National College of Ireland

Continuous User Authentication for Secure Messaging Using Advanced Machine Learning Models

MSc Research Project
Cyber Security

Kenechukwu Nwokedi
Student ID: x22248153

School of Computing
National College of Ireland

Supervisor: Liam McCabe

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: KENECHUKWU NWOKEDI

Student ID: x22248153

Programme: CYBERSECURITY **Year:** 2023/2024

Module: MSC RESEARCH PRACTICUM/INTERNSHIP

Supervisor: LIAM MCCABE

Submission Due Date: 12TH AUGUST 2024

Project Title: CONTINUOUS USER AUTHENTICATION FOR SECURE MESSAGING USING ADVANCED MACHINE LEARNING MODELS

Word Count: 9143 **Page Count** 33

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: N.K.C.

Date: 11TH AUGUST 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both	□

for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Table of Contents

TABLE OF FIGURES

TABLE OF ACRONYMS

ABBREVIATION	FULL PHRASE
ML	Machine Learning
DTC	Decision Tree Classifier
SVM	Support Vector Machine
RFC	Random Forest Classifier
GBC	Gradient Boosting Classifier
KMT	Keystroke Mouse Touch
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
D-CNN	Deep Convolutional Neural Network
LSTM-RNN	Long Short-Term Memory Recurrent Neural Network

Continuous User Authentication for Secure Messaging Using Advanced Machine Learning Models

Kenechukwu Nwokedi
x22248153

Abstract

Being unable to offer continuous verification once a person logs into a system is a major drawback of conventional authentication methods such as passwords and fingerprints (Dhaka, Rao, and Chaurasia, 2022) Without performing additional security checks, applications using these traditional methods are vulnerable to attacks from unauthorized users. This is a serious vulnerability, particularly for messaging services where users might keep open sessions for a long time. So, modern applications need to meet the needs of the ever-evolving cybersecurity landscape. In this study, a keystroke and mouse dataset are utilized for continuous authentication, extracting user behavioral patterns. Features such as average dwell time, flight time, and mouse trajectory are computed from the data and used to train different Machine Learning (ML) classification algorithms such as Decision Tree Classifier, Random Forest Classifier, Support Vector Machine, and Gradient Boosting Classifier. The algorithms use the patterns of behavior to classify users as "legitimate" or "illegitimate" and their performances are further compared using metrics like accuracy and F measure. Lastly, a custom messaging application is designed, and it was discovered that a selected trained model could be incorporated into the application framework for continuous, seamless authentication. The results suggest that machine learning models are effective at classifying users and behavioral biometric data can be used by applications for more advanced security.

Keywords: Continuous Authentication, Machine Learning, Cybersecurity, Behavioral Biometrics, Classifier, Secure Messaging.

1 Introduction

Secured messaging platforms are invaluable channels for maintaining the privacy, authenticity, and integrity of digital interactions in today's increasingly cyber-risky online world (R. Aswani, 2023). Reasons for which secured messaging is a critical requirement include protecting sensitive information, ensuring regulatory compliance, preventing data breaches, maintaining communication integrity, supporting business continuity, safeguarding intellectual property, enhancing trust, accommodating a remote workforce, and leveraging advanced security features. The FBI, in their annual internet crime report (FBI, 2024) highlights Business E-mail Compromise (BEC) as the second-costliest type of crime, after

investment scams, with 21, 489 complaints that amounted to $2.9 billion in losses in 2023 alone. The 2022 figures were 21,832 complaints with adjusted losses that amounted to over $2.7 billion (FBI, 2023). BEC describes a cybercrime in which cybercriminals compromise business messaging and carry out illegitimate transfers or diversion of funds. The emergence of cryptocurrencies makes the recovery of such funds a difficult task. As a result, there is a need for efforts to be focused on prevention, especially at the earliest possible point. Ensuring the security of messaging platforms is a critical step in this regard.

Since conventional authentication techniques, such as two-factor authentication (2FA) and passwords are widely used, easy to implement, and easy to manage (Colnago et al., 2018), they have long been the norm for protecting messaging platforms. However, as sophisticated cyber threats evolve and become harder to combat, these traditional techniques are becoming less effective. Being unable to offer continuous verification once a person logs into a system is a major drawback of conventional authentication methods. Without performing additional security measures, an unauthorized user has carte blanche to use the platform if they manage to get access to a session or device. This is a serious vulnerability, particularly for messaging services where users might keep open sessions for a long time.

To address these challenges, more sophisticated, robust, and continuous authentication methods are critical. A promising solution is continuous authentication, which takes advantage of behavioral biometrics like mouse movements and typing patterns (Abuhamad et al., 2020). By continuously observing user behavior, this method offers ongoing verification and enables the real-time detection of anomalies and the prevention of unauthorized access. Using machine learning techniques, this study intends to investigate the potency of continuous user identification via behavioral biometrics on a custom messaging network. The fundamental conceit of this theory is that it would be impossible to exploit credentials held by an unauthorized user, even in cases where the user's password or fingerprint has been illegally obtained. The aim of this research is to enhance the security of a custom messaging platform using machine learning techniques, by implementing a continuous user authentication system that leverages behavioural biometrics, such as typing patterns and mouse movements. Furthermore, the primary research question is formulated as:

Can non-conventional behavioral biometrics, such as typing patterns and mouse movements, effectively execute continuous authentication on a secured messaging platform?

The objectives of this project are outlined below:

1. To develop and train a machine learning model for continuous user authentication - The initial goal is to compare different machine learning models and train them using the proposed raw dataset. The selected trained model will analyse the collected behavioural data to differentiate between legitimate users and potential intruders.
2. To implement a continuous user authentication system within a custom messaging web application - The second objective seeks to integrate the trained model into a messaging application. This authentication system will continuously monitor user behaviour in the background and in real time to ensure that only authenticated users

can access and interact with the messaging application. The process seeks to be frictionless and unobtrusive, enhancing security and maintaining user friendliness in tandem.

3. To evaluate the usability of the system and its security via performance metrics: The final objective is to thoroughly examine the performance of the continuous authentication system. This includes evaluating its accuracy in identifying 'genuine' and 'fake' users, its impact on user experience, and its effectiveness in preventing unauthorized access.

2 Related Work

This section will consider some past implementations and applications of continuous authentication systems, focusing on authors who have explored the potential of behavioural biometrics and machine learning techniques in continuous authentication contexts.

Continuous Authentication in Secure Messaging

Dowling, Günther, and Poirrier in their 2022 paper, "Continuous Authentication in Secure Messaging", addressed the critical issue of post-compromise security in messaging communications (Dowling, Günther, and Poirrier 2022). The study begins by highlighting the primary challenge: once a user's device is fully compromised, an active adversary can easily impersonate the user by exploiting compromised session and long-term secrets. This is a significant vulnerability in widely used protocols like Signal. The study demonstrates that continuous authentication can lock out active attackers, provided that long-term secrets remain uncompromised. However, the proposed solution of leveraging long-term secrets stored in Trusted Platform Modules (TPM) or smart cards for continuous authentication raises practical concerns regarding the widespread adoption and user accessibility of such hardware-based security measures. TPMs and smart cards, while offering enhanced security, are not universally available or used, particularly among average users. This dependency on specialized hardware could hinder the practical deployment of the proposed solution on a broad scale.

Secure System of Continuous User Authentication Using Mouse Dynamics

In their paper, "Secure System of Continuous User Authentication Using Mouse Dynamics", Quraishi and Bedi (Quraishi and Bedi, 2022) sought to improve upon traditional user authentication techniques by examining mouse usage behavior and utilizing it to provide continuous identity verification while a user interacts with a specific but unidentified tool developed for the study. The study also bases its work on previous, similar experiments that have been conducted using various classifiers such as Decision Tree Classifier and SVM and focuses heavily on their resulting FAR and FRR metrics. The mouse data collecting tool was given to 23 volunteer participants and their data was collected over 4 weeks, although only 8 participants' data was used for the study. Since the system proposed in the study only seeks to identify "genuine" users and thus has only a single class, the one-class SVM was used to

identify a user as legitimate. It is also worth mentioning that the system was designed for an "open scenario" where there are no pre-established users at the start of the session and the system could be accessed by anyone. At the meeting point of FRR and FAR, the system achieves an EER (also known as Equal Error Rate) of 50% and the training accuracy for all 8 users ranges from about 89% - 91%. Although the accuracy was very high, indicating that users can be identified correctly, and the authentication system is non-intrusive and does not require additional user action besides regular mouse use, the average EER indicates that a significant number of malicious actors will be tagged as genuine while genuine users will be tagged as fake and refused access. Additionally, considering the limited data from only 8 users that the study employed, this could lead to a less than satisfactory performance in a real-world environment. Furthermore, the study fails to provide the test accuracy of the models in the results. Since this study only considered mouse dynamics with middling results, this suggests that using hybrid/multi-faceted traits in tandem like mouse dynamics with keystrokes metrics could improve the efficiency of the system.

