transmission of the keystroke data of a single key event will require the transmission of a total of 104 bytes of data in the HTTP payload. In an actual implementation of the concept, the arrival of this volume of keystroke data per each key event, which is in addition to the formal upload through the SUBMIT button, could be a concern.

Careful examination of any single record in the JSON log shown in Figure 5.18 shows that out of the 56 characters, only 21 carry biometric information while the rest are static data representing the format of the dictionary string. This format is what makes the log human readable, which was useful during experiments where manual

interpretation of the biometrics was required. However, in an actual implementation, these overheads will consume communication bandwidth unproductively.

The experiments described in 4.7.1.2 and 4.7.1.3 were carried out to examine the possibility of making this keystroke logging leaner.

Figure 5.21 is a screen capture of the byte file resulting from the typing of the same test sequence '1234ABCD' in the response panel when the modified keystroke logger was used. This file is equivalent to the JSON file resulting from the conventional keystroke logger. Therefore, the first 18 lines represent the 18 key events when the test sequence was typed. Each line holds 5 bytes, being the 5 bytes of the bit field encoding each key event as per the structure given in Figure 4.16. The first 4 bytes represent the truncated 9 digit timestamp in 'little-endian order'. The most significant 6 bits of the 5th byte represent the 6-bit key code. The LSB of this byte represents the type of the key event.



Fig. 5.21: The contents of the byte file resulting from the modified keystroke logging

This byte file is not human readable unlike the JSON log resulting from the conventional logging.

Figure 5.22 shows a screen capture of the protocol analyzer window for the above experiment. It will be noted that the lines in the hex dump starting at addresses 330h and 340h include the byte sequence 'd2 5b 23 29 05' which are identical to the 5 bytes in the first line in the byte file of Figure 5.21. This indicates that the 5-byte record representing each key event has in fact been transmitted to the server as a stream of 5 bytes. The 104 bytes of keystroke data of a single key event when the conventional keystroke logger was used, has been effectively reduced to 5 bytes, confirming that

the modified keystroke logger has successfully achieved significantly lean keystroke logging. This reduction would be beneficial in a practical implementation.

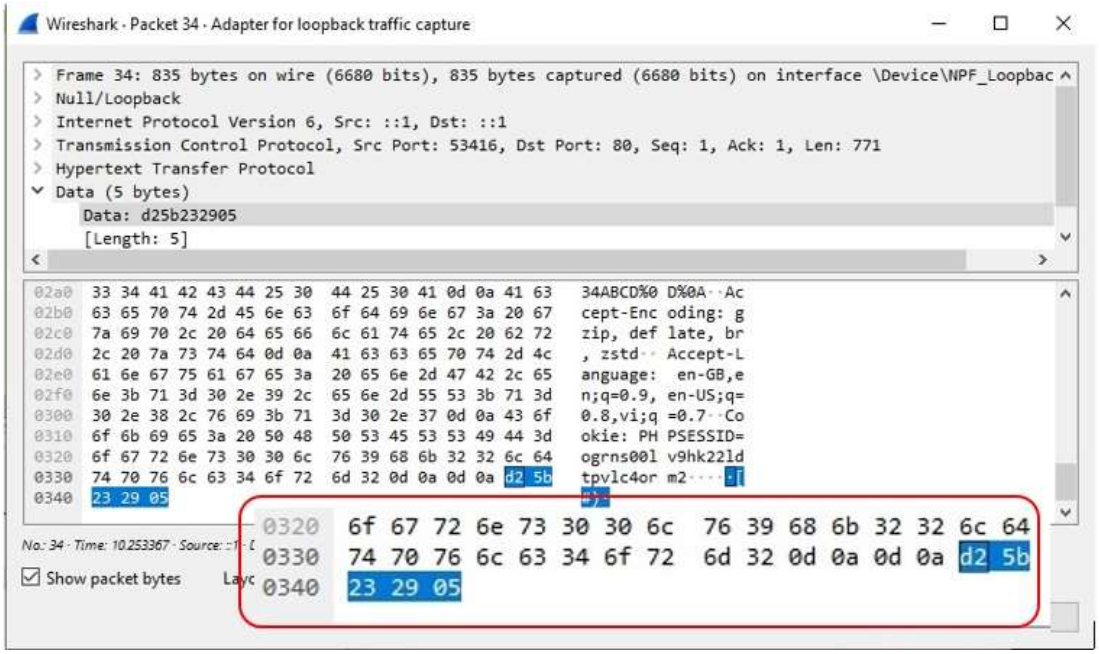


Fig. 5.22: Data on the wire when the test sequence '1234ABCD' was typed.

Figure 5.23 shows the variation of the HTTP payload size with the number of keys typed, for both types of loggers as derived from the separate protocol analyzer captures made while the two types of loggers were used.

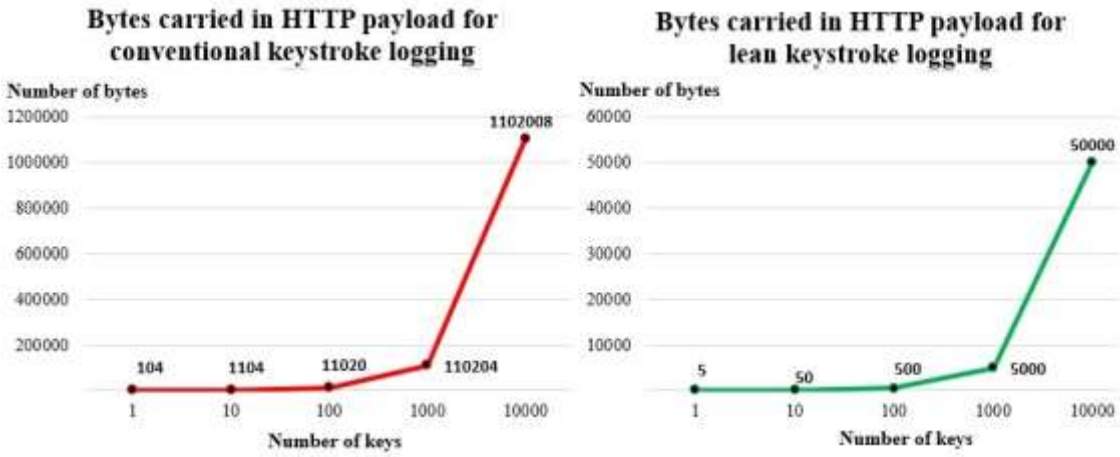


Fig. 5.23: Variation of HTTP payload size with the number of keys typed for both types of keystroke logging

The variation shown indicates that the lean keystroke logging works independent of the volume of input data. The variation of HTTP payload size resulting from lean keystroke logging is perfectly linear with each key event resulting in the transmission of exactly 5 bytes. The variation of payload size resulting from conventional keystroke logging is also linear to a great extent with each key event generating approximately

110 bytes on average. This is very close to the figure of 104 bytes that was observed for the isolated logging of a single key event. The small variations result when aligning the keystroke data within the payload by HTTP when the test session extends across several pages.

Transmitting data as raw bytes, as was done in the experiments that used a local host may not always be possible in practice. Raw bytes may imitate data items that have special meanings within the communication protocols used. Their interpretation as such may produce unpredictable results including the loss of the connection. Therefore it may be necessary to disguise the raw bytes with further encoding so as to escape the interpretation by the communication protocols. At the extreme, each byte may have to be coded using two bytes. When compared with an average of 110 bytes per key event, the doubling of 5 bytes to 10 bytes would still be lean.

In practice, online tests will be rendered using a secure protocol like HTTPS. This would have no bearing on the keystroke logging, whether conventional or lean, since the encoding of keystroke data is done before they are encrypted prior to being packed into the application layer payloads.

5.9 Interpretation of Results of the Clipboard Transfer Experiment

As mentioned in Chapter 4, the aim of this experiment was to identify the effect of clipboard transfers on the keystroke logging during an online test. A modified version of the test suit used for Test 01 was developed for this experiment.

Different input scenarios involving the typing of text from the key board and the pasting of text copied onto the clipboard were applied in response to the test.

Table 5.11 summarizes the observations made for each scenario of text input.

Table 5.11: Observations made upon different scenarios of text input

Input scenario	Observations made
The sequence '**Text via keyboard**' was typed into the answer textbox and submitted.	The sequence '**Text via keyboard**' was correctly logged in the answer file while the keystroke data of each character in that sequence were correctly logged in the JSON file.
The sequence '**Text via clipboard**' copied from another tab of same browser, was pasted onto the answer textbox using the 'paste' option in the browser's context menu, and submitted.	The sequence '**Text via clipboard**' was correctly logged in the answer file. However there were no keystroke data recorded on the JSON file.

Input scenario	Observations made
The sequence '**Text via clipboard**' copied from another tab, was pasted onto the answer textbox using the keyboard shortcut 'CTRL+V', and submitted.	The sequence '**Text via clipboard**' was correctly logged in the answer file. The keystroke data corresponding to the key combination 'CTRL+V' were recorded in the JSON file. But no data corresponding to the sequence itself were recorded.

Figure 5.24 is a screen capture of the response panel of the modified Test 01 showing a case where both the typed response and the responses pasted from the clipboard are in the same response panel, before the SUBMIT button was clicked.

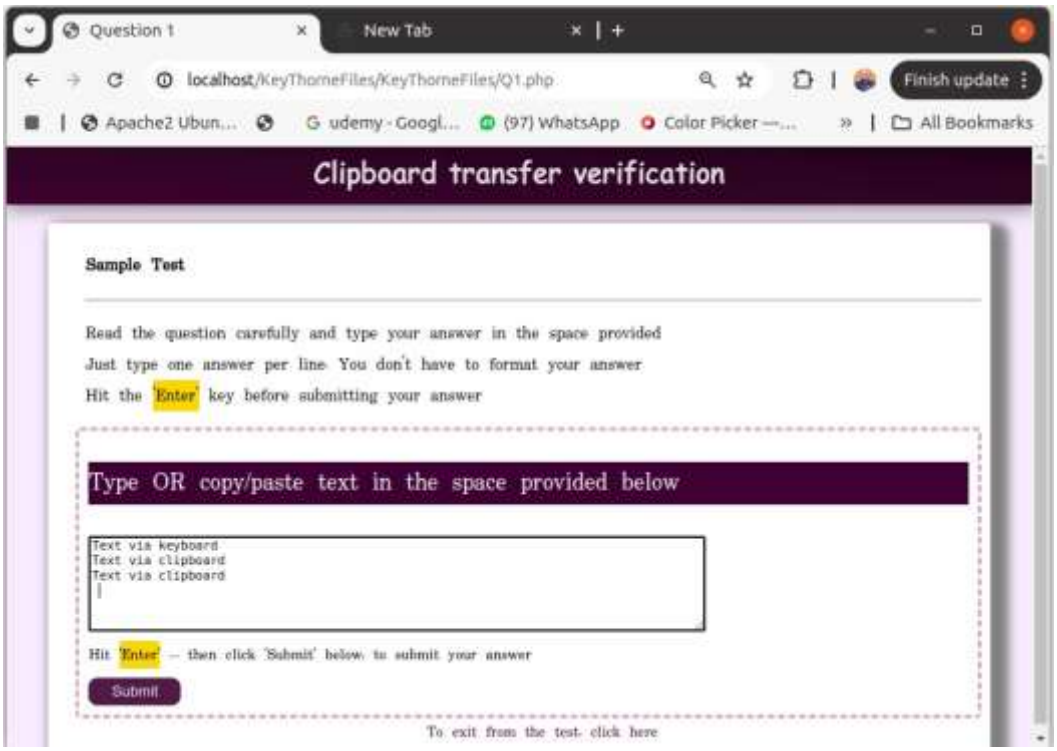


Fig. 5.24: Response Panel of the modified Test 01 to examine text transfer via the clipboard

Figure 5.25 indicates screen captures of the answer file and the JSON file resulting from the submission of the responses shown in Figure 5.24.

Careful examination of these two figures will indicate the following:

- The typed response has been correctly captured in the JSON file, character by character.
- However there is nothing in the JSON file to reflect the text pasted via the context menu.

- The keystrokes corresponding to the pressing of the CTRL key followed by the key ‘V’ when copied text were pasted using ‘CTRL+V’ have been logged in the JSON file, However, the sequence itself has not been logged.

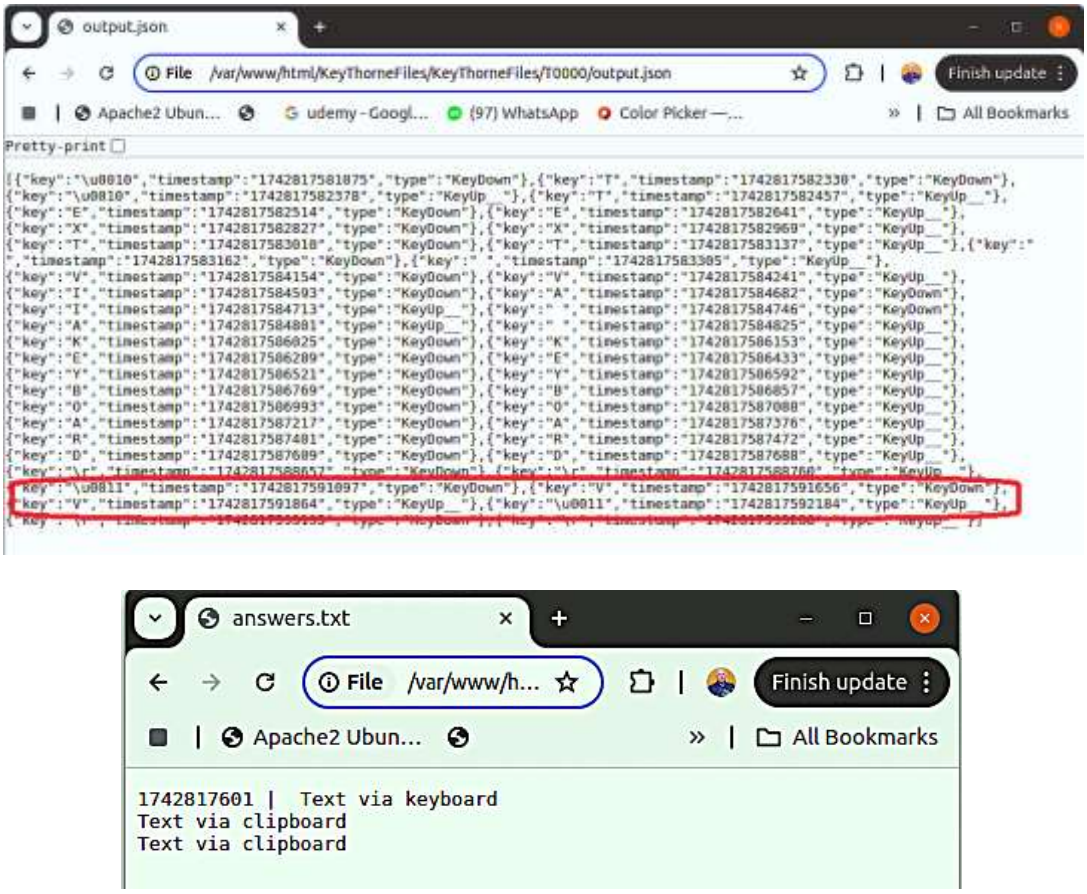


Fig. 5.25: Comparison of the responses captured separately on the JSON file and the answer file. (The keystroke data arising from the keyboard shortcut ‘CTRL+V’ are highlighted)

In the file ‘answers.txt’ storing the submitted text, all 3 sequences are present. Comparison of the timestamps confirms that the contents of both files relate to the same submission (note that the timestamp in the answer file is in ‘seconds’).

When the above tests were repeated with alternative sources for copying text as described in 4.7.2, the results were consistent with what is depicted in Table 5.11. As long as text is copied from a source outside the browser tab in which the keystroke logger is in execution, pasting them onto the response panel of the browser tab has no effect on the keystroke logging.

This indicates that the conventional keystroke logging function developed for the test suite in the experiments, disregards everything other than what is actually typed onto the text box. All that is required to detect clipboard transfers is to compare the contents of the answer submitted and the keystrokes logged. If the answer text file contains any

text for which corresponding entries are not found in the JSON file, those should essentially be text copied from another source on the same device. Hence the system used in the experiments is 'inherently capable' of detecting clipboard transfers. That includes text generated using LLM chatbots.

There are three important attributes associated with this anomaly detection.

i. Its accuracy is 100%. There is no uncertainty associated with it, statistical or otherwise.
ii. It is profile independent. It does not require the lookup of any data residing on a previously created profile of the candidate.
iii. Copying and pasting within the text box, which is not a fraudulent act, could be identified and exempted.

The last attribute listed above is a major advantage since even such copying and pasting is barred by most lockdown browsers.

The detection of clipboard transfer as verified above would partly deter the fraudulent act. While that would not prevent the lookup, making it impossible for the candidate to transfer the looked up text to the answer textbox without being detected would considerably diminish the undue advantage to the candidate. The candidate would then have to read and manually transcribe the answer rendered by the chatbot. As a result, the time advantage will be lost. In addition, the lookup is vulnerable to detection through the anomaly detection system.