Deep Learning Approaches for Continuous Authentication Based on Activity Patterns Using Mobile Sensing.

In their paper, "Deep Learning Approaches for Continuous Authentication Based on Activity Patterns Using Mobile Sensing" Mekruksavanich and Jitpattanakul introduced DeepAuthen (Mekruksavanich and Jitpattanakul 2021), a continuous authentication framework utilizing smartphone sensors (accelerometer, gyroscope, and magnetometer) to authenticate users based on their physical activity patterns measured by the sensors in their smartphone. The framework employs deep learning classifiers, including the proposed DeepConvLSTM network, to analyse data from three benchmark datasets. The study utilized the UCI-HAR dataset, which includes sensor data from 30 volunteers engaged in six daily activities: walking, walking upstairs, walking downstairs, sitting, standing, and lying down. The data, collected at a fixed rate of 50 Hz, consist of triaxial linear acceleration and angular velocity. Similarly, the WISDM-HARB dataset, involving 51 participants performing 18 different tasks, provided accelerometer and gyroscope data at 20 Hz. The HMOG dataset, collected from 100 participants across 24 periods, includes accelerometer, gyroscope, and magnetometer data along with various screen interaction metrics. Their result shows that with complex activity sequences like ascending stairs, descending stairs, and typing, the DeepConvLSTM model outperformed comparative studies by margins ranging from 2.6% to 5.47%.

Machine and Deep Learning Applications to Mouse Dynamics for Continuous User Authentication

In the study, "Machine and Deep Learning Applications to Mouse Dynamics for Continuous User Authentication" (Siddiqui et al. 2022), the researchers discussed the limitations of static authentication methods like passwords and highlighted the potential of continuous authentication, which continuously verifies users even after initial access. They focused on mouse dynamics as a biometric for continuous authentication and evaluated their dataset of

40 users using various machine learning and deep learning algorithms. The study found that deep learning models, particularly the 1D-CNN and LSTM-RNN, outperformed machine learning models in accuracy, with the ANN achieving a peak accuracy of 92.48%. The study also discussed the limitations of binary classification in mouse dynamics and proposed multi-class classification as a more detailed approach. The research highlights the need for further exploration in mouse dynamics, especially in high-intensity tasks like professional gaming, and suggests avenues for future research, including deeper models, alternative preprocessing methods, and transfer learning.

Comparing Machine Learning Classifiers for Continuous Authentication on Mobile Devices by Keystroke Dynamics.

The 2021 research paper, "Comparing Machine Learning Classifiers for Continuous Authentication on Mobile Devices by Keystroke Dynamics" (De-Marcos et al. 2021) contributed to the field by training and testing seven ML CA models using keystroke dynamics data from the soft keyboard of mobile phones. The dataset was derived from the HMOG dataset, which records interactions of 100 users during various sessions. Keystroke mechanics metrics considered included pressingTime, timeReleaseNextPress, and timeBetweenPress. The dataset was structured to include approximately 50% authorized and 50% unauthorized events for each user, enabling the evaluation of model performance. Ensemble algorithms like RFC, ETC, and GBC performed best, with average accuracy of over 70% for most metrics. GBC statistically outperformed other methods, followed by k-NN, while SVM performed the worst. The study also analyzed the importance of features, with pressingTime found to be the most significant, followed by timeReleaseNextPress, timeBetweenPress, keyCode, and nextKeyCode. The results suggest that combining individual predictions can improve reliability, mitigating false positives and negatives.

The existing literature on continuous user authentication using behavioural biometrics for messaging platforms explores the potential of leveraging typing patterns and mouse movements to enhance security. However, this body of work has several limitations. Many studies remain theoretical and lack real-world implementation, hindering the validation of their effectiveness in practical settings. Moreover, most of the studies reviewed in the existing literature are not focused on frictionless continuous authentication. While they explore the potential of leveraging typing patterns and mouse movements for enhanced security, they often involve additional steps or hardware requirements that may introduce friction into the user experience. For example, the solution proposed by Dowling, Günther, and Poirrier (2022) relies on specialized hardware like Trusted Platform Modules (TPM) or smart cards, which are not universally accessible or user-friendly. This lack of focus on frictionless authentication methods limits the practicality and widespread adoption of these techniques in real-world messaging platforms. This study stands out by offering a frictionless continuous authentication approach, ensuring that users do not need to engage in any additional activities outside of their normal interactions within the messaging platform. This approach not only enhances security but also prioritizes user convenience, a crucial factor for the practical deployment and widespread adoption of such authentication methods.

3 Research Methodology

This section describes the process of building the continuous authentication system, the custom messaging web application, and incorporating a trained model generated in the former into the latter. The dataset used in this research originated from (Nnamoko et al. 2022) – a collection of 1,760 KMT (keyboard, Mouse, and Touch) dynamics instances collected for use by academics at Edge Hill University's Computer Science Department, United Kingdom. This dataset contains KMT instances of legitimate and fraudulent owners of a fictitious card. To implement the machine learning classification part of the work, features such as the average dwell time, average mouse travel trajectory, and average flight time were extracted from the key and mouse events in the raw dataset. After this, various machine learning models such as Decision Tree Classifier, Random Forest Classifier, Support Vector Machine and Gradient Boosting Classifier were used to train the data set and their performances are examined using accuracy and F-score metrics. To build the messaging application, HTML, CSS, JavaScript, and Python programming languages were used.

3.1 Data pre-processing

In the dataset used in this research, 1,760 KMT dynamic instances were gathered throughout 88 user sessions (and 88 separate json. files) on the GUI application created by the researchers (Nnamoko et al. 2022). Each of the files has 20 instances of KMT data relating to a certain fabricated bank card, with the data cleanly divided into legitimate (10 “true data” or class “1” instances) and fraudulent (10 “false data” or class “0” instances). The dataset has a set of mouse and keyboard events which include for keys: key pressed (e.g. ‘b’ for key B), the event type (press or release), the input box in focus, whether text was changed, and timestamps in YYYY-MM-DD HH:MM:SS.SSSSSS scheme. For mouse events, these include the nature of the event (‘scroll up’, ‘scroll down’, ‘mouse press’, ‘left press’, ‘right press’ etc.) coordinates of the mouse at the exact time of the event, timestamps, epoch, and movement ID value.

3.2. Feature Extraction

The process of extracting distinct, pertinent information out of raw data is known as Feature Extraction (FE) (Salau and Jain 2019). During the feature extraction process, the following features were calculated: Average dwell time, average flight time, and average mouse trajectory. Below is a brief description of each feature:

Dwell/Hold Time: The duration that passes between pressing and releasing a key (T. Xi, et al. 2023)

Flight Time: In contrast to the dwell time, this is defined as the “latency between two keys” or press-to-press latency.

Average Dwell Time (dwell_avg): A keyboard event, this is the average length of time that a keyboard key is pressed. After finding the "press" and "release" key events, as well as the difference in their "Epoch" values, the dwell time for each key press was determined (Nnamoko et al. 2022).

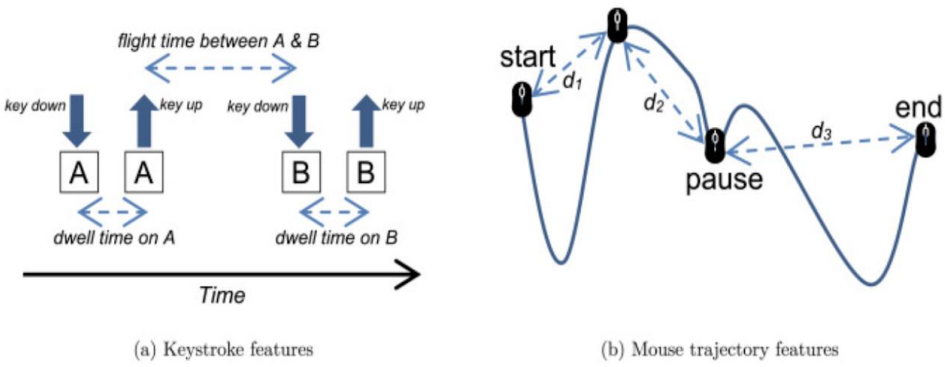
Computed as: dwell_avg = sum(dwell_list)/len(dwell_list)

Average Flight Time (flight_avg): Also, a keyboard event, can be defined as the average time between the release of a key and the press of the subsequent key. This was determined by averaging all variances in 'Epoch' numbers between a key release event and the subsequent key press event (Nnamoko et al. 2022).

Computed as: flight_avg = sum(flight_list)/len(dwell_list)

Average Mouse Trajectory(traj_avg): A mouse event, this is the mean distance covered by the mouse during a mouse movement. For every "movement" occurrence with the same "Movement ID," the total distances traveled from one coordinate to another were added up, producing a trajectory distance for each "Movement ID." Lastly, how far apart these "Movement IDs" were on average was calculated (Nnamoko et al. 2022).

Computed as: traj_avg = sum(traj_list)/len(traj_list)



(a) Keystroke features (b) Mouse trajectory features

Figure 1. Keystroke and Mouse Trajectory features extracted from the KMT data Source: Nnamoko et al. (2022)

3.2 Materials Used

Data processing, feature extraction and classification, and model training was implemented using Google Colaboratory (or Colab), a cloud-based platform built on Jupyter Notebooks which is easy to use and requires no charges for access to a powerful GPU (Carneiro et al., 2018). Programming languages adopted for the research include Python language for the backend component and HTML, CSS, and JavaScript for the frontend components. Python version 3.12.4 was installed, and the Flask Python framework was used because it offers practical tools that make it easy and quick to build a web application (Dyouri and Hancox, 2020) and it's also great for novice programmers. HTML was used to facilitate user interaction on the “Register” and “Chat” pages, CSS was used to style the pages and enhance user-friendliness, and JavaScript was employed on the application to capture keystroke and scroll events. Additionally, I used Visual Studio Code (version 1.88.1), an open-source text editor to write, edit, and run the various scripts as well as install and update all the necessary libraries.

4 Design Specification

This section will discuss the framework of the different machine algorithms used for training the dataset which include, Decision Tree Classifier, Random Forest Classifier, Support Vector Machine, and Gradient Boosting Classifier. These models were selected after a thorough examination of their performances in related works and their capabilities for the specific goals of user identification. Additionally, a detailed description of the proposed system and the approach used is provided in this section.

4.1 Decision Trees

Sorting instances according to feature values is how Decision Trees (DT) classify instances. A decision tree's nodes each represent a feature in an instance that needs to be classified, and its branches each stand for values that the node can take (Singh, Thakur, and Sharma, 2016). This classifier uses a decision tree as a predictive model to project observations about an object to inferences about the object’s target value. Decision Trees can manage feature interactions with ease, they’re simple to grasp and interpret (Singh et al. 2016). Outliers have little to no effect on the model because it is non-parametric, and it can handle data that is linearly inseparable. Additionally, decision trees can maintain accuracy and precision in the face of noise, deliver great performance for very little computing work, and can handle a wide range of data (e.g. nominal, numeric, textual), missing values, and redundant features. They also have considerable generalization capabilities. On the flip side, handling high dimensional data is challenging with this classifier. Even though there is reduced computing time, building the tree takes a long time. Decision Tree’s divide and conquer technique works well when there are a small number of highly relevant traits, but it suffers when there are several intricate interconnections. With more classes, errors spread through trees, which poses a significant issue.

4.2 Random Forest

The Random Forest (RF) is a supervised classification algorithm that sorts information by building several classifiers to sharpen prediction accuracy (Shaik and Srinivasan 2019). The test data set is subjected to the Random Forest algorithms, which build trees and computes together the outcome of the individuals to predict the class label. When using this algorithm, an ensemble approach, several decision trees are trained, and the class with the majority across all the trees in the ensemble is returned (Singh et al. 2016). Additionally, it could result in poorer accuracy to categorize a lot of data with a single classifier. Because of this, the Decision Tree Classification Algorithm is frequently employed in applications where a lot of data needs to be classified. In many classification situations, Random Forest is the most successful, typically just ahead of SVMs. They don't overfit, are scalable, quick, and easy to understand, all while requiring no parameter management. Furthermore, they also perform excellently against noise. For real-time prediction, the process grows slowly as the number of trees increases. Many attempts have been made to improve Random Forests, such as utilizing several attribute evaluation metrics in split selection and reducing the correlation between the trees.

4.3 Support Vector Machine

Although the Support Vector Machine (SVM) is complicated, it has a high accuracy rate, and it can also function well if your data isn't linearly separable in base feature space; provided you use the right kernel (Singh et al. 2016). Additionally, it avoids theoretical assurances of overfitting. The idea behind the SVM model is to maximize the shortest path between the closest sample point and the hyperplane. In contrast to K Nearest Neighbor, performance and accuracy metrics depend only on the amount of training cycles rather than the size of the data. SVM is especially common in text categorization cases where it is usual to work with extremely high dimensional areas. Furthermore, there is no relationship between complexity and feature count, it can generalize well and is resilient to large dimensional data. However, training speed is lower, and parameter selection affects how well it performs. Features selection and image classification are two classic applications of SVM. The evolutionary SVM is another model of this type that is used to address the dual optimization issue of SVM, and it can be used to create both an adaptable feature extractor and an efficient classifier.

4.4 Gradient Boosting Classifier

Utilizing a gradient boosting architecture, Gradient Boosting (or XGBoost) classifier is an ensemble machine learning technique based on decision trees (Salunke and Ouda. 2023). Gradient boosting is a supervised learning approach that combines the estimates of several weaker, simpler models to predict a target variable with a high degree of accuracy. For regression and classification issues, the gradient tree boosting technique is utilized (Bahad and Saxena 2020). Gradient boosting stirs in the opposite direction of the gradient seeking to minimize loss function. The effectiveness of a model at making predictions for a given set of parameters is indicated by its loss function. Additionally, the nature of problem determines which loss function should be applied. For instance, in this study, the loss function for binary

user classification is a metric based on the inability to identify a fraudulent user. Gradient Boosting is faster because it makes use of parallel processing and when used with a well-structured dataset, it performs quite well. It uses regularization to prevent overfitting and can handle missing values.

4.5 PROPOSED SYSTEM

ENROLMENT STAGE: The enrolment phase is a critical step in building the system which involves capturing and storing the keystroke and mouse patterns of users for future matching. This phase occurs after the model has been trained and integrated into the custom messaging platform. It begins with the user registration at the start of the session where the user provides specific details such as full name and address while typing and scrolling and the system collects behavioural data such as key press and release events and scroll events. Metrics such as dwell time, flight time, and average scroll delta are then calculated and stored securely in a database associated with the user's profile. This profile is stored securely and used as a reference for future authentication.

AUTHENTICATION STAGE: After successful authorization into the messaging application, the system runs in the background to capture key and scroll events at the exact times at which the events occur. In this phase, the calculated metrics from the key and scroll events are then continuously compared with the stored profile of the 'legitimate' user which has already been created in the enrolment stage. After this comparison process which employs the use of a machine learning model for classification, the system either logs the user out of the system or keeps them signed in.

5 Implementation

5.1 Introduction

This section details steps taken regarding the final stage of the implementation of the continuous user authentication system into the custom messaging platform. This section also contains details of the enrollment/registration phase, which is crucial for capturing and storing a given user's behavioral data for comparison and authentication.

5.2 Machine Learning Libraries

After uploading the raw dataset to Google Colab, the next stage was to import the necessary libraries necessary for the feature extraction and machine learning model training. The machine learning classes and functions such as (sklearn import tree) and (sklearn.model_selection import train_test_split) were imported from the scikit-learn library (Scikit-learn 2019). Meanwhile, other libraries such as pandas, math, and json were imported for data analysis, providing necessary math functions for calculations, and handling json data format respectively. In order to perform the core evaluation of the performance of the

machine learning model on the training set using metrics like accuracy and F1-score, libraries such as (sklearn.metrics import classification_report) and (sklearn import metrics) were also imported from the scikit-learn metrics submodule. The data is divided into training and test sets and the various selected models are trained with the training data as signified with X_train (for features) and y_train (for labels). The output class is then predicted using the trained models on the testing features (X_test), and (y_test) represents the "true" class output for the testing data.

5.3 Evaluation Metrics

To examine the effectiveness of our classification models, the performance metrics below are computed:

Accuracy: The number of accurate predictions (in percentage) made using the test data is known as accuracy. Mathematically, it is derived by summing up the true positives and true negatives and dividing by the sum of the total number of predictions (Naidu, Zuva, and Sibanda 2023).