5.10 Key findings

The key findings emanating from the interpretation of results of experiments are as follows:

- The system is capable of detecting the fraudulent act of impersonation with high accuracy comparable of that of biometrics based authentication systems.
- The system is capable of detecting the fraudulent acts of collaboration and lookup with dependable accuracy which is less than the accuracy of detecting impersonation.
- The system detects collaboration more accurately than lookup.
- Non-temporal features do not comprise a reliable source of anomaly detection.
- The volume of keystroke data that needs to be communicated to and logged at the back end is considerably large and contains a lot of redundancies. The lean keystroke logging system eliminates this irrationality.
- The authentication system is inherently capable of detecting clipboard transfers with absolute accuracy and independent of candidate profile.

5.11 Recapitulation

In this chapter, detailed interpretations on the results yielded by all the experiments that were conducted during this research were provided.

The most significant realization made through the interpretations was that the proposed anomaly detection system is capable of detecting impersonations with high accuracy. Other types of fraudulent acts are also detectable, but less accurately. It can be improved by using additional parameters.

Interpretation of results of the additional experiments showed that keystroke logging happens in real time, independent of explicit submissions by the candidate. The volume of data is irrationally high. The performance of an alternative keystroke logger developed to reduce the volume of data was also tested and confirmed.

It was also seen that the keystroke logging system used in the test suite does not log any text transferred via clipboard thus providing a vital clue for detecting clipboard transfers. This is particularly useful in detecting the transfer of text generated by LLM chatbots during online tests, which is a vital capability as far as learner integrity management is concerned.

The next chapter will deal with the establishment of proof of the concept, among other things.

CHAPTER 6

DISCUSSION

6.1 Chapter description

This chapter makes a benchmark comparison, establishes proof of the concept based on the results of the experiments interpreted in Chapter 5, provides answers to the research questions posed in Chapter 1, examines the drawbacks in the experiments, and discusses improvements that could be made to the authentication system.

6.2 Performance benchmark comparison

When selecting a benchmark system for performance comparison, it is necessary to ensure that it has attributes that are similar to those of the system under evaluation. However, due to the uniqueness associated with authentication during online tests which were highlighted in Chapters 1 and 2, it was difficult to locate such industry benchmark. In fact no systems used for the continuous authentication during online assessments in particular were locatable. Therefore, a system that provides a basis for reasonable comparison was chosen. The comparison is only in respect of detection of impersonation and not the other fraudulent acts.

The work of C. Murphy et al. titled “Shared Dataset on Natural Human-Computer Interaction to Support Continuous Authentication Research” [113] is a study carried out on continuous user authentication during general human-computer interaction sessions. Therefore, the scope is different from that of the current study, where the researcher particularly focusses on continuous authentication during online sessions, which is a unique case within the universal set of cases of human-computer interaction. Factors common to both studies are the focus on continuous authentication, use of ‘free text’, subject/machine binding, and the use of non-obtrusive keystroke dynamics. Hence, this work having reasonable commonality with the current study was chosen as the best available system for use as a benchmark.

The benchmark system had collected data from 103 users while they perform normal routine computer interaction which may or may not comprise online interaction. Since the system has not enforced typing constraints on the user and because the user has the choice of what should be typed during routine computer use, the system had used ‘free text’. It had enforced subject/machine binding where each user has been particularly instructed to use the same device. The system had captured mouse movements, mouse clicks, and information about the foreground and background processes in addition to keystroke data. Keystroke dynamics of key sequences involving more than two keys (n-graphs) had also been captured. Data had been collected over a long and

discontinuous period of two and a half years. As a result, the volume of biometric data collected from each user had been very large. Continuous authentication had been performed on sequences of 1000 keystrokes. The data collection had been performed in an 'uncontrolled environment' where users were typing as part of their routine functions and not playing any roles in the backdrop of a specific scenario. Although the system had been one of continuous authentication, only the traditional performance metrics like FAR and FRR which are insufficient for assessing continuous authentication systems had been used. Performance optimization had been done using the traditional approach used in access control where the system parameters are tuned to have an equal error rate (FAR = FRR).

The authentication system used in this study essentially used 'free text' and the authentication was continuous. Only 38 subjects were available for experimentation. The subjects played 'roles' under different scenarios such as genuine test taking, looking up, and collaboration. Therefore it could be said that data collection was performed under 'controlled circumstances' in the backdrop of specific scenarios. Subject/machine binding was enforced to the extent that each subject was required to use the same physical QWERTY keyboard for all tests. Only keystroke data were collected. Only the dwell time of individual keys and flight times of 2-key sequences were used for feature extraction. In addition, pause durations were also used. Therefore it could be said that the system used only monographs and di-graphs. Each roleplay involved a continuous online session that lasted around 30 to 40 minutes. As a result, the volume of biometric data collected from each roleplayer was comparatively small. Continuous authentication was performed on sequences of 40 'key events'. Since keystrokes may overlap and because key down events repeat when a key is held down continuously, 40 key events may not necessarily include 20 keystrokes. Since the system was one of continuous authentication, time related performance metrics like FAWI and T2FD were used in addition to the traditional metrics. Since non-rejection of genuine users was given prominence over the detection of imposters, no attempt was made to achieve equal error rate. Instead, the lowest possible FRR was achieved by selecting the classifiers that were least discriminating towards genuine users.

Table 6.1 gives a comparison between the chosen benchmark system and the imposter detection system used in the experiments of this research.

The benchmark system had recorded a FAR of 10.36% whereas the system used in the experiments of this research recorded a best performance of FAR = 6%. The average performance was much less, amounting to an FAR of 33%. It should be borne in mind that when the population size is small, average values are heavily influenced even by a few outliers. Also, the benchmark system used longer sequences of free text (1000) which would yield more representative biometrics which in turn would contribute towards a higher detection accuracy. However, this would be at the expense of authentication frequency.

Table 6.1: Performance comparison with a Benchmark System

	Benchmark system	System used in the current study
Number of subjects	103	38
Average size of biometric data per subject	12.9 M keystrokes	233 kBytes
Sequence length	1000 keystrokes	40 key events
Acquisition environment	Uncontrolled with subject/machine binding	Controlled, to simulate various scenarios during online tests, with subject/machine binding
Biometrics	Keystroke dynamics, mouse movements, mouse clicks	Keystroke dynamics
Other parameters acquired	Foreground and background process data	None
Keystroke entity	n-graphs	Monographs, di-graphs, and pauses
Anomaly detection method(s)	Gunetti & Picardi's algorithm	Binary classification using Naïve Bayes, Logistic Regression, Support Vector Machine, and Random Forest
Performance Metrics	Traditional	Both traditional and time dependent
Optimization	EER	Low FRR
Accuracy	FAR of 10.36%	Average FAR of 33% Minimum FAR of 6%

Considering the similarities and also paying due attention to the differences between the two systems as described above, it may be concluded that the continuous authentication system used in the experiment compares well with that of the chosen benchmark system.

6.3 Establishment of Proof of the Concept

This thesis proposed a concept for a continuous authentication system to be used for anomaly detection during online tests which would form the basis for test administering institutions to establish a system for managing learner integrity during online tests.

As indicated in the interpretation of the results of the experiments presented in Chapter 5, the anomaly detection system detects impersonation promptly, consistently, and with high accuracy. The fraudulent acts of looking up and collaboration are also detected promptly but with lesser accuracy. These limit the 'opportunity' for cheating candidates to engage in fraud during an online test.

The methods used for acquiring biometrics during the online tests used in the experiments were absolutely non-obtrusive and biometrics were acquired without the roleplayer being aware of the collection process. Therefore, the authentication process did not particularly require the cooperation of the roleplayer. No special devices were required to be used at the roleplayer's end for the acquisition of biometrics. Protocol analysis showed that the biometrics acquired are communicated to the server independent of any user initiated submissions.

Accordingly, the results of the experiments having indicated that a continuous authentication system built on the proposed concept is capable of performing anomaly detection during online tests with high accuracy, the proof of the proposed concept is thereby established.

A lean keystroke logging system and the capability to detect clipboard transfer of text from other sources were also realized during the experiments. These, not initially envisaged in the concept, provide strong reinforcement to a system based on the concept.

6.4 Revisiting the research questions

Having interpreted the results yielded by all the experiments conducted and proof of the concept having been established, it is now possible to answer the 'Research Questions' that were posed in Chapter 1, as follows.

RQ1 **What non-obtrusive, institution-ready methods capable of providing continuous authentication are available for managing learner integrity during online assessments?**

Being non-obtrusive in the context of this study refers to the authentication process being inconspicuous to the subject. A non-obtrusive process shall not provide any indication or clue whatsoever to the subject that the subject is under surveillance and that data pertaining to the subject are being acquired. The process should not cause any alteration to the natural behaviour and actions that would be exhibited by the subject in the absence of any surveillance.

Being institution-ready in the context of this study refers to the possibility for the authentication system to be implemented unilaterally by the test administering institution with no action or cooperation required from the subject. A system that

would require the subject to use a specific accessory or tool, perform a specific action, or adhere to a procedure that would not be required if the system was not used, will not be institution-ready.

Being capable of providing continuous authentication refers to the ability of an access restricted system to authenticate a subject continuously throughout the engagement of the subject with the system and not only at the instance of initial access. The objective is to detect subject switches throughout the session of engagement subsequent to authentication at initial access.

Table 6,2 shows the conformity matrix for authentication methods that are deployable during online assessments, with evaluations based on the learning acquired during the review of related work and from the results of experiments conducted in this study.

Table 6.2: Evaluation of Authentication Methods deployable during online tests

Method of authentication deployable during online tests	Non-obtrusiveness	Institution readiness	Ability to authenticate continuously
Password without 2FA	Does not conform	Does not conform	Does not conform
Password with 2FA	Does not conform	Does not conform	Does not conform
Fingerprint scan	Does not conform	Does not conform	Does not conform
Retina scan	Does not conform	Does not conform	Does not conform
Iris scan	Does not conform	Does not conform	Does not conform
360^0 scan of remote site	Conforms	Does not conform	Conforms
Webcam monitoring	Conforms	Does not conform	Conforms
Wear Cam monitoring	Does not conform	Does not conform	Partially conforms
Still picture monitoring	Conforms	Does not conform	Partially conforms
Acoustic monitoring	Conforms	Does not conform	Conforms
Facial recognition	Does not conform	Does not conform	Conforms
Gait analysis	Partially conforms	Does not conform	Partially conforms
Time related keystroke dynamics	Conforms	Conforms	Conforms
Non-temporal keystroke dynamics	Conforms	Conforms	Does not conform
Rhythm-click characteristics on pointing device	Conforms	Conforms	Conforms
Touch characteristics on touchscreen	Conforms	Does not conform	Conforms

As seen from the above conformity matrix, only two methods namely, time related keystroke dynamics and rhythm-click characteristics on pointing devices conform to all the three requirements. Therefore only these two methods that conform to all three requirements are presently available for the management of learner integrity during online tests.

RQ2 Does keystroke dynamics based continuous authentication when used in conjunction with lockdown browsers form the basis for the establishment of a system for the effective management of learner integrity during online assessments?

The concept presented in this research was for the establishment of such a system. It proposed the use of time related keystroke dynamics as the biometric to be used for authentication. Proof of the concept has been established which shows that keystroke dynamics based continuous authentication is effective during online tests. However, the concept did not specify that Lockdown Browsers should be used for taking of online tests.

As described in Chapter 1, a lockdown browser cannot detect impersonation, which is the most common fraudulent act during online tests. It has the ability to restrict look up and collaboration using the same device on which the test is being taken. It cannot prevent lookup from resources external to the device and also cannot prevent physical collaboration.

All experiments that were carried out to establish proof of the concept used industry standard browsers. The results of the experiments showed that the tested keystroke dynamics based authentication system is capable of detecting impersonation with high accuracy while lookup and collaboration too can be detected with relatively lower accuracy. In addition, the tested system can detect clipboard transfer of text with 100% accuracy without relying on user profiles which is particularly useful in curbing lookup on LLM. These indicate that the tested system that used industry standard browsers is capable of forming the basis for a system for managing learner integrity and does not require the use of a lockdown browser.

Several common functionalities of lockdown browsers contribute well towards maintaining learner integrity. These would deter looking up and collaboration using that device. Therefore, if the tested authentication system was used in conjunction with a lockdown browser, the effectiveness would be enhanced. The tested system would detect impersonation the lockdown browser cannot prevent while the lockdown browser would prevent lookup and collaboration using the device thus reducing the detection workload on the tested system since lookup and collaboration would be forthcoming only from sources outside the device.

It will be recalled that the tested system is capable of detecting clipboard transfers with 100% accuracy while being able to exempt clipboard transfers done within the response panel of the test browser tab. However, certain lockdown browsers completely suppress clipboard transfers altogether which would prevent the candidate from copying and pasting within the response panel. Since the possibility for the candidate to copy and paste the candidate's own typing within the response panel in the process of refining the response is useful to the candidate, a lockdown browser may eliminate room for this legitimate use whereas the tested system would permit it.

Accordingly, in response to the research question, it could be stated that keystroke Dynamics based continuous authentication forms the basis for a system for the effective management of Learner Integrity during online assessments even when used without Lockdown Browsers. When used in conjunction with Lockdown Browsers the effectiveness will be increased provided that the secure browser used does not inhibit clipboard transfers within the response panel of online tests.

RQ3 How should such a system be designed?

The design should build upon the three pillars, the non-obtrusiveness, institution readiness, and continuous authentication.

The design of the authentication system used in the experiment as described in Chapter 4 serves as an example for the design of a non-obtrusive system that provides continuous authentication for effective management of learner integrity during online tests. However, in a real life implementation, the mock test should be physical and conducted after verifying the identity of the candidates.

This experimental system may not be a good example for an 'institution-ready' system. Within the context of this study, institution readiness refers to the possibility for the authentication system to be implemented unilaterally by the test administering institution with no action or cooperation required from the subject. However a real life implementation would require the consideration of additional factors such as scalability, how it could be integrated with the institutional LMS, computational overhead within a multipurpose system, and deployment constraints both administrative and operational. These have not been tested in the experimental system, Therefore the 'institutional readiness' of the experimental system is yet to be determined.

Accordingly, the answer to the question as to 'How should an effective system for learner integrity management be designed?' is that continuous authentication capability and non-obtrusive operation could be borrowed from the design of this experimental system while, institution readiness should be verified through deployment.

RQ4 How should the effectiveness of the designed system be verified?

The designed system could be tested for its effectiveness through a trial implementation.

The experiments described in Chapter 4 serve as an example as to how a trial implementation should be carried out and how the non-obtrusiveness and continuous authentication capability could be verified.

Once these two aspects have been verified, institution readiness should be verified through pilot tests in an actual institutional system.

6.5 The drawbacks in the experiments

Although the experiments yielded results that contributed towards the successful establishment of proof of the concept, examining the few instances where the results were not consistent with what was envisaged in the concept will be useful for the practical implementation of authentication systems based on the concept.

6.5.1 Low accuracy in detecting fraudulent acts other than impersonation

It was observed that the continuous authentication system while being dependably accurate in the detection of impersonation, was comparatively less accurate in detecting the fraudulent acts of lookup and collaboration.

During the experiments, there was prior knowledge about the fraudulent practice that will be simulated for the purpose of testing. Therefore, it was possible to separately identify the performance of the authentication system when each fraudulent act was in progress. In a practical implementation of the concept, the authentication system will have no prior knowledge about what is in progress at the candidate's end. It will be able to detect anomalies but will not be able to identify as to what fraudulent act actually caused the anomalies. If impersonation was in progress, it will be detected with high accuracy. If some other fraudulent act was in progress, it would still be detected but with a lower accuracy. The authentication system should ideally detect anomalies due to any fraudulent act with approximately equal high accuracy.

When the fraudulent acts of impersonation, lookup, and collaboration are compared, it is seen that a person other than the candidate will be typing only in the case of impersonation. In lookup and collaboration, any anomaly in the biometrics will be solely due to the influence of the fraudulent act on the typing. As an example, the keystroke behavior of a candidate when typing out of his/her own thinking matches his thinking pattern and speed. When typing while reading from another source, the speed is influenced more by the speed of reading and comprehension rather than the speed of thinking. Pausing and hesitation while reading and typing would also be different from pauses made while thinking. The same is applicable during collaboration where the candidate types after listening to a third party. The influence of the fraudulent act on the keystroke dynamics due to 'the presence of an external source in the loop' is intuitively understood. During the experiments investigating collaboration and lookup, this influence was present but 'not insignificant'. This caused the accuracy of detecting the resulting anomalies to be less.

The reason for the low accuracy could be traced to the training of the classifiers used in the experiments. It will be recalled from Chapter 4 that for each feature, training datasets were created using the biometrics of Test 01. The classifiers have correctly learnt to distinguish between differences arising from the mismatch between biometrics of two persons. This is why impersonation was detected with high accuracy. But the classifiers have not learnt to discriminate anomalies in the biometrics of the same person caused through other factors during test taking. The classifiers have not had an opportunity to learn these because they had been trained using datasets in which impersonation was the sole cause of anomaly. As a result, an anomaly due to a fraudulent act other than impersonation would be detected only if the anomaly is sufficiently high to be comparable with an anomaly caused due to the biometrics of another person. This accounts for the low accuracy in detecting those acts.