Accuracy = TP+TN/(TP+TN+FP+FN), where TP = True Positive, TN = True Negative, FP = False Positive, and FN = False Negative

F1-score: The F1 Score is a valuable metric for calculating the efficiency of unbalanced datasets as it is determined by employing the Precision and Recall functions (Naidu et al. 2023) as thus:

F1-score = (P*R)/(P+R), where P = Precision and R = Recall

Precision: When a model predicts true positives, its precision is calculated by the ratio of accurate forecasts (Naidu et al. 2023). True positives divided by the addition of true positives and false positives is the formula for this.

Precision = TP/(TP+FP)

5.4 Implementation of the Web-Based Messaging Platform

For this project, continuous authentication is integrated into a web-based messaging platform. The primary goal is to utilize keystroke dynamics and mouse scroll patterns to validate the user's identity during their session.

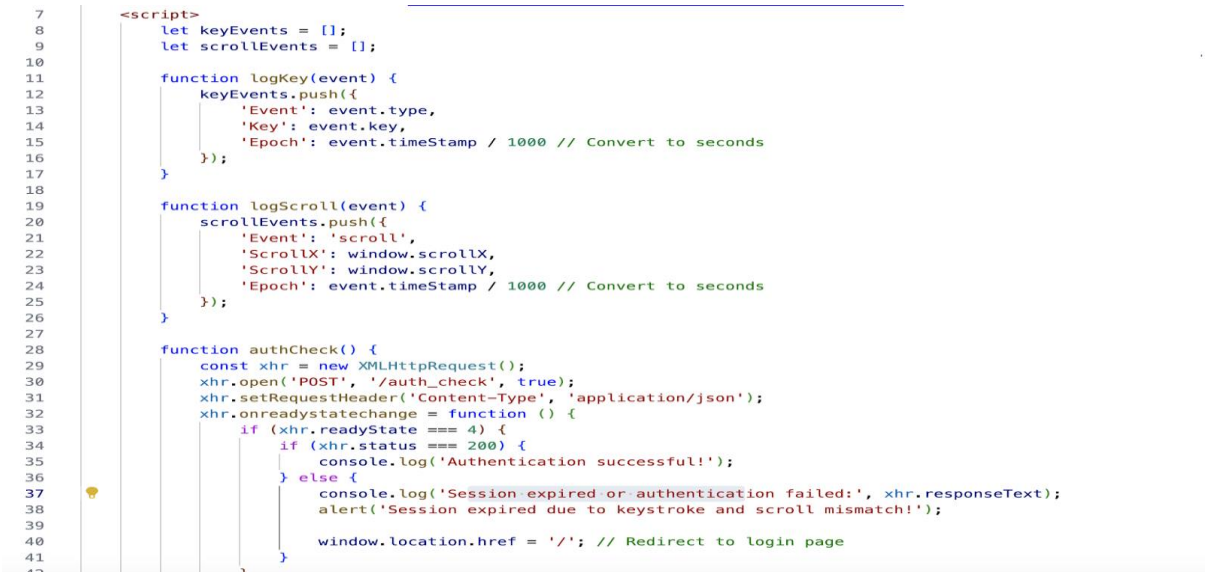
5.4.1 Front End Components

Two primary HTML files are configured:

a. User Registration Page: This page allows the users to create accounts by providing personal information (such as name, username, password, full name, and address) and capturing initial keystroke and mouse events. This page has integrated JavaScript scripts

with the functionality to collect keystrokes and mouse_scroll as the user is inputting their details to use it for future authentication.

b. Messaging Page: This is the main interface for sending messages and it continuously captures keystroke and scroll events. The messaging HTML file has the functionality to extract user keystrokes and mouse_scroll data for continuous authentication.



```
<script>
    let keyEvents = [];
    let scrollEvents = [];

    function logKey(event) {
        keyEvents.push({
            'Event': event.type,
            'Key': event.key,
            'Epoch': event.timeStamp / 1000 // Convert to seconds
        });
    }

    function logScroll(event) {
        scrollEvents.push({
            'Event': 'scroll',
            'ScrollX': window.scrollX,
            'ScrollY': window.scrollY,
            'Epoch': event.timeStamp / 1000 // Convert to seconds
        });
    }

    function authCheck() {
        const xhr = new XMLHttpRequest();
        xhr.open('POST', '/auth_check', true);
        xhr.setRequestHeader('Content-Type', 'application/json');
        xhr.onreadystatechange = function () {
            if (xhr.readyState === 4) {
                if (xhr.status === 200) {
                    console.log('Authentication successful!');
                } else {
                    console.log('Session expired or authentication failed:', xhr.responseText);
                    alert('Session expired due to keystroke and scroll mismatch!');

                    window.location.href = '/'; // Redirect to login page
                }
```

Figure 2. JavaScript code for Keystroke and Mouse_scroll Collection

The JavaScript code seen in the screenshot above is for event logging and the user interactions captured include keystroke events (logs ‘keydown’ and ‘keyup’ events, actual key, and timestamp) and scroll events (scroll positions and timestamps).

5.4.2 Backend Components

5.4.2.1 Flask Application Setup

The Flask application handles the main logic, and the code implements a Flask-based authentication system that utilizes keystroke dynamics and scroll patterns to verify user identity. Following the importation of the Flask object from the flask package, the name app is used to create an instance of your Flask application. Once the app instance is created, it is now used to respond to user inquiries and control incoming web requests. By using the Flask routing mechanism noted as @app.route, a standard Python function is converted into a Flask view function and said function's return value is then transformed into an HTTP response (Dyouri and Hancox, 2020). This response is then shown by an HTTP client, for example, a browser. To instruct the routing mechanism, @app.route(), that it should reply to web queries for the primary URL/, it is supplied the value '/'. The key components are detailed below:

- Registration Endpoint: Collects user data and initial metrics during registration.
- Authentication Check Endpoint: Periodically validates the user's session.

- Messaging Endpoint: Handles message sending.

```python
from flask import Flask, request, jsonify, session, redirect, url_for, render_template
import joblib
import numpy as np
import os
import json

app = Flask(__name__)
app.secret_key = "nmtpllseet34gssce4t"

# Define the file path for the model
model_path = os.path.join(os.path.dirname(__file__), 'server/svm4_model (1).pkl')

# Load the trained model
if os.path.exists(model_path):
    model = joblib.load(model_path)
else:
    raise FileNotFoundError(f"Model file not found: {model_path}")

users_db = {}


def calculate_metrics(key_events, scroll_events):
    dwell_times = []
    flight_times = []
    scroll_deltas = []

    for i, event in enumerate(key_events):
        if event['Event'] == 'pressed':
            key = event['Key']
            press_time = float(event['Epoch'])
            release_time = next((float(e['Epoch']) for e in key_events[i+1:] if e['Key'] == key and e['Event'] ==
            dwell_time = release_time - press_time
            dwell_times.append(dwell_time)

            if i > 0:
                previous_event = key_events[i-1]
                if previous_event['Event'] == 'pressed':
                    flight_time = press_time - float(previous_event['Epoch'])
                    flight_times.append(flight_time)
```

Figure 3. Flask Application Setup which includes importing the flask package and other relevant libraries and initialization with a secret key for session management

5.4.2.2 User Metrics Calculations and Saving of Biometric Data

Furthermore, the pre-trained model is loaded from a specified path using `joblib`. If the model file is not found, an error is raised. A simple, temporary, in-memory dictionary is used to store user data, including their password and keystroke metrics. At this stage, key and scroll events are computed to calculate user-specific metrics such as dwell times, flight times, and scroll deltas. These metrics are extremely essential for characterizing user behavior (Namnaik, Kurale, and Mahindrakar 2018). User data, specifically key events, are saved to a file in a directory named after the user. This ensures that user data can be retrieved and analyzed later.

```python
def calculate_metrics(key_events, scroll_events):
    dwell_times = []
    flight_times = []
    scroll_deltas = []

    for i, event in enumerate(key_events):
        if event['Event'] == 'pressed':
            key = event['Key']
            press_time = float(event['Epoch'])
            release_time = next((float(e['Epoch']) for e in key_events[i+1:] if e['Key'] == key and e['Event'] ==
            dwell_time = release_time - press_time
            dwell_times.append(dwell_time)

            if i > 0:
                previous_event = key_events[i-1]
                if previous_event['Event'] == 'pressed':
                    flight_time = press_time - float(previous_event['Epoch'])
                    flight_times.append(flight_time)

    for i in range(1, len(scroll_events)):
        delta_x = scroll_events[i]['ScrollX'] - scroll_events[i-1]['ScrollX']
        delta_y = scroll_events[i]['ScrollY'] - scroll_events[i-1]['ScrollY']
        scroll_deltas.append((delta_x, delta_y))

    average_trajectory = sum(dwell_times) / len(dwell_times) if dwell_times else 0
    average_scroll_delta = np.mean(scroll_deltas, axis=0) if scroll_deltas else (0, 0)

    return {
        'dwell_times': dwell_times,
        'flight_times': flight_times,
        'average_trajectory': average_trajectory,
        'average_scroll_delta': average_scroll_delta
    }
```