To correct this, it is necessary to improve the training datasets so that they include anomalous data resulting from fraudulent acts other than impersonation in addition to anomalous data resulting only from impersonation. To achieve this, test items of the types used in Tests 03 and 04 which simulated lookup and collaboration, should be included in the mock test.

6.5.2 Non-usability of the biometrics of certain subjects

Although 59 roleplayers had completed Test 01, effective profile creation was not possible for all of them. The JSON files of 6 roleplayers were found to be corrupted at the beginning of the profile creation phase while the biometrics of 15 more roleplayers were found unsuitable during profile creation. Out of 37 roleplayers who had completed Test 03, only 34 had usable biometrics and out of the 29 roleplayers who had completed Test 04, only 26 had usable biometrics.

Examination of the unusable JSON files revealed the following:

i. Some files were empty.
ii. Contents of some files were corrupted.
iii. Some files had recorded very few keystroke events.

In the experiments, the acquisition of biometrics was performed by an ECMAScript running on the test taking device and a PHP script running in the backend. The algorithm used by this PHP script is shown in Figure 6.1.

This PHP script is executed each time the client code in the ECMAScript submits a string of characters.

Careful analysis of the algorithm reveals that the script reads the contents already available in the file ‘output.json’ into an array, appends the character string received from the ECMAScript to the array, and overwrites the contents of ‘output.json’. The file is overwritten for every new character typed on the client device and the file grows

dynamically. The file 'output.json.' is first created when the very first character string received from the client side is written to it.

```
// This is executed each time the front end process uploads a batch of
//  keystroke data.

Inherit session parameters on server side

buffer ← {contents of candidate's JSON file as a single string}
if buffer is NULL          // No keystroke data received yet
    initialize buffer

initialize array as a NULL string
array ← JSON decoded contents of buffer
append the keystroke data received from front end to array
overwrite candidate's JSON file with JSON encoded contents of array

return
```

Fig. 6.1: Algorithms used by the back end script for logging keystroke data

There was not a single instance where at least an empty JSON file had not been created. This explains that both the PHP script and the ECMAScript have executed correctly during each test session. It also suggests that the only eventuality that could cause a JSON file to be empty is for null strings to be continuously received from the client end throughout the session, irrespective of what is being typed.

A JSON file could have become corrupted while in storage on the server, while being downloaded from the server, or while being held in local storage for experimentation. It may also be corrupted if the keystroke data received from the client side themselves were corrupt.

The first possibility could be ruled out because the tests were hosted in the cloud of a reputed service provider in which data integrity is assured as part of the paid service. The second possibility too could be ruled out because the downloading was performed using industry standard tools using Secure File Transfer Protocol (SFTP) where the download happens over an encrypted channel and many integrity checks are used to ensure the error-free end to end transfer of the data. The backed up versions of corrupted files that were made immediately upon download were also found to be identical to the original files. This suggests that the files have been downloaded in corrupted form. Accordingly, the third possibility too can be ruled out. Only the last possibility remains. Therefore, it may be reasonably concluded that corrupted data had been uploaded from the client device during the test and had been logged by the PHP script, resulting in a corrupted JSON file being created in the first instance.

Certain settings in the browser could influence the encoding of keystroke data by the ECMAScript. The encoding may have been performed incorrectly due to the use of incorrect codecs. Another reason could be the roleplayer using touchscreen keypads on a smart device contrary to the instructions. The usability of touchscreen keyboards have not been studied in the experiments. The versions of browsers supported on smart devices using different operating systems may not function identical to those on computers in all respects and may use codecs that are not supported by the encoding function in the ECMAScript.

The JSON file size being small would largely be due to the roleplayer ending the mock test prematurely and due to hap hazard test taking.

It is essential that all the above eventualities are prevented in an actual implementation of the concept.

6.5.3 Inadequate replication of human factors during tests in the experiments

In the experiments, online tests were conducted for taking by volunteers who roleplayed as candidates. There was actual online deployment of tests closely resembling those used by test administering institutions. Roleplayers responded to actual online tests which captured their submissions. Briefing for the roleplayers made through the intermediaries as well as through briefing pages included in the test suite itself prepared them adequately for roleplay as genuine online test candidates. The results indicated that the taking of tests were ‘functionally equivalent’ to real online tests.

However, whether the roleplayers were subjected to the human factors actual online test candidates are subjected is in question.

A candidate takes an online test in pursuit of some stake and the evaluation of his/her performance will lead to the placement of the candidate within a performance spectrum which would have a bearing on his academic career. Often, the rewards may be competitive where not only the absolute performance but the performance compared to other candidates also matters.

In academic programmes, it is common for students to bear a heavy academic workload which makes the management of time for adequate preparation a challenge. As a result, the candidate is often under considerable stress while preparing as well as during the test. The performance of the candidate is inevitably colored by these influences which have more to do with human factors rather than technical functionalities. The possibility for these factors to influence the candidate’s behavior and in turn, any biometrics derived from them cannot be ruled out.

The tests in the experiments did not carry any stakes for the candidates and they were aware that they are not under evaluation. The tests required zero preparation. The tests also did not enforce a time limit.

As a result, the roleplayers were not subjected to the stresses endured by a typical online test candidate. Although the experimental results have established proof of the concept, it is worthwhile considering whether the behavioral biometrics of a candidate captured while he/she was enduring time pressure and other stresses resulting from human factors would be representative to the same extent as those in the candidate's profile.

Therefore the inadequate replication of human factors need to be recognized as a drawback in the experiments.

6.6 Suggestions for improving the authentication system

6.6.1 Alternative classifiers for detection of fraudulent acts other than impersonation

The relatively low accuracy in detecting fraudulent acts other than impersonation was attributed to the lack of training of the classifiers on such anomalies due to lack of anomalous data. This could be overcome by using classifiers that learn 'unsupervised' where labeled data are not necessary. One-class binary classifiers are the most suitable.

Both the work of Y. Sang et al. [114] and the work of E. Yu and S. Cho [115] are examples where such classifiers had been used. One-class-SVM classifiers do not require a dataset labeled as anomalous.

6.6.2 Use of machine learning and Artificial Intelligence

As stated in 4.8, statistical feature extraction was preferred in the development of this system. All primary features were based on statistical parameters. Machine learning was used for anomaly detection using the derived features and the reasons were explained therein.

However, one important improvement that could be included in the system is streamlining the response to detection of fraudulent acts using 'Explainable Artificial Intelligence (XAI)'. This study was confined only to the detection of fraudulent acts and did not deal with the aftermath or how the test administration should react when an anomaly is detected. It has administrative implications. Any punitive action on a candidate who has indulged in fraud during the online test needs to be backed by conclusive evidence. As opposed to rigid classification, XAI would accompany each fraud detection with plausible insights as to why the detection was made. These are important for administrative and audit purposes.

6.6.3 Improvements to the profile creation process

The features that were well extracted during the experiments from the biometrics captured in Test 01 are the dwell time vector and the pause duration ensemble. The keystrokes captured during Test 01 contained sufficient key events to extract these two features that were adequately representative of the subject. The other vectors were either not extractable or poorly extracted during the experiments. The reason is that the test items of Test 01 do not provide adequate opportunities for typing the key sequences required for the extraction of these vectors.

Since the institution would invariably collect personal data of the candidates such as gender, dextrality, and age, soft biometric classification as described in the work of Jain et al. [61] could also be used.

To make the mock test similar to the subsequent online tests where the candidate is subject to human factors as described in 6.5.3, the mock test should be designed to replicate those human factors.

Therefore, the following improvements could be made to the process of profile creation:

i. Inclusion of test items in the mock test to cause more frequent typing of the required sequences.
ii. Inclusion of test items in the mock test that simulate fraudulent acts (like in Tests 03 and 04) so that they become sources of anomalous data representing fraudulent acts other than impersonation.
iii. Inclusion of classifiers that use keystroke dynamics as a soft biometric.
iv. Replication of human factor related conditions during the mock test by introducing stakes through reward or penalty and enforcing deterrents such as time pressure.

6.6.4 Improvements to the biometrics acquisition process

The process improvements to the acquisition of biometrics are largely non-technical. They include the prevention of the following during the mock test:

i. Candidate exiting the mock test prematurely.
ii. Candidate responding in a haphazard manner without providing plausible responses.
iii. Candidate skipping questions, providing a standard response to all questions, or providing very brief answers to questions requiring free text responses.

6.6.5 Improvements to the client – server communication during online tests and the processing of keystroke logs

The lean keystroke logging system that was developed and tested in the experiment has two limitations.

i. The truncation of the 13-digit timestamp of each key event to 9 digits cause the values of the year, month, and the day of the month in the timestamp (which were assumed to be invariant during a test of 3 hours) to be omitted. If the online test was conducted over a duration that included an instance when any of these three values underwent an increment, the 9-digit time will rollover at that instance. If two 9-digit timestamps on either side of such rollover instance were subtracted, it would not yield the correct time difference between the two timestamps.
ii. The byte file is incompatible with the scripts used at the back end that parse the JSON logs for analysis.

To overcome the first, any instances of rollover of year/month/date should be taken into account prior to making any time difference calculations.

```
open byte file in read mode
create JSON file in append mode
append '[' to JSON file

b[] ← Initialized byte array of 5 bytes
key = timestamp = type = NUL
dictionary_entry = NULL

while end of byte file is not reached {
        b[] ← 5 consecutive bytes from pointed location in byte file
        key = 6-bit_code_table_reverse_lookup(b[4]/2^2)
        if (b[4] is odd)
                type = 'KeyDown'
        else
                type = 'KeyUp__'
        timestamp = b[0] + b[1]*256 + b[2]*256^2 + b[3]*256^3
        dictionary_entry = '{"key":"'+key+'","timestamp":"'+str(timestamp)+'","type":"'+type+'"},'
        append dictionary_entry to JSON file
        increment pointer of byte file by 5 bytes
        }
decrement JSON file pointer by 1 byte
overwrite current byte with ']'

close all files
```

Fig. 6.2: Algorithm for converting 'byte file' to a JSON log

To overcome the second, the scripts could be modified to directly process the byte file. However, a better option would be to construct the required JSON log from the byte

file. This is always possible theoretically and could be done using a single pass file processing of the byte file. Figure 6.2 shows an algorithm that could be used to make this conversion.

It should be noted that the JSON file regenerated in the manner will not have 13-digit timestamps. This could have administrative implications for the institution because the timestamps must be consistent with the times of the actual test taking. The 13 digit timestamps could be recomputed by taking the actual time of the test into account.

If the system uses UTC synchronized timestamps, the leap second correction built into them as mentioned in Chapter 4 should be eliminated after considering the actual time of the test, before any time difference calculations are performed.

6.7 Identification of points of failure

It is pertinent to identify the possible points of failure of the tested system based on the experiences during the experiments.

i. The data acquisition layer[9]

   The keystroke data being corrupted during acquisition and logging was experienced in the experiments, during the mock test as well as the subsequent tests. Improper capture could result due to functionalities in the browsers. Corruption of data may also occur while in transit from the front end to the back end.

   Possible mitigation strategies are the integration with an LMS, including checksums in the keystroke logs, and relying on the forward error correction capabilities of the network protocols

ii. The classification layer[10]

   Failures may be caused within the classification layer due to all known limitations of machine learning, given the limited dataset available for training and testing. These include biased learning, overfitting, underfitting, and outlier distortion.

   The mitigation strategies are applicable to the feature extraction layer where the quality of the training data should be ensured at the instance of creating them.

[9] Refer the architecture presented in section 3.6.

[10] ibid.

6.8 Key takeaways from the discussion

- The benchmark comparison positions the developed system as a dependable system
- The experimental results establish proof of the concept proposed in Chapter 3.
- The research questions posed in Chapter 1 have been answered conclusively.
- The identified drawbacks in the experiment are the low accuracy of detecting fraudulent acts other than impersonation, some biometric sets collected from subjects been unusable for the experiments, and the inadequate replication of human factors applicable to online test taking during the experiments.
- Improvements suggested for the system are the use of alternative classifiers, improved profile creation, use of machine learning for primary feature extraction, use of XAI for anomaly detection, improved acquisition of biometrics, and further improvements to the lean keystroke logging system.

6.9 Recapitulation

In this chapter, a benchmark comparison was made. Proof of the concept proposed in Chapter 3 for a system for continuous authentication of candidates during online tests was established.. The research questions were answered conclusively, based on the results of the experiments. The drawbacks in the experiments conducted were identified and improvements that could be made in an actual implementation of an authentication system based on the concept were discussed.

CHAPTER 7

CONCLUSION

7.1 Chapter Description

This chapter presents the overall conclusion of this study, identifies the original contributions made, and provides directions for future research.

7.2 Overall conclusion made through the research

As stated in Chapter 1, this research was conducted in pursuit of an answer to the following research problem:

> ***"There is no existing, institution-ready, system for the effective management of Learner Integrity during online assessments through the non-obtrusive and continuous authentication of candidates"***

During the review of related work, 3 gaps were identified in the knowledge domain of technology assisted fraud prevention for the management of learner integrity.

The first gap was that the current application of technology is limited to the 'offline detection' of plagiarism. The authentication system designed and tested in the current study was shown to be capable of 'online detection' of several fraudulent acts including plagiarism.

The second gap was that the effectiveness of keystroke dynamics based authentication in the detection of fraudulent practices during 'online tests' is not verified. The current study verified through simulations that keystroke dynamics is effective for the continuous authentication during online tests.

The third gap was that Curbing academic dishonesty during online tests is currently limited to restricting the candidates' online activities to the test by enforcing the use of 'secure browsers' and to exercising 'remote proctoring' with heavy utilization of resources. The authentication system designed for the current study was found to be effective even without the use of secure browsers. The system also provided continuous authentication through the sparse usage of resources as opposed to heavy usage of resources required for remote proctoring.

Therefore, the findings of the current research has successfully bridged all 3 gaps that were identified.

The 4 research questions stemming from the research problem were answered conclusively in section 6.4.

Accordingly, the overall conclusions made through the research could be stated as follows:

- An institution-ready system for the effective management of Learner Integrity during online assessments could be established through the non-obtrusive and continuous authentication of candidates using time related keystroke dynamics.
- The system would also be able to deter undue lookup of LLM during online assessments through the accurate detection of clipboard transfer of text.
- Use of lockdown browsers although not an essential component of the system, would enhance its effectiveness.

The system being applicable only to online tests requiring text based responses and its effectiveness not been verified for the use of touch screen keyboards are its limitations. Elimination of these limitations could inspire future research.

7.3 Contributions of the research

The following original contributions resulted from this research to the knowledge domains of learner integrity management in online testing and biometric authentication:

i. Use of the 'Angle of departure' as a parameter for biometrics based authentication.
ii. Formal definition of 'pause durations' using 3 keys and its suitability for authentication during online tests.
iii. Unsuitability of non-temporal keystroke dynamics features for use as authentication parameters during online tests.
iv. Detection of clipboard transfer of text during online tests which effectively deters the lookup of LLM while taking online tests.

7.4 Directions for future work

The following are some directions for future research and development particularly in the domain of authentication during online assessments as well as in keystroke dynamics based continuous authentication in general.

7.4.1 Additional features of non-obtrusive nature

Use of additional features for classification would naturally boost the effectiveness of any authentication system. However, in the case of continuous authentication during

online assessments, the requirement for non-obtrusive acquisition of biometrics does not permit the use of biometrics other than the time related keystroke dynamics.

This research used monographs, i.e. dwell time of single keys and di-graphs, i.e. the flight time between two consecutive keys as biometrics. The pause durations were a special case of tri-graphs because the manner in which a pause was defined in this research involved 3 consecutive keys.

The work of K. Balagani et. al [116], establish that flight time related features lead to higher detection accuracy compared to dwell time related features. Since flight times are totally person dependent, this conclusion is intuitively acceptable. Using features based on tri-graphs and higher order n-graphs would provide for the extraction of more features that would lead to more accurate anomaly detection.

As an example, the following is the set of 20 tri-graphs with the highest occurrence in English text (web article of B. Keating [108]):

{ the, and, ing, her, hat, his, tha, ere, for, ent,
ion, ter, was, you, ith, ver, all, wit, thi, tio}

Once extracted, a multi-dimensional vector as done for the hot sequence vector could be constructed and anomaly detection could be performed using the angle of departure or some other suitable feature.

The extraction of di-graphs required the consideration of 4 distinct cases depending on the occurrence of each key event on the time axis. Similarly, the extraction of tri-graphs would require the consideration of 8 distinct cases. The complexity introduced by the cases of the first two keys being held down separately also needs to be considered. Hence the extraction of tri-graphs would be computationally more complex. More generally, the complexity of extracting *n*-graphs will increase exponentially with *n*.

One problem that could be anticipated is the scarcity of the tri-graph sequences in the typed text. In the experiments, the scarcity of even two-character sequences (di-graphs) within free text was experienced. The tri-graphs are bound to be even more scarce. Therefore, classifiers that could work efficiently on small datasets will be necessary.