Figure 4. Metric Calculation for Dwell time, Flight time, and Scroll Deltas

5.4.2.3 Endpoints

For the user registration endpoint, user registration data is processed, keystroke metrics are calculated for, the user data is saved, and the metrics are stored in a temporary, in-memory database. It also creates a session for the user upon successful registration. The authentication endpoint periodically checks user authentication based on the keystroke and scroll events, comparing the current user metrics with the registered metrics using the pre-trained model. Hence, it uses this comparison approach to either maintain or terminate the session based on the prediction with a continuous authentication interval of 60 seconds. The messaging endpoint renders the chat interface if the user is authenticated while the message sending endpoint processes and acknowledges user messages. The root endpoint renders the user registration page, and the Flask application is run in debug mode. It should be noted that the application is operating locally at http://127.0.0.1:5000/, where :5000 represents the port number and 127.0.0.1 corresponds to the machine's localhost.

```python
@app.route('/register', methods=['POST'])
def register():
    data = request.get_json()
    username = data.get('username')
    password = data.get('password')
    key_events = data.get('key_events', [])
    scroll_events = data.get('scroll_events', [])

    # Check if username already exists (prevent duplicate registrations)
    if username in users_db:
        return jsonify({'message': 'Username already exists!'}), 200

    # Calculate keystroke metrics
    metrics = calculate_metrics(key_events, scroll_events)
    feature_vector = metrics['dwell_times'] + metrics['flight_times'] + [metrics['average_trajectory']] + list(me

    users_db[username] = {
        'password': password,
        'keystroke_metrics': feature_vector
    }

    # Save user data
    save_user_data(username, key_events)

    session['username'] = username
    return jsonify({'message': 'Registration successful!'}), 201  # Created status code

```

Figure 5. Displaying the User Registration Endpoint

```python
@app.route('/auth_check', methods=['POST'])
def auth_check():
    if 'username' not in session:
        return jsonify({'message': 'Not authenticated!'}), 401

    username = session['username']

    data = request.get_json()
    key_events = data.get('key_events', [])
    scroll_events = data.get('scroll_events', [])
    metrics = calculate_metrics(key_events, scroll_events)
    feature_vector = metrics['dwell_times'] + metrics['flight_times'] + [metrics['average_trajectory']] + list(me

    # Save user data
    save_user_data(username, key_events)

    if not key_events:  # No keystrokes, user is idle
        return jsonify({'message': 'User is idle, no authentication required.'}), 200

    registered_metrics = users_db[username]['keystroke_metrics']
    prediction = model.predict([feature_vector])

    if prediction[0] != 1:
        session.pop('username', None)
        return jsonify({'message': 'Session expired due to keystroke and scroll mismatch!'}), 401

    return jsonify({'message': 'Authentication successful!'}), 200

```

Figure 6. Displaying the Authentication Endpoint

```python
@app.route('/messaging', methods=['GET'])
def messaging():
    if 'username' not in session:
        return redirect(url_for('/login_page'))
    return render_template('chat.html', username=session['username'])


@app.route('/send_message', methods=['POST'])
def send_message():
    data = request.get_json()
    message = data.get('message')
    # Save or process the message as needed
    return jsonify({'message': 'Message sent successfully!'}), 200


```

Figure 7. Displaying the Messaging Endpoint

```
140
141  @app.route('/', methods=['GET'])
142  def login_page():
143      return render_template('register.html')
144
145
146  if __name__ == '__main__':
147      app.run(debug=True)
```

Figure 8. Displaying the Login Endpoint

6 Evaluation

After data pre-processing and extracting the average dwell time, average mouse travel trajectory, and average flight time features, the dataset was split into training and test sets and various machine learning algorithms such as Decision Tree Classifier (DTC), Random Forest Classifier (RFC), Support Vector Machine (SVM), and Gradient Boosting Classifier (GBC) were used to train the models with the aid of different python libraries like pandas, matplotlib, sklearn, and pandas. The model's performance was then examined using accuracy and F-score metrics. After examining the performances, the SVM model was selected and extracted in the form of a Python Pickle file (PKL) using the pickle module. Following this, the front-end components and back-end components of the proposed messaging web application are built, and the model is incorporated into the application by utilizing the Flask python web framework in the codebase. To evaluate the functionality of the application, we then tried to make it distinguish between the two separate typing hands of a person, with the right hand typing and scroll pattern serving as User A and the left hand typing and scroll pattern serving as User B.

6.1 Evaluation Metrics

The following Table 6.1 shows the various learning algorithms and the results derived from their training model.

		Metrics			
MODELS	Hyperparameters	Minimum and Maximum Accuracy (%)	Minimum and Maximum Weighed F-Score (%)	Mean Accuracy (%)	Mean F-Measure (%)
Decision Tree Classifier (DTC)	Default setting Random state = 0	Min = 25 Max = 100	Min = 20 Max = 100	76.98	75.54
Random Forest Classifier	n_estimators = 150 Random state = 1	Min = Max = 50	Min = Max = 33	76.98	75.54

(RFC)					
Support Vector Machine (SVM)	kernel='linear', C= 1, gamma = 1	Min = 25 Max = 100	Min = 20 Max = 100	76.98	75.54
Gradient Boosting Classifier (GBC)	learning_rate=0.1, n_estimators=100	Min = 25 Max = 100	Min = 20 Max = 100	76.98	75.54

Table 6.1 A table comparing the selected machine learning models and their evaluation metrics

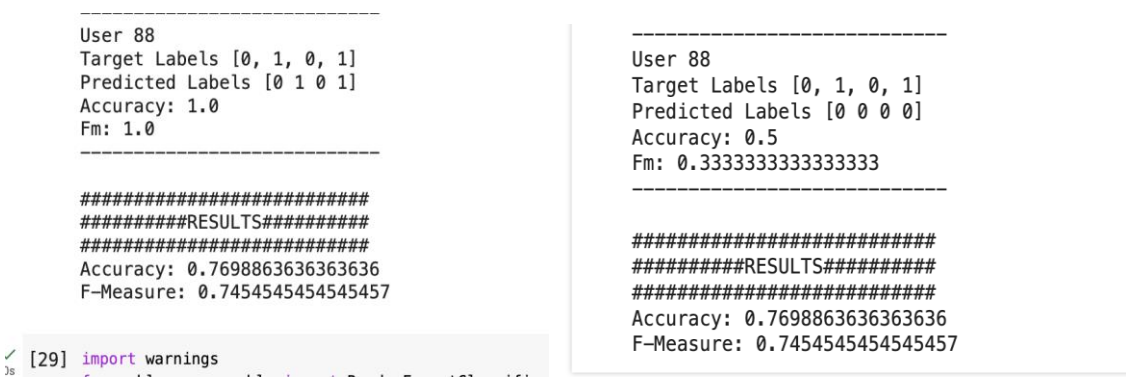


Figure 9. Showing the Performance Metrics for The Decision Tree and Random Forest Models Respectively

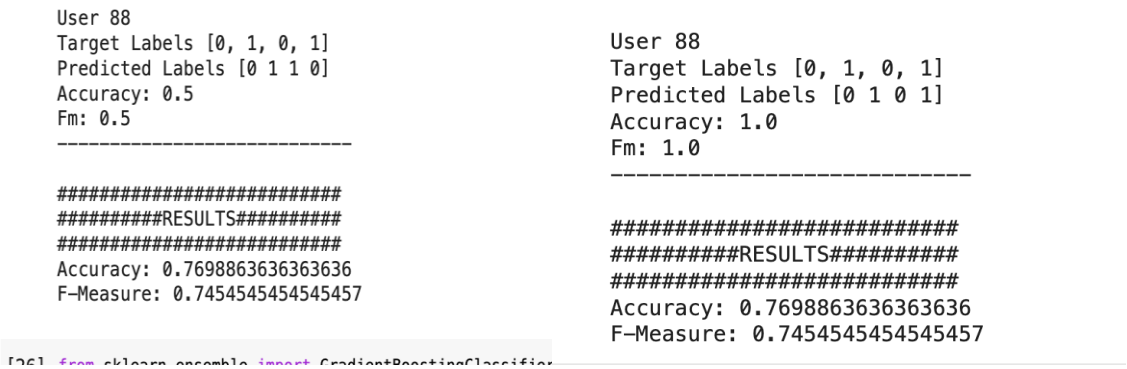


Figure 10. Showing the Performance Metrics for The Support Vector Machine and Gradient Boost Models Respectively

```
        clf = tree.DecisionTreeClassifier()
        clf.fit(X_train, y_train)
        predicted_labels = clf.predict(X_test)
        clf_rep = classification_report(y_test, predicted_labels, output_dict=True)
        acc = clf_rep['accuracy']
        fm = clf_rep['weighted avg']['f1-score']
        print('User', i)
        print('Target Labels', y_test)
        print('Predicted Labels', predicted_labels)
        print('Accuracy:', acc)
        print('Fm:', fm)
        print('----------------------------')

        acc_list.append(acc)
        fm_list.append(fm)



    final_acc = sum(acc_list)/len(acc_list)
    final_fm = sum(fm_list)/len(fm_list)


    with open('dtc_model.pkl', 'wb') as file:
        pickle.dump(clf, file)


    print(' ')
    print('############################')
    print('##########RESULTS##########')
    print('############################')
    print('Accuracy:', final_acc)
    print('F-Measure:', final_fm)
```

Figure 11. Decision Tree Classifier

```
      rfc = RandomForestClassifier(max_depth = None, min_samples_leaf=8, min_samples_split=3, ccp_alpha=0.0, n_estimators = 150, random_state = 1)
      rfc.fit(X_train, y_train)
      predicted_labels = rfc.predict(X_test)
      clf_rep = classification_report(y_test, predicted_labels, output_dict=True)
      acc = clf_rep['accuracy']
      fm = clf_rep['weighted avg']['f1-score']
      print('User', i)
      print('Target Labels', y_test)
      print('Predicted Labels', predicted_labels)
      print('Accuracy:', acc)
      print('Fm:', fm)
      print('----------------------------')

      acc_list2.append(acc)
      fm_list2.append(fm)

  with open('rfc_model.pkl', 'wb') as file:
      pickle.dump(rfc, file)


  final_acc = sum(acc_list)/len(acc_list)
  final_fm = sum(fm_list)/len(fm_list)


  print(' ')
  print('############################')
  print('##########RESULTS##########')
  print('############################')
  print('Accuracy:', final_acc)
  print('F-Measure:', final_fm)
```

Figure 12. Random Forest Classifier

```
        svm = SVC(kernel='linear', C= 1, gamma = 1)
        svm.fit(X_train, y_train)
        predicted_labels = svm.predict(X_test)
        clf_rep = classification_report(y_test, predicted_labels, output_dict=True)
        acc = clf_rep['accuracy']
        fm = clf_rep['weighted avg']['f1-score']
        print('User', i)
        print('Target Labels', y_test)
        print('Predicted Labels', predicted_labels)
        print('Accuracy:', acc)
        print('Fm:', fm)
        print('------------------------------')

        acc_list3.append(acc)
        fm_list3.append(fm)


    with open('svm4_model.pkl', 'wb') as file:
        pickle.dump(svm, file)

    final_acc = sum(acc_list)/len(acc_list)
    final_fm = sum(fm_list)/len(fm_list)


    print(' ')
    print('############################')
    print('##########RESULTS##########')
    print('############################')
    print('Accuracy:', final_acc)
    print('F-Measure:', final_fm)
```

Figure 13. Support Vector Model Classifier

```
[ ]
    #---------------------------------
    y = final_df['label'].tolist() # carries out the train test split and the ML prediction
    X = final_df.drop(['label'], axis=1)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)


    gbc = GradientBoostingClassifier()
    gbc.fit(X_train, y_train)
    predicted_labels = gbc.predict(X_test)
    clf_rep = classification_report(y_test, predicted_labels, output_dict=True)
    acc = clf_rep['accuracy']
    fm = clf_rep['weighted avg']['f1-score']
    print('User', i)
    print('Target Labels', y_test)
    print('Predicted Labels', predicted_labels)
    print('Accuracy:', acc)
    print('Fm:', fm)
    print('------------------------------')

    acc_list4.append(acc)
    fm_list4.append(fm)


with open('gbc_model.pkl', 'wb') as file:
    pickle.dump(gbc, file)

final_acc = sum(acc_list)/len(acc_list)
final_fm = sum(fm_list)/len(fm_list)


print(' ')
print('############################')
print('##########RESULTS##########')
print('############################')
```

Figure 14. Gradient Boosting Classifier

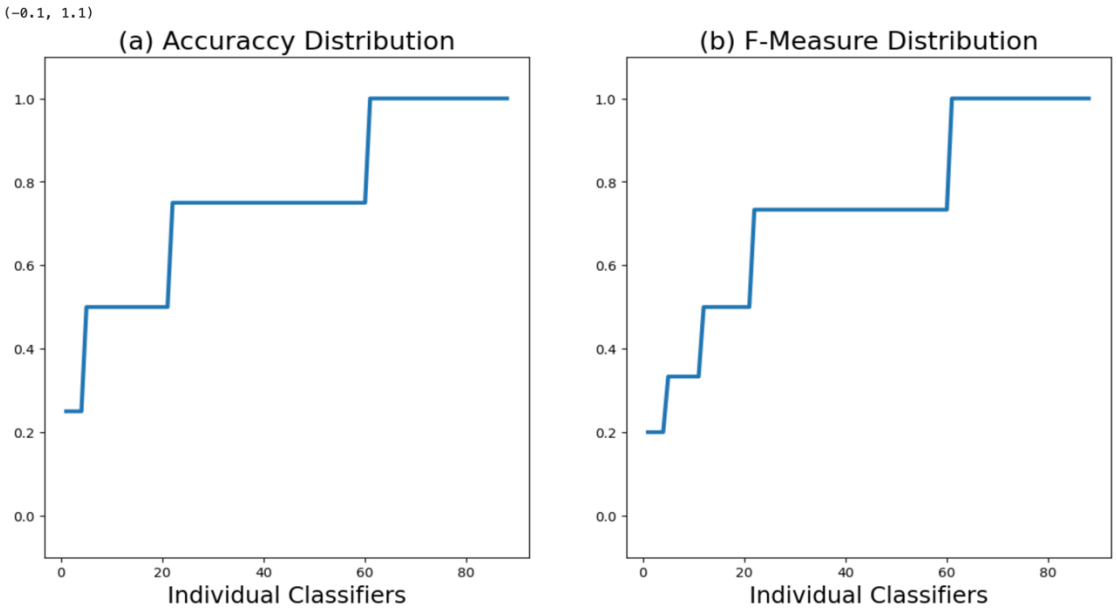


Figure 15. F-Measure and Accuracy distribution among the classifiers

To apply machine learning methods for the classification task, we trained all the various models as seen in Table 6.1 above on the proposed dataset, splitting the dataset into training and testing sections of 80% train and 20% test. Table 6.1 also shows each of the model's cumulative accuracy and F-Score with regard to the 88 classifiers. I used the scikit-learn implementation of Decision Tree in its default setting; Random Forest and Gradient Boosting had n_estimators values of 150 and 100 respectively with Gradient Boosting having a learning rate of 0.1. The Support Vector Machine used the 'kernel' and 'gamma' hyperparameters. The table also shows the minimum and maximum weighted F-Score and accuracy obtained from each of the 88 classifiers for each of the models. It can also be seen that all the models have similar scores of 76.98% mean accuracy and 75.54% mean F-measure.

6.2 Testing the Messaging Web Application

Inputs: User A (researcher's right hand) registers their Full Name, Address, Phone Number, Username, Education, and password on the messaging web application and their keystroke and scroll events are calculated, logged, and converted to JSON string and the dwell times, flight times, average trajectory, and average scroll delta are combined to form a feature vector. The system stores the username, password, keystroke and scroll events in a temporary dictionary on the local drive while the website is running. User A is successfully registered and is directed to the messaging platform. User B (researcher's left hand), acting as a different and unauthorized user, can now type in the message box. User B's keystroke and scroll events are also calculated, logged, and converted to json string and are combined to form a feature vector.

Expected Results: The trained model should compare User B's feature vector to the feature vector stored in the database users_db and return a value of 1 for legitimate users and 0 for illegitimate users. If the user is determined to be illegitimate this website should log User B out with a message stating 'Session expired due to keystroke and scroll mismatch!'. Authentication should happen in 60-second intervals as programmed in the script.

Actual Results: The system was able to distinguish between User A (left hand) and User B (right hand), logging User B out of the web app by halting typing and returning the message 'Session expired due to keystroke and scroll mismatch!' in just about 60 seconds. However, the web application had inconsistent results as it could not always keep a particular legitimate user logged in and continuously authenticated for a sustainable duration of time. This means that genuine users may sometimes be tagged as illegitimate and logged out of the application.