The use of rhythm click characteristics to supplement keystroke dynamics is another area that could be exploited for continuous authentication during online tests. Since candidates taking online tests would invariably use a pointing device or a pointing mechanism to navigate through the test suite, the rhythm-clicks will also be available for capture. In addition, in objective tests like MCQ where typing is not required and responses are provided only through the pointing device, keystroke dynamics will be irrelevant and rhythm-click characteristics will be the only biometric available. Rhythm-clicks are also a non-obtrusively acquirable biometric. Therefore, combining

it with keystroke dynamics would not affect the non-obtrusive nature of the authentication system.

Rhythm-click characteristics are relatively unexplored compared to keystroke dynamics. Therefore, there could be a lot of room for innovative fusion of the two in continuous authentication during online tests.

The exploration of the use of other non-obtrusive features for continuous authentication during online tests would be an important area of future research.

7.4.2 Improved classifiers

The research used 4 types of classifiers. That was sufficient for the proof of the concept. Trying out other types of classifiers for the same datasets used would be another useful avenue of exploration. That would broaden the selection of the best classifier.

Since the datasets are bound to be small and often one-dimensional, learning machines that are optimized for such datasets should be focused on.

Incorporation of ensemble detection for authentication during online tests is another option that could be explored. Ensemble learning refers to the training of several models of classifiers on the same datasets and combining their predictions to obtain accuracies greater than what could have been achieved with the models used in isolation. It should be clarified that the use of 3 classifiers in the experiments was not a case of ensemble detection because the classifiers operated on different datasets generated through different features. While that too was effective in improving the accuracy of imposter detection, ensemble detection has not been tried out in this research and it remains to be explored.

Several variants of ensemble learning exist and one common application is to use the same data source or its simple variants to train different prediction models. When applied to the system under reference, datasets of the angle of departure between dwell time vectors could be used to train 3 different models, say Logistic Regression, Random Forest and Support Vector Machines. The predictions of all 3 are combined using some criterion such as majority vote or weighted averaging. The expectation is that the strengths of all 3 models will be combined while their weaknesses will be offset.

Further, more advanced concepts of machine learning such as the use of ‘deep neural networks’ could also be applied to realize more effective classification.

7.4.3 Profile updating

In the review of related work, the time variance of biometrics in general was discussed. When keystroke dynamics are considered, the likelihood for the biometrics to change with time is obvious. As a person becomes more and more conversant with a keyboard, the typing characteristics would also become more refined. The speed will generally improve, with an accompanying reduction of flight times and even the dwell times. Non-temporal features would also improve. On the other hand, as a person grows older, degradation of vision and flexibility of hand muscles would cause the typing skills to diminish.

As a result, the profiles created at a certain point in time need to be updated periodically so that they reflect the current typing traits of the candidate. Instead of having regular mock tests, the online tests that the candidate take could be used as opportunities to acquire latest biometrics. Therefore, during the online tests, while authentication is in progress, the acquired biometrics may be used to update the features stored in the profiles, to create new datasets, and to retrain the classifiers on the updated biometrics. This should be done only if the overall test taking has been rated as genuine.

Profile updating is not new and is a standard feature in all biometrics based authentication systems. However, unlike in other online applications, online tests provide more opportunities for regular profile updating due to the larger volume of biometrics that could be captured during the online tests themselves.

One popular update strategy that is easily employed in keystroke dynamics based systems is 'self-updating' where the system creates a newer profile of the subject from the authentication data acquired during a session where the user has been ascertained as genuine and authentic with high confidence, which is then used to overwrite the old profile. Alternatively, supervised learning could be used using labelled keystroke data derived from an authentically confident session, to retrain the existing model. Both schemes are readily adaptable to scenarios of continuous authentication during online tests.

Building suitable unsupervised profile update strategies particularly for continuous authentication systems used in online tests has research scope.

7.4.4 Mitigation strategies

The concept proposed in this research dealt only with the detection of fraudulent practices during online tests. How the test administration should react to an anomaly detection is another area that requires consideration. It is also vital in a practical implementation of the concept.

In Chapter 4, the difference in the mitigation strategies used for authentication during online tests and those used in other security applications was discussed. Access restricted online services usually employ ruthless mitigation strategies when intrusion

is detected. The inability to use such strategies in online tests was also explained in Chapter 4.

Instead of resorting to denial of services, an approach more suited for online tests would be to permit the fraudulent test taking to continue while recording the anomaly detections in a manner that they could be used as solid evidence for administrative action against the fraudulent test taking at a later stage. The use of XAI mentioned in 6.6.2 is also relevant here.

Mitigation strategies particularly suited for online tests is a relatively unexplored subject. The strategy should ensure that the reaction of the system in the event of an anomaly detection does not unduly deter a genuine candidate while catching an imposter or a cheating candidate off-guard.

7.4.5 Population analysis

The profile creation envisaged in the concept treated candidates as individual subjects and all profiles were created using only the biometrics of the individual.

Population analysis refers to the systematic study of groups of persons, in this case the candidates, to identify their group traits. As an example, candidates coming from a particular geographical area or educational background may display a higher level of language proficiency which would be reflected in their keystroke dynamics. Candidates belonging to certain age groups may exhibit less tendency to indulge in fraud. Candidates who are skilled in touch typing would have distinct keystroke dynamics due to their prompt typing sans hunt and peck. By collecting additional data that aids in classifying the candidates into such groups, and identifying the candidate population as a collection of groups, certain traits may be extended across the profiles of all the candidates in a group even when the biometrics do not exhibit them individually.

This opens doors for new methods of profile creation which would ultimately result in better anomaly detection. Hence, that would comprise an important research area.

7.4.6 Device dependence

The dependence of keystroke dynamics on the device used for test taking was discussed in the previous chapters.

While it is advantageous for authentication systems to be device independent, the effect of different devices on authentication during online tests is something that merits exploration.

Researching the effect of touchscreen keyboards could also be a part of this study. The keystroke dynamics exhibited by the same individual when typing on a physical keyboard and when using a touchscreen keyboard could be significantly different. How the keystroke loggers used with touchscreen keyboards sense the key presses is

also different. Since the trend towards using smart devices for test taking is increasing, extending the scope of this research to include smart devices would be worthwhile.

7.4.7 Adaptation of the system to other languages

Typing is inevitably language dependent. Many languages and especially those that do not use the Roman alphabet, use extended character sets which require peculiar key combinations to generate the characters. The keystroke dynamics will be greatly influenced by the language used. The same person may exhibit radically different keystroke biometrics when typing in different languages.

The features to be extracted for authentication would also differ. For example, hot sequences in English will be different to the hot sequences in other languages using the Roman alphabet like French or German.

Tailoring the system to cater to different languages would be a useful development for continuous authentication systems used in online tests.

7.5 Recapitulation

This chapter was devoted to presenting the overall conclusion of the research, contributions of the research, and for the suggestion of directions for future work.

The sincere expectation of the researcher is that future work built upon this study and inspired by the suggested directions as well as other independent work would contribute towards strengthening the capabilities of test administering institutions to effectively manage the integrity of online assessments and thereby maintain true merit systems within the academia.

REFERENCES

[1] M. Akintolu and S. E. Adewoye, "Systematic Review of the Role of Open and Distance Education in Achieving the Sustainable Development Goals (SDGs)," *Int. J. Learn. Teach. Educ. Res.*, vol. 23, no. 12, pp. 113–129, Dec. 2024.

[2] A. Afolabi, H. A. Fatoye, A. Abiodun, and A. A. Adewole, "Emerging opportunities in the open and distance education and e-learning platforms in Nigeria," in *Proc. Int. Stakeholders' Conf. Enhancing Inclusive Equitable Quality Educ. Through Open Distance Educ.*, Ibadan, Nigeria, Dec. 2021,

[3] A. Qayyum and O. Zawacki-Richter, *Open and Distance Education in a Digital Age* (SpringerBriefs in Education). Cham, Switzerland: Springer, 2018, pp. 1–8.

[4] "Assessing large cohorts," Bristol Inst. Learn. Teach. Blog, Oct. 14, 2019. [Online]. Available: https://bilt.online/assessing-large-cohorts-2/ (accessed Feb. 12, 2025).

[5] P. van Vliet, "Scaling up student assessment: Issues and solutions," DigitalCommons@University of Nebraska - Lincoln, 2018. [Online]. Available: http://digitalcommons.unl.edu/ipts/27 (accessed Feb. 12, 2025).

[6] O. L. Holden, M. E. Norris, and V. A. Kuhlmeier, "Academic integrity in online assessment: A research review," *Front. Educ.*, vol. 6, pp. 1–13, Jul. 2021.

[7] J. R. Young, "Online Classes See Cheating go high-tech," *Chron. High. Educ.*, Jun. 2012. [Online]. Available:https://www.chronicle.com/article/online-classes-see-cheating-go-high-tech/

[8] J. Adkins, C. Kenkel, and C. L. Lim, "Deterrents to online academic dishonesty," *Int. J. Learn. High. Educ.*, vol. 1, no. 1, pp. 17–22, Jan. 2005. [Online]. Available:https://digitalcommons.butler.edu/cgi/viewcontent.cgi?article=1190&context=cob_papers

[9] D. Vincent, "Promoting Academic Integrity in Assessment in Online Distance Learning," Technology-Mediated Learning, Sep. 22, 2013. [Online]. Available: https://journals.library.mun.ca/index.php/TML/article/view/835

[10] T. B. Michael and M. A. Williams, "Student equity: Discouraging cheating in online courses," *Admin. Issues J. Educ. Pract. Res.*, vol. 3, no. 2, pp. 1–12, Oct. 2013.

[11] N. Chadaga, "Lockdown browser," HackerEarth, 2024. [Online]. Available: https://www.hackerearth.com/recruit/glossary/lockdown-browser/ (accessed Mar. 15, 2024).

[12] K. Mugambi, J. Sansa-Otim, and T. Bulega, "Student and e-assessment authentication in distance learning," *RUFORUM Working Document Series*, no. 14, pt. 3, pp. 65–70, 2016.

[13] D. Temoshok, J. Fenton, Y.-Y. Choong, N. Lefkovitz, A. Regenscheid, R. Galluzzo, and J. Richer, *Digital Identity Guidelines - Authentication and Authenticator Management*, NIST Special Publication SP 800-63B-4, Jul. 2025.

[14] N. Chourasia and P. Joeg, "Authenticate user by keystroke dynamics," *Int. J. Eng. Res. Technol. (IJERT)*, vol. 3, no. 4, pp. 1239–1241, Apr. 2014.

[15] A. A. E. Ahmed and I. Traoré, "A new biometric technology based on mouse dynamics," *IEEE Trans. Dependable Secure Comput.*, vol. 4, no. 3, pp. 165–179, Jul.–Sep. 2007.

[16] C. A. Fidas, M. Belk, A. Constantinides, D. Portugal, P. Martins, A. M. Pietron, A. Pitsillides, and N. Avouris, "Ensuring academic integrity and trust in online learning environments: A longitudinal study of an AI-centered proctoring system in tertiary educational institutions," *Educ. Sci.*, vol. 13, no. 6, p. 566, Jun. 2023.

[17] B. Fang, "Addressing academic dishonesty in the age of ubiquitous technology," EDUCAUSE Rev., Sep. 5, 2012. [Online]. Available: https://er.educause.edu/articles/2012/9/addressing-academic-dishonesty-in-the-age-of-ubiquitous-technology (accessed Jan. 3, 2025).

[18] D. L. McCabe, L. K. Trevino, and K. D. Butterfield, "Cheating in academic institutions: A decade of research," *Ethics Behav.*, vol. 11, no. 3, pp. 219–232, 2001.

[19] A. R. But, Z. Ahmad, R. F. Ali, and Q. Iqbal, "Academic integrity: A five-decade journey, from cheating to a lack of understanding of academic dishonesty," *J. Excell. Soc. Sci.*, vol. 3, no. 3, pp. 211–231, 2024.

[20] T. Bretag, S. Mahmud, M. Wallace, R. Walker, C. James, M. Green, J. East, U. McGowan, and L. Patridge, "Core elements of exemplary academic integrity policy in Australian higher education," *Int. J. Educ. Integr.*, vol. 7, no. 2, pp. 3–12, 2011.

[21] K. Sevnarayan and K. B. Maphoto, "Exploring the dark side of online distance learning: Cheating behaviours, contributing factors, and strategies to enhance the integrity of online assessment," *J. Acad. Ethics*, vol. 22, pp. 51–70, 2024.

[22] K. Mate and J. Weidenhofer, "Considerations and strategies for effective online assessment with a focus on the biomedical sciences," *FASEB Bioadv.*, vol. 4, no. 1, pp. 9–21, Oct. 2021.

[23] C. M. Kell, Y. Thandar, A. K. Bhundoo, F. Haffejee, B. Mbhele, and J. Ducray, "Academic integrity in the information age: insights from health sciences students at a South African University," *J. Appl. Res. High. Educ.*, vol. 17, no. 7, pp. 16–28, 2025.

[24] C. L. Huang, X. Shao, C. Wu, and S. C. Yang, "Navigating the digital learning landscape: insights into ethical dilemmas and academic misconduct among university students," *Int. J. Educ. Technol. High. Educ.*, vol. 22, Art. no. 29, 2025.

[25] C. K. Y. Chan, "Is AI changing the rules of academic misconduct? An in-depth look at students' perceptions of 'AI-giarism,'" *arXiv preprint arXiv:2306.03358*, 2023.

[26] National Center for Education Statistics, "Testing Integrity: Issues and Recommendations for Best Practice," U.S. Department of Education, Washington, DC, USA, Rep. NCES 2013-454, Feb. 2013. [Online]. Available: https://ies.ed.gov/use-work/resource-library/report/issue-brief/testing-integrity-issues-and-recommendations-best-practice

[27] S. Pudasaini, L. Miralles-Pechuán, D. Lillis, and M. L. Salvador, "Survey on plagiarism detection in large language models: The impact of ChatGPT and Gemini on academic integrity," *arXiv preprint arXiv:2407.13105*, 2024.

[28] D. Kundu, A. Mehta, R. Kumar, N. Lal, A. Anand, A. Singh, and R. R. Shah, "Keystroke dynamics against academic dishonesty in the age of LLMs," *arXiv preprint arXiv:2406.15335*, 2024.

[29] F. Sabrina, S. Azad, S. Sohail, and S. Thakur, "Ensuring academic integrity in online assessments: A literature review and recommendations," *Int. J. Inf. Educ. Technol.*, vol. 12, no. 1, pp. 60–68, Jan. 2022.

[30] R. Hamaamin, O. A. Ali, and S. Kareem, "Biometric systems: A comprehensive review," *Basrah J. Sci.*, vol. 42, no. 1, pp. 146–167, Apr. 2024.

[31] *Biometrics and Standards*, ITU-T Technology Watch Report, International Telecommunication Union, Geneva, Switzerland, Dec. 2009.

[32] F. Galton, "Personal identification and description," *J. Anthropol. Inst. Great Britain Ireland*, vol. 18, pp. 177–191, 1889.

[33] F. Galton, *Finger Prints*. London, UK: Macmillan & Co., 1892, p. 3. 1892

[34] R. B. Hill and R. L. Laub, "Method of identifying individuals by means of the pattern of blood vessels of the retina," U.S. Patent 3,937,953, Feb. 10, 1976.

[35] L. Flom and A. Safir, "Iris recognition system," U.S. Patent 4,641,349, Feb. 3, 1987.

[36] I. Alsaadi, "Physiological biometric authentication systems, advantages, disadvantages and future development: A review," *Int. J. Sci. Technol. Res.*, vol. 4, no. 12, pp.285-289, Oct. 2015.

[37] J. Galbally, J. Fierrez, R. Cappelli, and G. Marcialis, "Introduction to presentation attack detection in fingerprint biometrics," in *Handbook of Biometric Anti-Spoofing*, S. Marcel, J. Fierrez, and N. Evans, Eds. Cham, Switzerland: Springer, Seattle, WA, USA, 2022, pp. 3–16.

[38] M. Sharma and H. Elmiligi, "Behavioral biometrics: Past, present and future," in *Recent Advances in Biometrics*, BoD – Books on Demand, Mar.2022, ch. 4, pp. 1–20. DOI: 10.5772/intechopen.102841

[39] R. V. Yampolskiy and V. Govindaraju, "Behavioural biometrics: a survey and classification," *Int. J. Biom.*, vol. 1, no. 1, pp. 81–113, 2008.

[40] "Behavioural biometrics," ENISA, Europa.eu, 2025. [Online]. Available: https://www.enisa.europa.eu/publications/behavioural-biometrics (accessed Aug. 13, 2025).

[41] M. El-Abed, R. Giot, B. Hemery, and C. Rosenberger, "A study of users' acceptance and satisfaction of biometric systems," in *Proc. IEEE Int. Carnahan Conf. Secur. Technol. (ICCST)*, Location Unknown, Oct. 2010, pp. 170–178. doi.org/10.1504/IJBM.2008.018665

[42] P. S. Teh, A. B. J. Teoh, and S. Yue, "A survey of keystroke dynamics biometrics," *Sci. World J.*, vol. 2013, pp. 1–24, 2013.

[43] M. Helander, *A Guide to Human Factors and Ergonomics*. Boca Raton, FL, USA: CRC Press, 2005.

[44] O. Ciobanu, C. C. Gavat, and R. Cozmei, "The keyboard remains the least ergonomically designed computer device," in *Proc. 5th IEEE Int. Conf. E-Health Bioeng. (EHB)*, Iasi, Romania, Nov. 2015, pp. 1–4.

[45] S. P. Banerjee and D. L. Woodard, "Biometric authentication and identification using keystroke dynamics: A survey," *J. Pattern Recognit. Res.*, vol. 7, no. 1, pp. 116–139, Jan. 2012.

[46] R. Giot, M. El-Abed, and C. Rosenberger, "Keystroke dynamics overview," ResearchGate, Jun. 2011. [Online]. Available: https://www.researchgate.net/publication/221912833_Keystroke_Dynamics_Overview (accessed Feb. 12, 2025).

[47] H. Nonaka and M. Kurihara, "Sensing pressure for authentication system using keystroke dynamics," in *Proc. Int. Conf. Comput. Intell. (ICCI)*, Istanbul, Turkey, Dec. 2004, pp. 19-22.

[48] P. H. Pisani and A. C. Lorena, "A systematic review on keystroke dynamics," *J. Brazilian Comput. Soc.*, vol. 23, no. 5, pp. 428–440, Jul. 2013.

[49] B. Sayed, I. Traore, I. Woungang, and M. Obaidat, "Biometric authentication using mouse gesture dynamics," *IEEE Syst. J.*, vol. 7, no. 2, pp. 262–274, Jun. 2013.

[50] S. Khan, C. Devlen, M. Manno, and D. Hou, "Mouse dynamics behavioral biometrics: A survey,"*ACM Computing Surveys*, vol. 56, no. 6, Art. no. 154, Aug. 2022, pp.1-33, [Online]. Available: https://doi.org/10.1145/364031

[51] H. Gamboa and A. L. N. Fred, "A behavioral biometric system based on human-computer interaction," in *Proc. SPIE - Int. Soc. Opt. Eng.*, vol.5404, Aug. 2004, pp. 381-392.doi: 10.1117/12.541419.

[52] T.-Y. Chang, J. C. Tsai, Y.-J. Yang, and P.-C. Cheng, "User authentication using rhythm click characteristics for non-keyboard devices," in *Proc. Int. Conf. Asia Agriculture Animal (IPCBEE)*, Hong Kong, Jan. 2011, pp. 114-118.

[53] S. Cho, C. Han, D. H. Han, and H.-I. Kim, "Web based keystroke dynamics identity verification using neural network," *J. Organ. Comput. Electron. Commerce*, vol. 10, no. 4, pp. 295–307, Dec. 2000.

[54] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Lisbon, Portugal, Aug. 2009, pp. 23–32.

[55] A. Wahab, D. Hou, and S. Schuckers, "A user study of keystroke dynamics as second factor in web MFA," in *Proc. 13th ACM Conf. Data Appl. Secur. Privacy (CODASPY '23)*, Charlotte, NC, USA, Apr. 2023, pp. 61–72.

[56] B. Glass, G. Jenkinson, Y. Liu, M. A. Sasse, and F. Stajano, "The usability canary in the security coal mine: A cognitive framework for evaluation and design of usable authentication solutions," *arXiv preprint arXiv:1607.03394*, 2016.

[57] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication," *Future Gener. Comput. Syst.*, vol. 16, no. 4, pp. 351–359, Feb. 2000.

[58] R. Giot, M. El-Abed, and C. Rosenberger, "Keystroke dynamics with low constraints SVM based passphrase enrollment," in *Proc. IEEE 3rd Int. Conf. Biom.: Theory, Appl. Syst. (BTAS)*, Washington, DC, USA, 2009, pp. 1–6.

[59] I. Sluganovic, A. Karlovic, P. Bosilj, S. Horvat, and M. Sare, "User authentication based on keystroke dynamics analysis," in *Proc. 35th Int. Conv. MIPRO*, Opatija, Croatia, May 2012, pp. 2136–2141.

[60] L. C. F. Araújo, L. H. R. Sucupira, M. G. Lizarraga, L. L. Ling, and J. B. T. Yabu-Uti, "User authentication through typing biometrics features," *IEEE Trans. Signal Process.*, vol. 53, no. 2, pp. 851–855, Feb. 2005.

[61] A. K. Jain, S. C. Dass, and K. Nandakumar, "Soft biometric traits for personal recognition systems," in *Biometric Authentication* (ICBA 2004), D. Zhang and A. K. Jain, Eds. (Lecture Notes in Computer Science, vol. 3072). Berlin, Heidelberg, Germany: Springer, 2004, pp.731-738.

[62] O. Arigbabu, S. Ahmad, W. Adnan, S. Yussof, V. Iranmanesh, and F. Malalla, "Estimating body related soft biometric traits in video frames," *The Scientific World Journal*, vol. 2014, Art. no. 460973, Jul. 2014.

[63] R. Giot and C. Rosenberger, "A new soft biometric approach for keystroke dynamics based on gender recognition," *Int. J. Inf. Technol. Manage.*, vol. 11, no. 1/2, pp. 35-49, Jan. 2012.

[64] A. A. Buker, A. Vinciarelli, and G. Roffo, "Type like a man! Inferring gender from keystroke dynamics in live-chats," *IEEE Intell. Syst.*, vol. 34, no.6, pp. 53–59, Nov.-Dec. 2019. doi: 10.1109/MIS.2019.2948514.

[65] A. Pentel, "High precision handedness detection based on short input keystroke dynamics," Aug. *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Larnaca, Cyprus, 2017, pp. 1-5, doi: 10.1109/IISA.2017.8316380.

[66] S. Z. S. Idrus, E. Cherrier, C. Rosenberger, and P. Bours, "Soft biometrics database: A benchmark for keystroke dynamics biometric systems," in *Proc. Int. Conf. Biometrics Special Interest Group* (BIOSIG), Darmstadt, Germany, Jan. 2013, pp. 1–8.

[67] A. Kumar, A. Patwari, and S. Sabale, "User authentication by typing pattern for computer and computer based devices," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 3, no. 10, pp. 8132–8134, Oct. 2014.

[68] I. Tsimperidis, P. Yoo, K. Taha, A. Mylonas, and V. Katos, "R2BN: An adaptive model for keystroke-dynamics-based educational level classification," *IEEE Trans. Cybern.*, vol. 50, no. 11, pp. 4736–4746, Nov. 2020.

[69] C. Epp, M. Lippold, and R. L. Mandryk, "Identifying emotional states using keystroke dynamics," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.* (CHI), Vancouver, BC, Canada, May 2011, pp. 715–724.

[70] R. Giot, M. El-Abed, and C. Rosenberger, "Keystroke dynamics authentication for collaborative systems," in *Proc. IEEE Int. Symp. Collaborative Technol. Syst.* (CTS), Baltimore, MD, USA, Nov. 2009, pp. 172–179.

[71] R. Shadaman, A. Wahab, M. Manno, M. Lukaszewski, D. Hou, and F. Hussain, "Keystroke dynamics: Concepts, techniques, and applications," *arXiv preprint arXiv:2303.11186*, Mar. 2023.

[72] K. Niinuma and A. K. Jain, "Continuous user authentication using temporal information," in *Proc. SPIE 7667, Biometric Technology for Human Identification VII*, Orlando, FL, USA, Apr. 2010, Art. no. 76670A.

[73] I. C. Stylios, O. Thanou, I. Androulidakis, and E. Zaitseva, "A review of continuous authentication using behavioral biometrics," in *Proc. SouthEast Eur. Design Autom. Comput. Eng. Comput. Netw. Soc. Media Conf.* (SEEDA-CECNSM), Kastoria, Greece, Sep. 2016, pp. 1–8.

[74] C. C. Tappert, M. Villani, and S.-H. Cha, "Keystroke biometric identification and authentication on long-text input," in *Behavioral Biometrics for Human Identification: Intelligent Applications*, L. Wang and J. Geng, Eds. Hershey, PA, USA: IGI Global, Jan. 2009, pp. 342–367.

[75] H. Çeker and S. Upadhyaya, "User authentication with keystroke dynamics in long-text data," in *Proc. IEEE 8th Int. Conf. Biometrics Theory Appl. Syst.* (BTAS), Niagara Falls, NY, USA, 2016, pp. 1–6, doi: 10.1109/BTAS.2016.7791182.

[76] P. Bours and H. Barghouthi, "Continuous authentication using biometric keystroke dynamics," in *Proc. Norwegian Inf. Secur. Conf.* (NISK), Oslo, Norway, 2009, pp. 1–12.

[77] E. Alsolamy, "An examination of keystroke dynamics for continuous authentication," Doctoral dissertation, Inf. Secur. Inst., Queensland Univ. of Technol., Brisbane, QLD, Australia, 2012.

[78] T. Sim and R. Janakiraman, "Are digraphs good for free-text keystroke dynamics?," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* (CVPR), Minneapolis, MN, USA, Jul. 2007, pp. 1–6.

[79] H. Al-Assam, H. Sellahewa, and S. Jassim, "Accuracy and security evaluation of multi-factor biometric authentication," *Int. J. Inf. Secur. Res.*, vol. 1, no. 1, pp. 11–19, Mar. 2011.

[80] R. Giot, M. El-Abed, and C. Rosenberger, "GREYC keystroke: A benchmark for keystroke dynamics biometric systems," in *Proc. IEEE 3rd Int. Conf. Biometrics: Theory Appl. Syst.* (BTAS), Washington, DC, USA, 2009, pp. 1–6.

[81] T. Sim, "Continuous authentication - Still a thing after 20 years?," IEEE Biometrics Council Webinar, Jan. 17, 2024. [Online video]. Available: https://youtu.be/LOdowYjNDZA (accessed Feb. 15, 2025).

[82] P. Bours and S. Mondal, "Performance evaluation of continuous authentication systems," *IET Biometrics*, vol. 4, no. 4, pp. 220–226, Jun. 2015.

[83] D. Senarath, S. Tharinda, M. Vishvajith, S. Rasnayaka, S. Wickramanayake, and D. Meedeniya, "Re-evaluating keystroke dynamics for continuous authentication," in *Proc. 3rd Int. Conf. Adv. Res. Comput.* (ICARC), Belihuloya, Sri Lanka, 2023, pp. 202–207.

[84] M. Trojahn, F. Arndt, and F. Ortmeier, "Authentication with keystroke dynamics on touchscreen keypads-effect of different n-graph combinations," in *Proc. 3rd Int. Conf. Mobile Services, Resources, and Users* (Mobility), Lisbon, Portugal, Jan. 2013, pp. 69–74.

[85] N. Ahmad, A. Szymkowiak, and P. A. Campbell, "Keystroke dynamics in the pre-touchscreen era," *Front. Hum. Neurosci.*, vol. 7, Art. no. 835, Dec. 2013.

[86] I. Buhan and P. Hartel, "The state of the art in abuse of biometrics," Centre for Telematics and Information Technology (CTIT), University of Twente, Enschede, The Netherlands, Tech. Rep. TR-CTIT-05-34, Sep. 2005.

[87] C. Gottschlich, "Skilled impostor attacks against fingerprint verification systems and its remedy," *arXiv preprint arXiv:1503.04729*, 2015.

[88] S. Coghlan, T. Miller, and J. Paterson, "Good proctor or 'big brother'? Ethics of online exam supervision technologies," *Philos. Technol.*, vol. 34, no. 4, pp. 1581–1606, Dec. 2021.

[89] K. Becker and H. Meng, "Identifying statistically actionable collusion in remote proctored exams," *J. Appl. Test. Technol.*, vol. 23, no. S1, pp. 54–61, Mar. 2022.

[90] D. L. McCabe and L. K. Treviño, "Academic dishonesty: Honor codes and other contextual influences," *J. High. Educ.*, vol. 64, no. 5, pp. 522–538, Sep. 1993.

[91] J. Liebers, C. Burschik, U. Gruenefeld, and S. Schneegass, "Exploring the stability of behavioral biometrics in virtual reality in a remote field study: Towards implicit and continuous user identification through body movements," *arXiv preprint arXiv:2310.15003*, Oct. 2023.

[92] R. Giot, B. Dorizzi, and C. Rosenberger, "Analysis of template update strategies for keystroke dynamics," in *Proc. IEEE Workshop Comput. Intell. Biometrics Identity Manage.* (CIBIM), Paris, France, May 2011, pp. 1–8.

[93] P. H. Pisani, A. C. Lorena, and A. C. P. L. F. de Carvalho, "Adaptive biometric systems using ensembles," *IEEE Intell. Syst.*, vol. 33, no. 2, pp. 19–28, Mar./Apr. 2018.

[94] N. Poh, A. Rattani, and F. Roli, "Critical analysis of adaptive biometric systems," *IET Biometrics*, vol. 1, no. 4, pp. 191–199, Dec. 2012.

[95] H. Bhatt, S. Bharadwaj, R. Singh, M. Vatsa, A. Noore, and A. Ross, "On co-training online biometric classifiers," in *Proc. IEEE Int. Joint Conf. Biometrics* (IJCB), Washington, DC, USA, Oct. 2011, pp. 1–7.

[96] T. Susnjak and T. McIntosh, "ChatGPT: The end of online exam integrity?," *Educ. Sci.*, vol. 14, no. 6, Art. no. 656, Jun. 2024.

[97] P. Santoki, "Design and implementation of detection of key logger," *Int. J. Eng. Dev. Res.*, vol. 2, no. 2, pp. 1999–2017, Jun. 2014.

[98] S. P. Bejo, B. Kumar, P. Banerjee, P. Jha, A. N. Singh, and M. K. Dehury, "Design, analysis and implementation of an advanced keylogger to defend cyber threats," *International Journal of Computer Sciences and Engineering*, vol. 11, no. 3, pp. 12–18, Mar. 2023.

[99] A. Bhardwaj and S. Goundar, "Keyloggers-silent cyber security weapons," *Netw. Secur.*, vol. 2021, no. 7, pp. 13–19, Jul. 2021.

[100] R. M. Martín and M. Apfelthaler, "A task segment framework to study keylogged translation processes," *Int. J. Transl. Interpreting Res.*, vol. 14, no. 2, pp. 8–31, Jul. 2022.

[101] C. Ritzer, "German court: Monitoring of employees by key logger is not allowed," Data Protection Report, Aug. 08, 2017. [Online]. Available: https://www.dataprotectionreport.com/2017/08/german-court-monitoring-of-employees-by-key-logger-is-not-allowed/ (accessed Apr. 29, 2025).

[102] A. Wengelin and V. Johansson, "Investigating writing processes with keystroke logging," in *Digital Writing Technologies in Higher Education: Theory, Research, and Practice*, M. Leijten and L. Van Waes, Eds. Cham, Switzerland: Springer, 2023, pp. 405–420.

[103] Y. Atoum, L. Chen, A. X. Liu, S. D. H. Hsu, and X. Liu, "Automated online exam proctoring," *IEEE Trans. Multimedia*, vol. 19, no. 7, pp. 1609–1624, Jul. 2017.

[104] O. Holden, M. E. Norris, and V. A. KUKuhlmeier, "Academic integrity in online assessment: A research review," *Front. Educ.*, vol. 6, Art. no. 639814, Mar. 2021.

[105] M. Torrance and D. Galbraith, "The processing demands of writing," in *Handbook of Writing Research*, C. MacArthur, S. Graham, and J. Fitzgerald, Eds. New York, NY, USA: Guilford Publications, 2006, pp. 67–80.

[106] Y. Deng and Y. Zhong, "Keystroke dynamics user authentication based on Gaussian mixture model and deep belief nets," *ISRN Signal Process.*, vol. 2013, Art. no. 565843, Jan. 2013.

[107] C. Boutin, Ed., "NIST to turn atomic clocks ahead one second on June 30," National Institute of Standards and Technology, Tech. Rep., Aug. 06, 2009. [Online]. Available: https://www.nist.gov/pml/time-and-frequency-division/time-realization/leap-seconds (accessed Aug. 13, 2024).

[108] B. Keating, "Letter frequencies in the English language," University of Notre Dame. [Online]. Available: https://www3.nd.edu/~busiforc/handouts/cryptography/Letter%20Frequencies.html (accessed Jul. 04, 2024).

[109] J. Huang, D. Hou, S. Schuckers, T. Law, and A. Sherwin, "Benchmarking keystroke authentication algorithms," in *Proc. IEEE Workshop Inf. Forensics Secur.* (WIFS), Rennes, France, 2017, pp. 1–6.

[110] MathWorks, "Choosing the best machine learning classification model and avoiding overfitting," MathWorks Corporate Website, 2024. [Online]. Available: https://www.mathworks.com/campaigns/offers/next/choosing-the-best-machine-learning-classification-model-and-avoiding-overfitting.html (accessed Feb. 15, 2025).

[111] M. Awad and R. Khanna, *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Berkeley, CA, USA: ApressOpen, 2015.