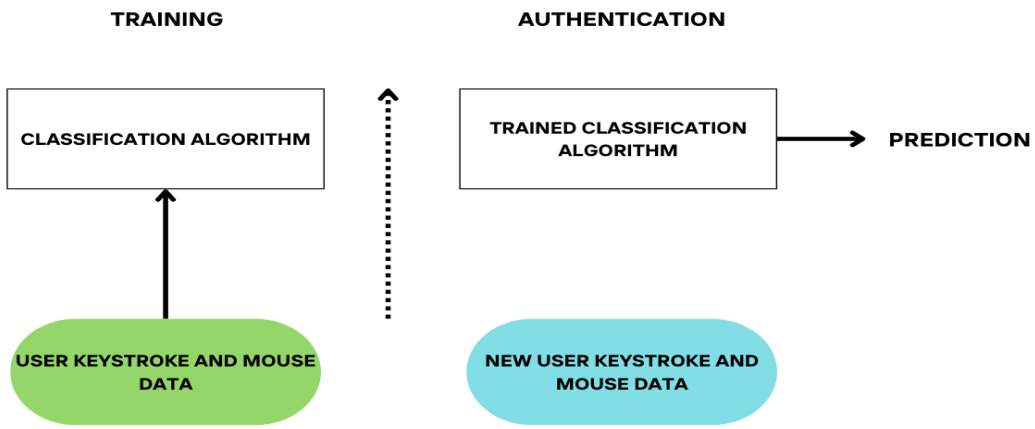


Figure 16. A Graphic Representation of the Continuous Authentication Flow

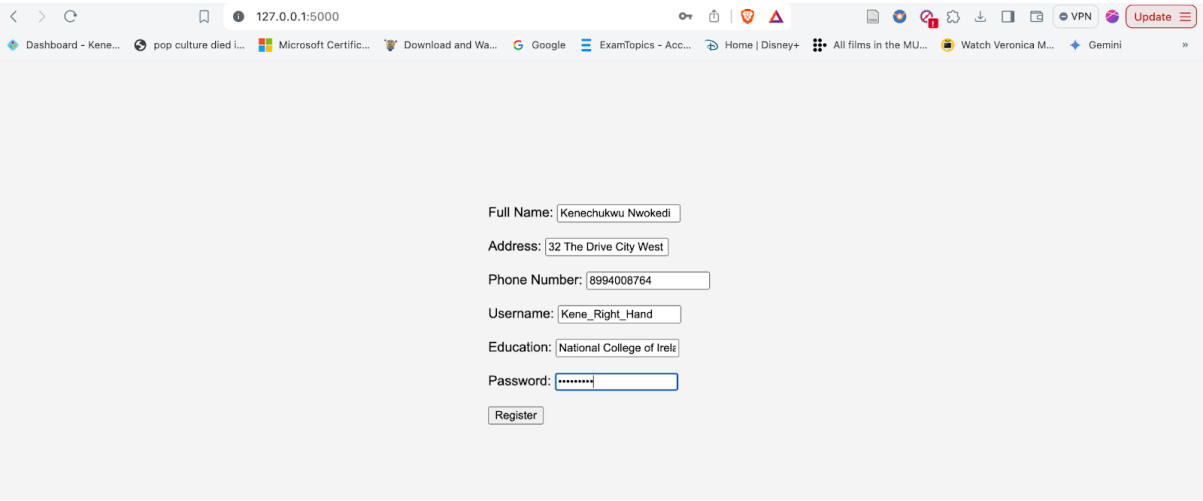


Figure 17. A Display of the Registration Page Where the User Inputs Various Details Including Full Name, Address, And Username

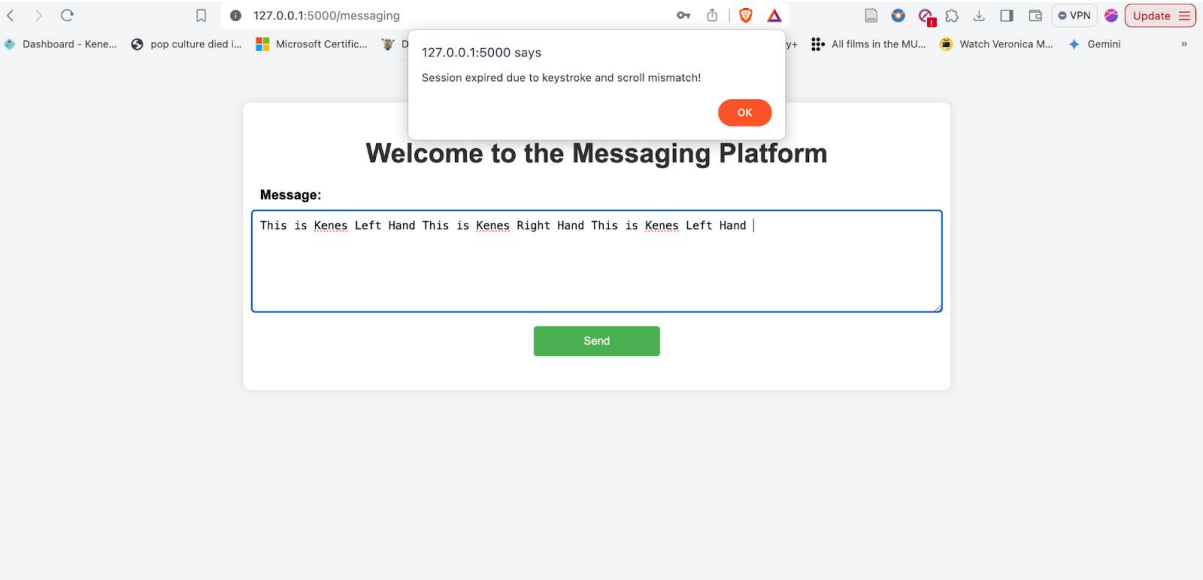


Figure 18. A Display of the Messaging Platform Page Detailing a Scenario where the User has been logged out due to a keystroke and scroll mismatch

6.2 Discussion

To evaluate the performance of the machine learning models, we collected metrics such as the F measure (or F1) score and the accuracy since one of this research's primary objectives was to evaluate various machine-learning models and their ability to examine and interpret acquired data. More importantly, we were interested in observing how they distinguished between different users and whether they could identify the data or not. Although the accuracy metric is important in this study, it can be argued that the F1 is even more important as it indicates whether the model's recall is strong. The strong recall points to a low FPR (False Positive Rate) and FNR (False Negative Rate) (Pelto, Vanamala, and Dave 2023). As this research is based on a real-world application, a high FPR might allow a malicious actor to obtain unauthorized access and a high FNR might prevent a legitimate user from accessing the messaging platform. Additionally, it can be seen from Table 6.3 that the selected models all produced similar mean accuracy and F measure values indicating a consistency in authentication performance. This could be because the data pre-processing implementation was uniformly applied across the models and so classifier inputs were similar. This may also be a result of limited extracted features (Mondal and Bours 2015). If more time had been assigned to this project, I would have obtained more detailed features, for example, average dwell time to average flight time ratio, for more distinct results across models. Extraction of more behavioral patterns as well as features would have helped the model learn more about a given user and hence improved the predictive accuracy. Additionally, since the default configurations for some of the classifiers were used, hyperparameter tuning could also have helped to reveal in-depth performance potential of the models. DTC, GBC, and SVM had the same minimum and maximum F1 score and accuracy values among users, which means the models performed well in some classes and faltered in others. In comparison to previous research in the literature review section, which often required additional hardware

requirements (like TPM or smart cards) or steps for enhanced authentication, the results achieved in this study prove that a more seamless, frictionless method of continuously authenticating users is possible without jeopardizing the user friendliness of the application or system. All users need to do is continue with their regular typing activities and the continuous authentication is performed without much hassle. The results of this study also proved that some of the previous studies were accurate in their assessment of deep machine learning such as D-CNN and LSTM-RNN being able to outperform other machine learning models as they achieved peak accuracy scores within the 92% range – the highest accuracy score for this study was around 77%. SVM was selected for the messaging application created for this study because it is one of the most useful in real-world scenarios where datasets are usually imbalanced, which means that there are noticeably differing numbers of samples in various classes (EITCA. 2023). In cases regarding fraud detection, SVM is especially useful because the number of fraudulent transactions is typically far lower than the number of valid transactions and SVM assigns the fraudulent samples more weight. In this sense, SVMs can manage unbalanced datasets by giving samples from various class weights. SVM was also selected for its versatile kernel function. Two factors must be considered when training a Support Vector Machine using the Radial Basis Function kernel: C and gamma (Scikit-learn 2019). The parameter C which all SVM kernels have, balances the simplicity of the decision surface against training example misclassification. A high C seeks to accurately classify every training example while a low C smooths the decision surface whereas Gamma highlights the degree of impact a single training example has. In this case, SVMs makes use of kernel functions to handle data that is both linearly and nonlinearly separable by converting the original input space into a higher-dimensional feature space where the data points can be separated linearly. Furthermore, since the machine learning models were trained and extracted through the online Colab platform, frequent errors were encountered when trying to load some of the models and incorporate them in the Visual Studio space due to either incompatibility between versions of the scikit-learn library or the environments themselves. Efforts should be made to ensure platform consistency and library version consistency between environments.