[112] G. Tubin, "Keylogging: How it works, impact, and 5 defensive measures," Cynet, Jul. 22, 2024. [Online]. Available: https://www.cynet.com/cybersecurity/keylogging-how-it-works-impact-and-5-defensive-measures/ (accessed Feb. 15, 2025).

[113] C. Murphy, J. Huang, D. Hou, and S. Schuckers, "Shared dataset on natural human-computer interaction to support continuous authentication research," in *Proc. IEEE Int. Joint Conf. Biometrics* (IJCB), Denver, CO, USA, 2017, pp. 525–530.

[114] Y. Sang, H. Shen, and P. Fan, "Novel impostors detection in keystroke dynamics by support vector machine," in *Proc. 5th Int. Conf. Parallel Distrib. Comput. Appl. Technol.* (PDCAT), Singapore, Dec. 2004, pp. 624–628.

[115] E. Yu and S. Cho, "Keystroke dynamics identity verification - Its problems and practical solutions," *Comput. Secur.*, vol. 23, no. 5, pp. 428–440, Jul. 2004.

[116] K. S. Balagani, V. V. Phoha, A. Ray, and S. Phoha, "On the discriminability of keystroke feature vectors used in fixed text keystroke authentication," *Pattern Recognit. Lett.*, vol. 32, no. 8, pp. 1070–1080, Jun. 2011.

BIBLIOGRAPHY

[1] R. Kumar, *Research Methodology: A Step-by-Step Guide for Beginners*. New Delhi, India: Sage Publications India Pvt. Ltd., 2014.

[2] T. Lucey, *Quantitative Techniques*, 5th ed. UK: ELST, 2001.

[3] K. Thompson and D. M. Ritchie, *UNIX Programmer's Manual*, 3rd ed. Bell Telephone Laboratories Inc., Feb. 1973.

[4] P. Sharma, *Fundamentals of Web Technology*, 1st ed. S K Kataria & Sons, 2013.

[5] D. Goodman and M. Morrison, *JavaScript Bible*, 5th ed. USA: Wiley Publishing Inc., 2004.

[6] G. Crisp, *The e-Assessment Handbook*, 1st ed. New York, NY, USA: Continuum International Publishing Group, 2008.

- END -

APPENDIX - A

ALGORITHMS USED IN THE EXPERIMENTS

Algorithm – 1: Extraction of dwell time vector from biometrics:

```
// Get registered user's identity as a command line argument
RegID ← command line argument 1


// Open biometrics file of registered user in Test 01 and read contents into
dictionary 'key_presses'
key_presses ← load_from_json_file(biometrics file of RegID of Test 01)

set buffer_size equal to length_of dictionary 'key_presses'

// Initialize matrix to hold key event data

Number_of_rows = buffer_size
Number of columns = 3 {
        Column 0 to hold the 'key' of data type 'string' initialized to NULL
        Column 1 to hold the 'timestamp' of data type 'integer' initialized to 0
        Column 2 to hold the 'event' of data type 'integer' initialized to 0
        }

//Populate the matrix

for i in range (buffer_size) do
        key[i] ← key_presses[i]['key']
        timestamp[i] ←
convert_string_to_integer(key_presses[i]['timestamp'])
        if (key_presses[i]['type']=='KeyDown')
                event[i] ← -1
        if (key_presses[i]['type']=='KeyUp__')
                event[i] ← 1


// Create array of unit vectors
unit_vectors ← ['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G','H',
                'I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y',
                'Z',' ','.',',','[',']','/',"\\",'=','-',';','`','TAB',"CR","BS","SHIFT"]


directions ← length_of(unit_vectors)
```

```
// initialize vectors to hold mean and standard deviation of dwell time
mu_dwell ← [1D array of float values of length equal to directions,
                                                    initialized with 0.0]
sigma_dwell ← [1D array of float values of length equal to directions,
                                                    initialized with  0.0]

// Populate mu_dwell[] in the first run
for each value of the index in array unit_vectors[] do
        extract a tuple of indices having that unit vector from key_presses
        initialize depress_count
        initialize dwell
        initialize dwell_total

         if (there are presses of that particular key (i.e. tuple not NULL))
                for each item pointed by the indices in the tuple do
                        check whether it is a KeyDown
                        if next item is a keyup
                                dwell ← difference in timestamp values
                                dwell_total =  dwell_total + dwell
                                depress_count = depress_count + 1

                mu_dwell = dwell_total/depress_count


// Populate sigma_dwell

for each value of the index in array unit_vectors[] do
        extract a tuple of indices having that unit vector from key_presses
        initialize depress_count
        initialize dwell
        initialize variance
        initialize variance_total

        if (there are presses of that particular key (i.e. tuple not NULL))
                for each item pointed by the indices in the tuple do
                        check whether it is a KeyDown
                        if next item is a keyUp
                           variance ← (difference in timestamp values -
mu_dwell of current index])^2
                                variance_total =  variance_total + variance
                                depress_count = depress_count + 1

                 sigma_dwell = (variance_total / depress_count)^(0.5)
// Populate mu_dwell[] in the second run by eliminating outliers

outlier_flag = 0        // Initially assume that there are no outliers
```

```
for each value of the index in array unit_vectors[] do
        extract a tuple of indices having that unit vector from key_presses
        initialize depress_count
        initialize dwell
        initialize dwell_total

         if (there are presses of that particular key (i.e. tuple not NULL))
                for each item pointed by the indices in the tuple do
                        check whether it is a KeyDown
                        if next item is a keyup
                                dwell ← difference in timestamp values
                                threshold = 3*sigma_dwell
                                offset = abs(dwell – mu_dwell)
                                if (offset > threshold AND unit_vector is NOT
'SHIFT')
                                        dwell = mu_dwell
                                        outlier_flag = 1    // at least 1 outlier found
                                dwell_total =  dwell_total + dwell
                                depress_count = depress_count + 1
                mu_dwell = dwell_total/depress_count


// repopulate sigma_dwell[] if there were outliers in mu_dwell[]
if (outlier_flag == 1)
        for each value of the index in array unit_vectors[] do
                extract a tuple of indices having that unit vector from
key_presses
                initialize depress_count
                initialize dwell
                initialize variance
                initialize variance_total

                if (there are presses of that particular key (i.e. tuple not NUL))
                        for each item pointed by the indices in the tuple do
                                check whether it is a KeyDown
                                if next item is a keyup
                                        variance ←  (difference in timestamp values
                                - mu_dwell of current index])^2
                                        variance_total =  variance_total + variance
                                        depress_count = depress_count + 1

                         sigma_dwell = (variance_total / depress_count)^(0.5)
save mu_dwell[] as both .npy and .csv in profile folder of RegID
save sigma_dwell[] as both .npy and .csv in profile folder of RegID
close all files
```

Algorithm – 2: Extraction of pause duration ensemble from biometrics:

```
// Get registered user's identity as a command line argument
RegID ← command line argument 1


// Open biometrics file of registered user and read contents into
//  dictionary  'key_presses'
key_presses ← load_from_json_file(biometrics file of RegID)
set buffer_size equal to length_of dictionary 'key_presses'

// Initialize  matrix to hold key event data

Number_of_rows = buffer_size
Number of columns = 3 {
        Column 0 to hold the 'key' of data type 'string' initialized to NULL
        Column 1 to hold the 'timestamp' of data type 'integer' initialized to 0
        Column 2 to hold the 'event' of data type 'integer' initialized to 0
        }

//Populate the matrix

for i in range (buffer_size) do
        key[i] ← key_presses[i]['key']
        timestamp[i] ←
convert_string_to_integer(key_presses[i]['timestamp'])
        if (key_presses[i]['type']=='KeyDown')
                event[i] ← -1
        if (key_presses[i]['type']=='KeyUp__')
                event[i] ← 1

// Initialize array to hold pause durations ensemble
pause_times = [1-D array of 5000 elements holding 0]

j = 2
n = 0
while (j < buffer_size – 3) do
      pause_start = pause_end = 0
      previous_key = next_key = b' '                  // NULL byte
      if event is a KeyDown and it is a SPACE
        k = j - 1                              // backtrack to previous event
        while previous event is not a KeyUp do
          k = k - 1                            // Backtrack to the previous KeyUp
        if it is not a WHITESPACE
          pause_start = timestamp[j][0]
```

```
        previous_key = key[k][0]
        k = j + 1                          // point to event next to event 'j'
        while next event is not a KeyDown of a SPACE do
          k = k + 1                        // Advance to next KeyDown
        if it is not a WHITESPACE or NULL
          pause_end = timestamp[k][0]
          next_key = key[k][0]

      if (pause_end - pause_start > 0)   // We take only non-zero positive values
        pause_times[n] = pause_end - pause_start
        n = n + 1

      j = j + 1        // Repeat the while loop


trim trailing zeros of pause_times[]
save pause_times[] as both .npy and .csv in profile folder of RegID
```

Algorithm – 3: Extraction of flight time vectors (generic) from biometrics:

// This algorithm may be used to extract any of the 4 flight time vectors
// Dictionary dict[] must be defined appropriately to match the vector type
// For quick sequences it must be defined dynamically

// Get registered user's identity as a command line argument
RegID ← command line argument 1

// Open biometrics file of registered user and read contents into dictionary 'key_presses'
key_presses ← load_from_json_file(biometrics file of RegID)
set buffer_size equal to length_of dictionary 'key_presses'

// Initialize matrix to hold key event data

Number_of_rows = buffer_size
Number of columns = 3 {
    Column 0 to hold the 'key' of data type 'string' initialized to NULL
    Column 1 to hold the 'timestamp' of data type 'integer' initialized to 0
    Column 2 to hold the 'event' of data type 'integer' initialized to 0
    }

//Populate the matrix

for i in range (buffer_size) **do**
    key[i] ← key_presses[i]['key']
    timestamp[i] ← convert_string_to_integer(key_presses[i]['timestamp'])
    if (key_presses[i]['type']=='KeyDown')
        event[i] ← -1
    if (key_presses[i]['type']=='KeyUp__')
        event[i] ← 1

define dictionary of 2-character sequence as dict[]
initialize 3-D array to hold flight times as flight[][][] // a slice for each type of flight time

set dimensions equal to length of dict[] // Dimensionality of the vector
row_indices = [1-D array of integers of size 'dimensions' initialized to 0]

first_char = ' '
second_char = ' '
2key_sequence = ' '

```
j = 0
while (j < buffer_size – 3) do
  t1 = t2 = t3 = t4 = 0

  if it is a 'KeyDown'
        first_char = key[j][0]
        k = j                    // save loop counter
        t1 = timestamp[k][0]  // record the timestamp
        k = k + 1                // Move to next item in event data

        if next event is also a 'KeyDown'
          get value of k by invoking function proc1(k, buffer_size -3)
        else                     // If next event is a ‘KeyUp’
          t2 = timestamp[k][0]
          invoke function proc2(k, buffer_size -3)

        if first_char and second_char form a 2-key sequence of the required
kind
          row = column = 0
          2key_sequence = (first_char+second_char).decode('utf-8')
          // Now we have the sequence as a two character string
          // reverse lookup its key in dictionary dict[] to find the row number
in 3-D array
          m = 0
          for m in range(dimensions) do
            if (dict[str(m)]==2key_sequence)
              row = m

          column = row_indices[row]

          // Store flight time of the 1st type, in slice 0 | We avoid cases
          // where either of the 't' are zero
          if (t2 != 0 and t3 != 0)
            flight[0][row][column] = t3 - t2
          // Store flight time of the 2nd type, in slice 1
          if (t1 != 0 and t3 != 0)
            flight[1][row][column] = t3 - t1
          // Store flight time of the 3rd type, in slice 2
          if (t2 != 0 and t4 != 0)
            flight[2][row][column] = t4 - t2
          // Store flight time of the 4th type, in slice 3
          if (t1 != 0 and t4 != 0)
            flight[3][row][column] = t4 - t1
```

```
        j = j + 1          // increment loop counter to repeat the while loop

// Define new 3-D array to hold truncated flight times
max_row_length = row_indices[maximum value in array row_indices[]]
initialize 3-D array to hold flight times as flight_new[][][]


s = r = c = 0
for s in range(4) do                                    // slice
      for r in range(len(dict[])  do                    // row in slice
         for c in range(max_row_length) do              // cell on row
            flight_new[s][r][c]=flight[s][r][c]

// Eliminate outliers
s = r = c = 0
   mu = 0.0
   std = 0.0
   for s in range(4) do                          // slice
      for r in range(len(dict[]) do              // row in slice
         remove the zeros of row flight_new[s][r]
         mu ← mean of row flight_new[s][r]
         std ← standard deviation of row flight_new[s][r]
         for c in range(max_row_length) do  // cell on row
            if (flight_new[s][r][c] >= mu+3*std OR flight_new[s][r][c] <= mu-
3*std)
               flight_new[s][r][c] = mu

// The main algorithm ends here
// What follows are the pseudo code of the specific functions called by the
main algorithm



function proc1()  ------------------------------------------------------------------------
// This is called when it is detected that 1st key is being held down
// Accepts two parameters passed as k and limit
// Returns one parameter 'k'

 if it is the case of 1st key being held down
      advance along array in search of KeyUp by incrementing k
      t2 = timestamp[k][0]
      k = proc2(k,limit)
else                          // If KeyDown was due to pressing of 2nd key
      k = proc3(k,limit)
return(k)
```

```
// ---------------------------------------------------------------------------------------------
-----

function proc2()  ------------------------------------------------------------------------
// called from proc1 when 1st Key has been pressed and released
// Accepts two parameters passed as k and limit
// Returns one parameter 'k'

k = k + 1

if next event is a KeyDown
      second_char = key[k][0]
      t3 = timestamp[k][0]
      k = k + 1                  // Look at next event
      if that is also a KeyDown
         if(key[k][0] == second_char)   // If it is the case of 2nd key being held
down
            go in search of next KeyUp by incrementing k
            t4 = timestamp[k][0]
         else                    // If it is the case of a 3rd key been pressed
            k = proc5(k,limit)
      if it is a KeyUp
           t4 = timestamp[k][0]

   return(k)

// ---------------------------------------------------------------------------------------------
-----


function proc3()  ------------------------------------------------------------------------
// called from proc1 when two keys have been pressed consecutively |
// Accepts two parameters passed as k and limit
// Returns one parameter 'k'

k = k + 1            // Advance to next event
if next event is also a KeyDown // i.e. A 3rd key has been pressed before
releasing 1 or 2
      Go in search of next KeyUp by incrementing k
         if what was released is the 1st key
            t2 = timestamp[k][0]
            Go in search of next KeyUp by incrementing k
            if what was released is the 2nd key
               nop()
            else                  // if what was released is the 3rd key
```

```
            Go in search of next KeyUp by incrementing k
          t4 = timestamp[k][0]
        else
          if what was released is the 2nd key
            t4 = timestamp[k][0]
            Go in search of next KeyUp by incrementing k
            if what was released is the 1st key
              nop()
            if what was released is the 3rd key
              Go in search of next KeyUp by incrementing k
            t2 = timestamp[k][0]
          else                    // If what was released is the 3rd key
            Go in search of next KeyUp by incrementing k
            if what was released is the 2nd key
              t4 = timestamp[k][0]
            else                  // If what was released is the 1st key
              t2 = timestamp[k][0]

    else      // If next event was a KeyUp
      if what was released is the 1st key
        t2 = timestamp[k][0]
        advance to next event
        if it is a KeyUp
          t4 = timestamp[k][0]
        else                  // if a 3rd key has been pressed before releasing 2nd
key
          Go in search of next KeyUp by incrementing k
          if what was released is the 2nd key
              t4 = timestamp[k][0]
          else                  // If what was released is the 3rd key
              Go in search of next KeyUp by incrementing k
              t4 = timestamp[k][0]

      else     // if what was released is the 2nd key
        t4 = timestamp[k][0]
        k = k + 1
        if  next event is a KeyUp  // i.e. release of 1st key
          t2 = timestamp[k][0]
        else                  // If a 3rd key has been pressed before releasing 1st
key
          Go in search of next KeyUp by incrementing k
          if what has been released is the 1st key
            t2 = timestamp[k][0]
          else                // If what has been released is the 3rd key
            Go in search of next KeyUp by incrementing k
```

```
            t2 = timestamp[k][0]

   return(k)

// ------------------------------------------------------------------------------------------
-----

function proc4()  ------------------------------------------------------------------------
// This was not required in the present implementation
// Accepts two parameters passed as k and limit
// Returns 'k'  itself for scripts to work

 return(k)

// ------------------------------------------------------------------------------------------
----

function proc5()  ------------------------------------------------------------------------
// called from proc2 when two keys have been pressed consecutively
//  after release of 1st key
// Accepts two parameters passed as k and limit
// Returns one parameter 'k'

while event is a KeyDown do
        Go in search of next KeyUp by incrementing k
   if what is released is the 2nd key
        nop()
   else                        // If what is released is the 3rd key
        Go in search of next KeyUp by incrementing k
   t4 = timestamp[k][0]

   return(k)
// ------------------------------------------------------------------------------------------
-


function nop()  ------------------------------------------------------------------------
// This function does nothing
// Acts as a place holder for the conditional branches of the
// scripts to execute correctly

   return
// ------------------------------------------------------------------------------------------
-
```

Algorithm – 4: Generation of training data for genuine test taking:

// Training a classifier requires both genuine and anomalous data points
// This algorithm generates data points to be labelled as 'genuine'

player ← command line argument

Open JSON file of test 01 of *player* as dictionary array *key_events[]*
Retrieve vector *mu_dwell[] of player*
Initialize *mu_dwell_hat[]*
Initialize blank array *autotheta[]* to store *theta*

Point to the first dictionary element in *key_events[]*
Initialize the index of array *autotheta[]*
while the end of *key_events[]* is not reached **do**
    $m = 0$
    while $m < 40$ **do**
      Parse the array *key_events[]* populating *mu_dwell_hat[]* along the way
        $m = m + 1$
    Determine subspace of *mu_dwell_hat[]*
    Project *mu_dwell[]* into the subspace
    theta ← *angle (mu_dwell_hat[],projected mu_dwell[])*
    Push *theta* to array *autotheta[]*
    Label *theta* as *'genuine'*
    Advance the index of *autotheta[]* by 1
    Advance the index of *key_events[]* by 40

Save array *autotheta[]* by updating appropriate column of matrix *theta[][]* in folder of *player*

Close all open files

Algorithm – 5: Generation of training data for fraudulent test taking:

// Training a classifier requires both genuine and anomalous data points
// This algorithm generates data points to be labelled as 'fraudulent'
// by making some other player the test taker, so that the biometrics are
// deliberately mismatched, which will result in anomalous data points

player ← command line argument 1
other_player ← command line argument 2

Open JSON file of test 01 of *other_player* as dictionary array *key_events[]*
Retrieve vector *mu_dwell[] of player*
Initialize *mu_dwell_hat[]*
Initialize blank array *crosstheta[]* to store *theta*

Point to the first dictionary element in *key_events[]*
Initialize the index of array *crosstheta[]*
while the end of *key_events[]* is not reached **do**
    $m = 0$
    while $m < 40$ **do**
      parse the array *key_events[]* populating *mu_dwell_hat[]* along the way
        $m = m + 1$
    Determine subspace of *mu_dwell_hat[]*
    Project *mu_dwell[]* into the subspace
    theta ← *angle (mu_dwell_hat[],projected mu_dwell[])*
    Push *theta* to array *crosstheta[]*
    Label *theta* as '*fraudulent*'
    Advance the index of *crosstheta[]* by 1
    Advance the index of *key_events[]* by M dictionary elements

Save array *crosstheta[]* by updating appropriate column of matrix *theta[][]* in folder of *player*
Close all open files

Algorithm – 6: Simulation of genuine and impersonated test taking:

// This algorithm can be used to simulate both 'genuine' and 'fraudulent'
// test taking by impersonation, offline.
// Genuine test taking is simulated by making player_1 = player_2

player_1 ← command line argument 1
player_2 ← command line argument 2

Retrieve dwell_time vector of *player_1* as *mu_dwell[]*
Retrieve best classifier of player_1 for dwell time features as *model()*
Open JSON file of test 02 of *player_2* as dictionary array *key_events[]*

genuine ← TRUE
point to first dictionary element of *key_events[]*
while end of *key_events[]* is **not** encountered **do**
    create blank matrix *mu_dwell_hat[]* with structure similar to *mu_dwell[]*
    read *M* consecutive key events from *key_events[]*
    compute *mu_dwell_hat[]* of the *N* keys within the *M* events
    determine subspace of *mu_dwell_hat[]*
    project *mu_dwell[]* to subspace of *mu_dwell_hat[]*
    theta ← angle between *mu_dwell_hat[]* and projected *mu_dwell[]*
    genuine ← *model.predict(theta)*
    advance pointer of *key_events[]* by *M* dictionary elements

save file *genuine[]* as both .npy and .csv

close all files

Algorithm – 7: Simulation of fraudulent test taking:

// This algorithm can be used for the offline simulation of
// fraudulent acts of test taking other than impersonation
// namely, test taking while looking up and
// test taking while collaborating.
// Command line argument 'test' will determine the type of
// fraudulent act.
// For lookup, test = 03
// For collaboration, test = 04

player ← command line argument 1
test ← either '03' or '04' as command line argument 2

Retrieve dwell_time vector of *player* as *mu_dwell[]*
Retrieve best classifier of player for dwell time features as *model()*
Open JSON file of *test* of *player* as dictionary array *key_events[]*

genuine ← TRUE
point to first dictionary element of *key_events[]*
while end of *key_events[]* is **not** encountered **do**
    create blank matrix *mu_dwell_hat[]* with structure similar to *mu_dwell[]*
    read *M* consecutive key events from *key_events[]*
    compute *mu_dwell_hat[]* of the *N* keys within the *M* events
    determine subspace of *mu_dwell_hat[]*
    project *mu_dwell[]* to subspace of *mu_dwell_hat[]*
    theta ← angle between *mu_dwell_hat[]* and projected *mu_dwell[]*
    genuine ← *model.predict(theta)*
    advance pointer of *key_events[]* by *M* dictionary elements

save file *genuine[]* as both .npy and .csv

close all files

Algorithm – 8: Selection of anomalous data sources for a roleplayer:

// Since the training dataset needs to be balanced, in addition to N data
// points from the roleplayer, N/5 data points of another 5 roleplayers
// must be taken as anomalous data points to make the balanced dataset
// of size 2N.
// This algorithm is used to select the 5 other roleplayers who will be the
// source of anomalous data for the roleplayer under consideration.

RegID ← command line argument

// Initialize variables to hold the strings indicating the
// 5 roleplayers to be determined.

ImpID1 = ImpID2 = ImpID3 = ImpID4 = ImpID5 = ' '

Retrieve the theta matrix of *RegID* as *theta[][]*
Define dictionary of IDs as *dict[]*

Identify the column of *theta[][]* corresponding to *RegID* as *selftheta[]*
mu ← mean value of *selftheta[]*

// Initialize array to hold variance values
theta_var[][] ← {{ID, data_type: int, initially 0},
{variance, data_type:integer, initially 0}}
Number_of_rows = len(*dict[]*)

// Populate *theta_var[][]*
for *i* in range(len(*dict[]*)) **do**
theta_var[i][0] = i // This will be used to cross refer the ID
theta_var[i][1] = abs[*mu* – mean(*theta[i]*)) // absolute difference of means
theta_var[][] ← *theta_var[][]* sorted ascending on *theta_var[1]*

// Now the rows with smallest difference of mean values are on top,
// Row 0 will have 0, being the difference of mean of *RegID* with itself
// Next 5 rows will have the entries of 5 roleplayers with smallest difference

*ImpID1 = dict[*str(*theta_var[1][0])]*
*ImpID2 = dict[*str(*theta_var[2][0])]*
*ImpID3 = dict[*str(*theta_var[3][0])]*
*ImpID4 = dict[*str(*theta_var[4][0])]*
*ImpID5 = dict[*str(*theta_var[5][0])]*

Close all files

Algorithm – 9: Generation of training dataset for a roleplayer for a given feature:

```
// Since the training dataset needs to be balanced, in addition to N data
//  points from the roleplayer, N/5 data points of another 5 roleplayers
//  must be taken as anomalous data points to make the balanced dataset
//  of size 2N.
// This algorithm is used to build this dataset for a given feature
//  for the roleplayer under consideration.
// The anomalous data sources ImpID? must be determined beforehand
//  using Algorithm – 8. The standard output of Algorithm – 8 may be
//  redirected to a file which can be used to automate the execution of the
//  program that runs on this algorithm. As the default,
//  inputs are given as command line arguments.

RegID  ← command line argument 1
ImpID1 ← command line argument 2
ImpID2 ← command line argument 3
ImpID3 ← command line argument 4
ImpID4 ← command line argument 5
ImpID5 ← command line argument 6

// Initialize matrix to hold dataset for the given feature
dataset ← {{theta, data_type: float, initially 0.0},
           {label, data_type: integer, initially 0}}
Number_of_rows = 2*N

Retrieve theta matrix of RegID as theta[][]
Define dictionary of IDs as dict[]

// Identify the relevant columns of the players in theta[][]
for lc in dict do
    if (dict[lc]==RegID)
      genuine_h = int(lc)
    if (dict[lc]==Imp1)
      imp1_h = int(lc)
    if (dict[lc]==Imp2)
      imp2_h = int(lc)
    if (dict[lc]==Imp3)
      imp3_h = int(lc)
    if (dict[lc]==Imp4)
      imp4_h = int(lc)
    if (dict[lc]==Imp5)
      imp5_h = int(lc)
```

// Populate the *dataset* matrix

// First N rows
for *i* in range(N) **do**
    dataset [i][0] = theta[:,genuine_h][i]
    dataset [i][1] = 0 // Label = 0; genuine data point

// Next N rows
for *i* in range(N/5) **do**
    *dataset [*N + *i][0] = theta[:,Imp1_h][i]*
    *dataset [*N + *i][1] = 0* // Label = 1; anomalous data point
    *dataset [*N + N*/5* + *i][0] = theta[:,Imp2_h][i]*
    *dataset [*N + N*/5* + *i][1] = 0* // Label = 1; anomalous data point
    *dataset [*N + 2*N*/5* + *i][0] = theta[:,Imp3_h][i]*
    *dataset [*N + 2*N*/5* + *i][1] = 0* // Label = 1; anomalous data point
    *dataset [*N + 3*N*/5* + *i][0] = theta[:,Imp4_h][i]*
    *dataset [*N + 3*N*/5* + *i][1] = 0* // Label = 1; anomalous data point
    *dataset [*N + 4*N*/5* + *i][0] = theta[:,Imp5_h][i]*
    *dataset [*N + 4*N*/5* + *i][1] = 0* // Label = 1; anomalous data point

// Now the two column matrix *dataset[][]* is populated with first N rows
// labelled as 'genuine' and the second N rows labeled as 'fraudulent'
// The set is therefore balanced.

Save *dataset[][]* as both .csv and .npy in profile folder of *RegID*

Close all files

Algorithm – 10: Generic algorithm to train and test a classifier:

// This algorithm is used to train and test a classifier of given type
// for a given variant of training dataset for a given roleplayer

RegID ← command line argument 1
classifier_type ← command line argument 2
dataset_variant ← command line argument 3
test_size ← command line argument 4

retrieve the dataset specified by *dataset_variant* as *data[][]*

classifier ← model_train_test(*classifier_type,data[][],test_size*)

save *classifier* as .pk1 file in profile folder of *RegID*

Function model_train_test() ----------
// Accepts 3 parameters
// classifier type
// dataset
// Size of fraction of dataset to be used for testing

Split *data[][]* into training and testing set based on *test_size*
Determine the training model based on *classifier_type*
Fit the training data to the model
Predict the label using model and test data

return(model)

Algorithm – 11: Selection of best classifier for a vector based feature:

```
//  This algorithm is used to select the best classifier for a given
//    vector based feature for a given roleplayer

RegID ← command line argument 1
feature_type ← command line argument  2

// Open biometrics file of Test 01 of registered user and read contents
//  into dictionary 'key_presses'
key_presses  ←   load_from_json_file(biometrics file of RegID)

buffer_size = length_of(key_presses)

// Retrieve matrix holding details of classifiers for the feature_type
classifiers[]  ←   sliceof(LMdetails[],feature_type)

// Retrieve the feature vector corresponding to feature_type from profile
//  of RegID
vector[] ← feature vector corresponding to feature_type  in profile of RegID

// Initialize vector to hold classifier performance
classifier_performance ← {FRR, data_type: float, initially 0.0}
                                Number_of_rows = 16


// Simulate taking of Test 01 by RegID while authenticating
//  using each of the 16 classifiers in turn

for i in range(16) do
        model()  ← classifiers[i]
        // Initialize matrix to store simulation data
           sim_data ← {{theta, data_type: float, initially 0.0},
                       {genuine, data_type: Boolean, initially TRUE}}
        Number_of_rows = int(buffer_size/M + 1)

        genuine ← TRUE
        loop_count = 0
        point to first dictionary element of key_presses[]
        while end of key_presses[] is NOT encountered do
                create blank matrix vector_hat[] with structure similar to vector[]
                read M consecutive key events from key_ presses[]
                compute vector_hat[] of the N keys within the M events
                determine subspace of mu_dwell_hat[]
```

project *vector[]* to subspace of *vector_hat[]*
theta ← angle between *vector_hat[]* and projected *vector[]*
genuine ← *model.predict(theta)*
push *theta* and *genuine* into *sim_data[loop_count]*
advance pointer of *key_ presses[]* by **M** dictionary elements
increment *loop_count*
trim trailing zeros in sim_data[]
false_rejections = 0
j = 0
for j in range (loop_count) **do**
if *sim_data[genuine]* == FALSE
false_rejections = false_rejections + 1
FRR ← *false_rejections/loop_count*
classifier_performance[i] ← *FRR*

// Next, we select the element pointing to the best classifier
n ← index of smallest element of *classifier_performance[]*
best_classifier ← *classifiers[n]*
update entry of *RegID* in *best_classifiers* database with *best_classifier*

close all files

Algorithm – 12: Selection of best classifier for an ensemble based feature:

```
//  This algorithm is used to select the best classifier for a given
//   ensemble based feature for a given roleplayer

RegID ← command line argument 1
feature_type ← command line argument  2

// Open biometrics file of Test 01 of registered user and read contents
//  into dictionary 'key_presses'
key_presses  ←   load_from_json_file(biometrics file of RegID)

buffer_size = length_of(key_presses)

// Retrieve matrix holding details of classifiers for the feature_type
classifiers[]  ←   sliceof(LMdetails[],feature_type)

// Initialize vector to hold classifier performance
classifier_performance ← {FRR, data_type: float, initially 0.0}
                                Number_of_rows = 16


// Simulate taking of Test 01 by RegID while authenticating
//  using each of the 16 classifiers in turn

for i in range(16) do
        model() ← classifiers[i]
        // Initialize matrix to store simulation data
           sim_data ← {{mean_pause, data_type: float, initially 0.0},
                       {genuine, data_type: Boolean, initially TRUE}}
        Number_of_rows = int(buffer_size/M + 1)

        genuine ← TRUE
        loop_count = 0
        point to first dictionary element of key_presses[]
        while end of key_presses[] is NOT encountered do
                read M consecutive key events from key_ presses[]
                pause_count = 0
                pause_total = 0
                for each pause encountered within M key events do
                        compute pause                   // as in Algorithm 2
                        pause_total = pause_total + pause
                        pause_count = pause_count + 1
                mean_pause = pause_total/pause_count
```

```
            genuine ← model.predict(theta)
            push mean_pause and genuine into sim_data[loop_count]
            advance pointer of key_ presses[] by M dictionary elements
            increment loop_count
        trim trailing zeros in sim_data[]
        false_rejections = 0
        j = 0
        for j in range (loop_count) do
            if sim_data[genuine] == FALSE
                false_rejections = false_rejections + 1
        FRR ← false_rejections/loop_count
        classifier_performance[i] ← FRR

// Next, we select the element pointing to the best classifier
n ← index of smallest element of classifier_performance[]
best_classifier ← classifiers[n]
update entry of RegID in best_classifiers database with best_classifier

close all files
```

APPENDIX - B

INDEX OF SOURCE LISTINGS

Given below is a complete list of all computer program scripts that were written exclusively for use in the experiments, with a brief description of what each script does.

#	Name of source listing file	Type	Description
1	analCommonDatadwell.py	Python script	Performs statistical analysis on the data other than FAWI of the candidate, stored in the common data matrices
2	analFAWIdwell.py	Python script	Performs statistical analysis on the FAWI data of the candidate, stored in the common data matrices
3	analflight2.py	Python script	Analyzes the summarized results of flight time related data derived from simulations of the running of Test 02
4	analLMname.py	Python script	Extracts relevant details from the filename of learning machine
5	analperf.py	Python script	Provides a visual readout of the performance recorded by ‘performance.py’ on standard output
6	buildCombDetMat.py	Python script	Builds matrix with overall detection performance of best machines of all candidates when taking Test 02
7	buildCombDetMat3.py	Python script	Builds matrix with overall detection performance of best machines of all candidates when taking Test 03

#	Name of source listing file	Type	Description
8	buildCombDetMat4.py	Python script	Builds matrix with overall detection performance of best machines of all candidates when taking Test 04
9	buildLM3LR.py	Python script	Builds classifier of type Logistic Regression using TrainingSet3.csv
10	buildLM3LRwOE.py	Python script	Builds classifier of type Logistic Regression using TrainingSet3wOE.csv
11	buildLM3NB.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet3.csv
12	buildLM3NBwOE.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet3wOE.csv
13	buildLM3RF.py	Python script	Builds classifier of type Random Forest using TrainingSet3.csv
14	buildLM3RFwOE.py	Python script	Builds classifier of type Random Forest using TrainingSet3wOE.csv
15	buildLM3SVM.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet3.csv
16	buildLM3SVMwOE.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet3wOE.csv
17	buildLM4LR.py	Python script	Builds classifier of type Logistic Regression using TrainingSet4.csv
18	buildLM4LRwOE.py	Python script	Builds classifier of type Logistic Regression using TrainingSet4wOE.csv

#	Name of source listing file	Type	Description
19	buildLM4NB.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet4.csv
20	buildLM4NBwOE.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet4wOE.csv
21	buildLM4RF.py	Python script	Builds classifier of type Random Forest using TrainingSet4.csv
22	buildLM4RFwOE.py	Python script	Builds classifier of type Random Forest using TrainingSet4wOE.csv
23	buildLM4SVM.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet4.csv
24	buildLM4SVMwOE.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet4wOE.csv
25	buildLM5LR.py	Python script	Builds classifier of type Logistic Regression using TrainingSet5.csv
26	buildLM5LRwOE.py	Python script	Builds classifier of type Logistic Regression using TrainingSet5wOE.csv
27	buildLM5NB.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet5.csv
28	buildLM5NBwOE.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet5wOE.csv
29	buildLM5RF.py	Python script	Builds classifier of type Random Forest using TrainingSet5.csv

#	Name of source listing file	Type	Description
30	buildLM5RFwOE.py	Python script	Builds classifier of type Random Forest using TrainingSet5wOE.csv
31	buildLM5SVM.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet5.csv
32	buildLM5SVMwOE.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet5wOE.csv
33	buildLM6LR.py	Python script	Builds classifier of type Logistic Regression using TrainingSet6.csv
34	buildLM6LRwOE.py	Python script	Builds classifier of type Logistic Regression using TrainingSet6wOE.csv
35	buildLM6NB.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet6.csv
36	buildLM6NBwOE.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet6wOE.csv
37	buildLM6RF.py	Python script	Builds classifier of type Random Forest using TrainingSet6.csv
38	buildLM6RFwOE.py	Python script	Builds classifier of type Random Forest using TrainingSet6wOE.csv
39	buildLM6SVM.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet6.csv
40	buildLM6SVMwOE.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet6wOE.csv

#	Name of source listing file	Type	Description
41	buildLM8DTwCV.py	Python script	Builds classifier of type Decision Tree with Cross-Validation using TrainingSet8.csv (Not used)
42	buildLM8LR.py	Python script	Builds classifier of type Logistic Regression using TrainingSet8.csv
43	buildLM8LRwOE.py	Python script	Builds classifier of type Logistic Regression using TrainingSet8wOE.csv
44	buildLM8NB.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet8.csv
45	buildLM8NBwOE.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet8wOE.csv
46	buildLM8RF.py	Python script	Builds classifier of type Random Forest using TrainingSet8.csv
47	buildLM8RFwOE.py	Python script	Builds classifier of type Random Forest using TrainingSet8wOE.csv
48	buildLM8SVM.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet8.csv
49	buildLM8SVMwOE.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet8wOE.csv
50	buildLM9LR.py	Python script	Builds classifier of type Logistic Regression using TrainingSet9.csv
51	buildLM9LRwOE.py	Python script	Builds classifier of type Logistic Regression using TrainingSet9wOE.csv

#	Name of source listing file	Type	Description
52	buildLM9NB.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet9.csv
53	buildLM9NBwOE.py	Python script	Builds classifier of type Naïve Bayes using TrainingSet9wOE.csv
54	buildLM9RF.py	Python script	Builds classifier of type Random Forest using TrainingSet9.csv
55	buildLM9RFwOE.py	Python script	Builds classifier of type Random Forest using TrainingSet9wOE.csv
56	buildLM9SVM.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet9.csv
57	buildLM9SVMwOE.py	Python script	Builds classifier of type Support Vector Machine using TrainingSet9wOE.csv
58	calcFAWIdwell.py	Python script	Updates matrix performance.csv/.npy with FAWI of impersonations
59	calcFRWIdwell.py	Python script	Updates matrix performance.csv/.npy with FRWI of genuine test taking
60	checkflight.py	Python script	Produces report of flight time related features of all candidates
61	checkpausetimes.py	Python script	Extracts and saves pause times related parameters of all candidates
62	checkQQall.py	Python script	Displays QQall matrix (Auto and cross theta values) of a single candidate

#	Name of source listing file	Type	Description
63	constructFAWImatrix.py	Python script	Centralizes FAWI data of all candidates into a common matrix
64	createBestMachineMatrix.py	Python script	Creates empty 'Best Machines' matrix
65	createCombDetMat2_extended.py	Python script	build the blank matrix to store all performance parameters of Test 02 of all candidates taking test genuinely
66	createCombDetMat3_extended.py	Python script	build the blank matrix to store all performance parameters of Test 03 of all candidates taking test genuinely
67	createCombDetMat4_extended.py	Python script	build the blank matrix to store all performance parameters of Test 04 of all candidates taking test genuinely
68	createCombinedDetectionsFile.py	Python script	extract performance data when impersonation is monitored by all 3 best machines and builds a central database for given candidate
69	createCommonArrays.py	Python script	Creates empty common arrays to store performance metrics for all candidates
70	createCommonMetricsDatabase.py	Python script	Creates the blank metrics tables for Tests 03, 04
71	createLMdetailsMatrix.py	Python script	Builds a matrix containing details of all classifiers used

#	Name of source listing file	Type	Description
72	createMasterPerformanceFile2.py	Python script	Creates master performance database during impersonation for a given candidate for combined detection by all 3 classifiers
73	createSelfCombinedDetectionsFile.py	Python script	extract performance data when genuine taking of test 02 is monitored by all 3 best machines and builds a central database for all candidates
74	createSelfCombinedDetectionsFile_Test03.py	Python script	extract performance data when taking of test 03 is monitored by all 3 best machines and builds a central database for all candidates
75	createtr3n4.py	Python script	Creates training dataset 3 and 4 for given candidate
76	createtr3n4wOE.py	Python script	Creates training dataset 3 and 4 for given candidate with outliers eliminated
77	createtr5n6.py	Python script	Creates training dataset 5 and 6 for given candidate
78	createtr5n6wOE.py	Python script	Creates training dataset 5 and 6 for given candidate with outliers eliminated
79	createtr8n9.py	Python script	Creates training dataset 8 and 9 for given candidate

#	Name of source listing file	Type	Description
80	createtr8n9wOE.py	Python script	Creates training dataset 8 and 9 for given candidate with outliers eliminated
81	crossbeta.py	Python script	Generates and stores angle of departure between hot sequence vectors of a given candidate/imposter pair
82	crossthetaall.py	Python script	Generates and stores angle of departure between dwell time vectors of a given candidate/imposter pair
83	crossthetavaribuf.py	Python script	Generates and stores angle of departure between dwell time vectors of a given candidate/imposter pair for monitoring at constant time intervals instead of buffers of fixed size (not used)
84	dwell.py	Python script	Extract dwell time vector for a given candidate from Test 01
85	flight2.py	Python script	Extract flight time vectors for a given candidate from Test 01
86	flightfeature_oneseq.py	Python script	Produces a summary of flight time related features for a selected hot sequence for a given candidate on standard output
87	flightfeatures.py	Python script	Produces a summary of flight time related features for given candidate on standard output

#	Name of source listing file	Type	Description
88	getNonTemporalsExp1.py	Python script	Create and populate the common matrix NonTemporalsExp1.npy/csv for genuine test taking during Experiment 1.
89	livedwell.py	Python script	Captures and analyzes feature vectors associated with 'Dwell Time' of a candidate against the captured feature vectors in the profile.
90	liveflight2.py	Python script	Captures and analyzes feature vectors associated with 'Flight Time' of a candidate against the captured feature vectors in the profile.
91	liveflight2summary.py	Python script	Generates a summary matrix of simulated impersonation of all players by all players
92	MachineStats.py	Python script	Generates and saves statistical data about the performance of learning machines
93	mergeDynAccVectors.py	Python script	Merges all dynamic accuracy vectors created by 'SIMexp1dynacc.py' into a single matrix
94	mergeFAIvectors.py	Python script	Merges all FAI vectors captured during 'SIMexp3anal.py' and' SIMexp4anal.py' into two single matrices and stores them
95	performance.py	Python script	Analyzes and records the detector results generated by SIMexp2.py for a given candidate with each other candidate as imposter

#	Name of source listing file	Type	Description
96	performance2.py	Python script	Analyzes and records the detector results created by SIMexp2comb.py for a candidate while taking Test 02 genuinely.
97	performance3.py	Python script	Analyzes and records the detector results created by SIMexp2comb.py for a candidate while taking Test 03 (looking up).
98	performance3_.py	Python script	Same as performance3.py but considers only the detection of dwell time based and pause duration based classifiers
99	performance4.py	Python script	Analyzes and records the detector results created by SIMexp2comb.py for a candidate while taking Test 04 (collaborating).
100	performanceflight.py	Python script	Analyzes and records the detector results generated by SIMexp2flight.py for a given candidate with each other candidate as imposter
101	performanceSelf.py	Python script	Same as 'performance.py' but focuses on genuine test taking by given candidate
102	QQmerge2.py	Python script	Takes all 'crosstheta' files for a given candidate and merges them into a single matrix (saved as both .csv and .npy)
103	selectImposters.py	Python script	Selects the best 5 imposters for a given candidate based on 'dwell time' related features

#	Name of source listing file	Type	Description
104	selectImposters2.py	Python script	Selects the best 5 imposters for a given candidate based on 'pause duration' related features
105	selectImposters3.py	Python script	Selects the best 5 imposters for a given candidate based on 'flight time' related feature 'hot sequences'
106	selectLM4dwell.py	Python script	Identifies the dwell time based best classifier for a given candidate and loggs in the 'BestMachines' database
107	selectLM4flight.py	Python script	Identifies the pause duration based best classifier for a given candidate and loggs in the 'BestMachines' database
108	selectLM4hot.py	Python script	Identifies the hot sequence flight time based best classifier for a given candidate and loggs in the 'BestMachines' database
109	selftheta.py	Python script	Simulates the generation of 'self theta' matrix (angle of departure during Test 01)
110	selftheta_A.py	Python script	Same as 'sleftheta.py' but accepts command line arguments (for ease of automation)
111	SIMexp1dynacc.py	Python script	computes the dynamic accuracy of genuine detection during taking of Test 01
112	SIMexp1plotdynacc.py	Python script	plots current genuine detections of Experiment 1 for given candidate

#	Name of source listing file	Type	Description
113	SIMexp2.py	Python script	Simulates candidate taking Test 02 either genuinely or as an imposter and records detections by dwell time based classifier
114	SIMexp2anal.py	Python script	Analyses results generated by 'SIMexp2.py' and produces a report on standard output.
115	SIMexp2anal2.py	Python script	Same as 'SIMexp2anal.py' but stores the analysis report
116	SIMexp2comb2.py	Python script	Simulates candidate taking Test 02 either genuinely or as an imposter and records the combined detection by all 3 best classifiers.
117	SIMexp2dwell.py	Python script	Simulates candidate taking Test 02 either genuinely or as an imposter and records the detection by dwell time based best classifier.
118	SIMexp2flight.py	Python script	Simulates candidate taking Test 02 either genuinely or as an imposter and records the detection by pause duration based best classifier.
119	SIMexp2hot.py	Python script	Simulates candidate taking Test 02 either genuinely or as an imposter and records the detection by hot sequence flight time based best classifier.
120	SIMexp2pilot.py	Python script	Simulates candidate taking Test 02 genuinely and records the combined detection by all 3 best classifiers (to be used as a comparison benchmark)

#	Name of source listing file	Type	Description
121	SIMexp2ResultsSummary.py	Python script	Records the results of a simulated taking of Test 02 involving a candidate/imposter pair and records the results in a structured array
122	SIMexp2selected.py	Python script	Same as 'SIMexp2<*>.py' but with provision to select the type of classifier through a command line argument.
123	SIMexp2self.py	Python script	Simulates the genuine taking of Test 02 by given candidate and records the detection by each classifier
124	SIMexp2spcResultsSummary.py	Python script	Same as 'SIMexp2ResultsSummary.py' but simulates taking of Test 01 instead of Test 02
125	SIMexp3anal.py	Python script	Analyzes the results of 'SIMexp3comb.py' by a candidate (looking up) by comparing against pilot test (Test 02)
126	SIMexp3catchupdynacc.py	Python script	creates dynamic accuracy catchup matrix for Test 04 for a threshold factor given as command line argument
127	SIMexp3comb.py	Python script	Simulates candidate taking Test 03 (simulated lookup) and records the combined detection by all 3 best classifiers.
128	SIMexp3dynacc.py	Python script	compute the dynamic accuracy of genuine detection during taking of Test 03
129	SIMexp3plotdynacc.py	Python script	plots current genuine detections of Experiment 3 for given candidate

#	Name of source listing file	Type	Description
130	SIMexp4anal.py	Python script	Analyzes the results of 'SIMexp4comb.py' by a candidate (collaboration) by comparing against pilot test (Test 02)
131	SIMexp4catchupdynacc.py	Python script	creates dynamic accuracy catchup matrix for Test 03 for a threshold factor given as command line argument
132	SIMexp4comb.py	Python script	Simulates candidate taking Test 04 (simulated collaboration) and records the combined detection by all 3 best classifiers.
133	SIMexp4dynacc.py	Python script	compute the dynamic accuracy of genuine detection during taking of Test 04
134	SIMexp4plotdynacc.py	Python script	plots current genuine detections of Experiment 4 for given candidate
135	AnalAll.sh	UNIX shell script	automates the execution of SIMexp2.py and SIMexp2anal.py for all candidates
136	azdownload.sh	UNIX shell script	Download a single file (given as command line argument) from MS AZUR virtual machine
137	azupload.sh	UNIX shell script	Upload a single file (given as command line argument)to MS AZUR virtual machine
138	buildAllLM.sh	UNIX shell script	Automates the execution of 'buildLM*.py' for all candidates so that all classifiers will be created for all candidates

#	Name of source listing file	Type	Description
139	BuildNewMachines.sh	UNIX shell script	Automates the execution of buildLM5*.py, buildLM6*.py buildLM8*.py and builds LM9*.py for all candidates
140	chkQQ.sh	UNIX shell script	Lists the QQ (cross theta) files in the profiles of all candidates
141	chkSIMexp2files.sh	UNIX shell script	Checks for the creation of output files of Test 02 for all candidates
142	createcrosstheta_all.sh	UNIX shell script	Automates the execution of 'crosstheta_all.py' for all candidates
143	createprofile.sh	UNIX shell script	Extracts 'dwell time' and 'flight times' vectors for a given candidate and stores them in the 'profiles' folder [Used initially]
144	createvars.sh	UNIX shell script	Creates and exports environmental variables useful for several other scripts
145	QQmerge2.sh	UNIX shell script	Automates the execution of 'QQmerge2.py' for all candidates
146	selectLM4dwell.sh	UNIX shell script	Automates the execution of 'selectLM4dewll.py' for all candidates
147	selectLM4flight.sh	UNIX shell script	Automates the execution of 'selectLM4flight.py' for all candidates
148	SIMexp2All.sh	UNIX shell script	Automates the simulation of impersonation of all candidates by all other candidates through the nested calling of 'SIMexp2*.py'

#	Name of source listing file	Type	Description
149	SIMexp2anal.sh	UNIX shell script	Automates the consecutive execution of 'SIMExp2.py' and 'SIMexp2anal.py' for all candidates
150	SIMexp2comb.sh	UNIX shell script	Automates the execution of 'SIMexp2comb2.py' for all candidates

APPENDIX - C

RELATED PUBLICATIONS

The following publications were authored during the course of the research and dealt with various aspects connected with the research.

#	Title of publication	Co-authors	Status as at 2026.06.04	Remarks
1	Are 'Online Tests' Realistic in Sri Lanka?	Indika Perera	Presented at the annual sessions, Institution of Engineers, Sri Lanka, 2019 and published in the transactions	-
2	Non-obtrusive Imposter Detection During Online Assessments	Indika Perera Shantha Fernando	Published in the journal 'Engineer' of the Institution of Engineers, Sri Lanka, January 2025	-
3	The Challenge of Online Assessments in Sri Lanka	Indika Perera Shantha Fernando	Presented at the Research Symposium in the annual sessions of the Institution of Engineers, Sri Lanka, 2025 and published in the transactions	Received the 'Late Eng. E. O. E. Pereira award' for the best paper presented at the 2025 annual sessions.
4	Extension of non-obtrusive Imposter Detection during Online Tests to detect fraudulent acts other than Impersonation	Indika Perera Shantha Fernando	Structured abstract presented at the 6th CINEC International Research Symposium 2025 Full paper published in the 'Journal of Innovation in Science, Engineering and Technology (JISET)'	Poster presentation of the structured abstract was adjudged the best in the track 'Computing and Information Technology'
5	Lean Keystroke Logging for Continuous Authentication During Online Assessments	Thushan Sivalingam Nandana Rajatheva Indika Perera Shantha Fernando	Presented at the 2nd ADScAI research conference 2026, University of Moratuwa. Accepted for publication in the conference proceedings.	-
6	Non-obtrusive detection of clipboard transfers during online tests	Indika Perera Shantha Fernando	Submitted to ICTer 2026, acceptance decision being awaited.	-