7 Conclusion and Future Work

The main aim of this study was to build a custom messaging platform that will continuously monitor user behaviour in real-time by analysing their behavioural biometrics, such as typing and mouse patterns, and ensuring that only authenticated users can access and interact with the platform. The idea was to gain an in-depth understanding of how effective behavioural biometrics enhances security as opposed to conventional authentication approaches. To achieve this, different appropriate machine learning models (DTC, RFC, SVM, GBC) were developed, compared, and trained using a raw dataset to differentiate between real users and fake/potential intruders. After the trained model was integrated into the messaging platform, tests were further conducted to evaluate the platform's accuracy in distinguishing between users and hence its ability to prevent unauthorized access, and its usability.

7.1 ACHIEVEMENT OF OBJECTIVES

This study accomplished its research objectives by following a step-by-step implementation and methodology. The researcher trained the raw dataset, performing classification using well-established machine learning algorithms such as Decision Tree Classifier, Random Forest Classifier, Support Vector Machine, and Gradient Boosting Classifier. Promising results were produced, compared, and analyzed to determine effectiveness regarding user identification and authentication. Furthermore, feature extraction was performed on the input data to prioritize relevancy, calculating for the average flight time, dwell time, and mouse trajectory. Additionally, a custom messaging platform in the form of a Flask application was designed and the selected trained model was integrated into the platform using the Flask Python framework. Lastly, the messaging platform was evaluated for security and usability level, ensuring that only authenticated users could access and interact with the platform and that the authentication process was seamless without sacrificing user friendliness.

7.2 KEY FINDINGS

This research produced some significant findings. Firstly, machine learning algorithms, when implemented and enhanced through feature extraction with a relevant dataset of reasonable size and quality, can effectively classify users of a system as real or illegitimate. Secondly, the Flask Python framework can be used to build web messaging applications easily utilizing only a single Python file without requiring any overly complex code or specific directory structure. Additionally, by analysing keystroke and scroll patterns, the system can effectively classify users and hence, enhance security. An extra layer of protection is a perk of this method, ensuring that even if login credentials are compromised, unauthorized access can still be detected and prevented. Furthermore, a trained machine learning model used to identify users can be integrated into applications requiring high-level security to perform user authentication and enhance security of said systems.

7.3 IMPLICATIONS AND EFFICACY

This research has noteworthy ramifications. A continuous authentication system that grants access to a messaging platform by analyzing biometric data helps in increasing security and providing quick and early detection of fraudulent activity. The efficiency of the research is evident in the high level of accuracy and F-measure of the machine learning classification as well as the platform's ability to spot and log out potential fake users whose keystrokes and mouse data do not match with saved user data. The research shows that a methodological approach to model training techniques and behavioral biometric data analysis and implementation can provide a more secure, reliable, and seamless means of authentication than other traditional means like passwords or fingerprints. The research is also useful in the context of helping other engineers and/or professionals build secure applications whose main purpose is to perform seamless user authentication by analyzing biometric user behavior.

7.4 LIMITATIONS

Despite the potential of this research, there are some limitations. Firstly, the strength of the system is diminished as it only considers specific interactions like messaging and registration and doesn't really consider other kinds of activities that the user may perform, for example, editing a document. Secondly, the system is based on a trained model for distinguishing users. The performance of the model diminishes when the test dataset is not representative of all the possible users of the system and this manifests itself in increased false rejection or acceptance rate. Hence, capturing subtleties in different users' behavior is essential to how the model performs. Furthermore, False Positives and Negatives result in the false rejection of legitimate users (Pelto, Vanamala, and Dave 2023), either due to natural changes in behavior, such as the typing speed or mouse scrolls. As such, false rejection of the users can lead to frustration and severely affect user experience. Finally, typing and scrolling behavior may vary after some time due to a variety of controllable and uncontrollable factors such as tiredness, stress, or hardware change (Teh et al., 2016). The authentication system may not accommodate these changes and increase false rejection rates. Hence, adaptive learning techniques to accommodate changes in user behavior are important if accuracy must be sustained over time (Ahmed and Traore, 2007).

7.5 FUTURE WORK AND COMMERCIALIZATION

For larger commercial purposes, this research opens avenues for developing robust and adaptable continuous authentication systems that cater to increasingly complex cyber threats. The potential for integration into applications for authentication is significant. Future research could focus on:

- Securing storage and transmission of the keystroke and scroll data to prevent data interception or hijacking by malicious actors.
- Exploring and implementing ensemble techniques that will combine the strengths of various classifiers to further enhance the results for user authentication.
- Developing techniques to detect and curb behavioral spoofing which will enhance the legitimacy of the authentication process.

References

Abuhamad, M., Abusnaina, A., Nyang, D.H. and Mohaisen, D. (2020). 'Sensor-based Continuous Authentication of Smartphones' Users Using Behavioral Biometrics: A Contemporary Survey. *IEEE Internet of Things Journal*, pp.1–1. doi:https://doi.org/10.1109/jiot.2020.3020076.

Colnago, J., Devlin, S., Oates, M., Swoopes, C., Bauer, L., Cranor, L. and Christin, N. (2018). 'It's Not Actually That Horrible'. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. [online] doi:https://doi.org/10.1145/3173574.3174030.

de-Marcos, L., Martínez-Herráiz, J.-J., Junquera-Sánchez, J., Cilleruelo, C. and Pages-Arévalo, C. (2021). Comparing Machine Learning Classifiers for Continuous Authentication on Mobile Devices by Keystroke Dynamics. *Electronics*, 10(14), p.1622. doi:https://doi.org/10.3390/electronics10141622.

Dowling, B., Günther, F. and Poirrier, A. (2022). Continuous Authentication in Secure Messaging. *Lecture Notes in Computer Science*, pp.361–381. doi:https://doi.org/10.1007/978-3-031-17146-8_18.

Edu, J., Mulligan, C., Pierazzi, F., Polakis, J., Suarez-Tangil, G. and Such, J. (2022). Exploring the Security and Privacy Risks of Chatbots in Messaging Services. *Proceedings of the 22nd ACM Internet Measurement Conference*. doi:https://doi.org/10.1145/3517745.3561433.

FBI (2023). *Internet Crime Report 2023*. [online] Department of Justice. Available at: https://www.ic3.gov/Media/PDF/AnnualReport/2022_IC3Report.pdf [Accessed 1 Jul. 2024].

FBI (2024). *Internet Crime Report 2023*. [online] Department of Justice. Available at: https://www.ic3.gov/Media/PDF/AnnualReport/2023_IC3Report.pdf [Accessed 1 Jul. 2024].

Mekruksavanich, S. and Jitpattanakul, A. (2021). Deep Learning Approaches for Continuous Authentication Based on Activity Patterns Using Mobile Sensing. *Sensors*, 21(22), p.7519. doi:https://doi.org/10.3390/s21227519.

Nnamoko, N., Barrowclough, J., Liptrott, M. and Korkontzelos, I. (2022). A Behaviour Biometrics Dataset for User Identification and Authentication. *Data in Brief*, 45, p.108728. doi:https://doi.org/10.1016/j.dib.2022.108728.

Siddiqui, N., Dave, R., Vanamala, M. and Seliya, N. (2022). Machine and Deep Learning Applications to Mouse Dynamics for Continuous User Authentication. *Machine Learning and Knowledge Extraction*, 4(2), pp.502–518. doi:https://doi.org/10.3390/make4020023.

Carneiro, T., Medeiros Da Nobrega, R.V., Nepomuceno, T., Bian, G.-B., De Albuquerque, V.H.C. and Filho, P.P.R. (2018). 'Performance Analysis of Google Colaboratory as a Tool

for Accelerating Deep Learning Applications', *IEEE Access*, 6, pp.61677–61685. doi:https://doi.org/10.1109/access.2018.2874767.

Quraishi, S.J. and Bedi, S.S. (2022). 'Secure System of Continuous User Authentication Using Mouse Dynamics', *3rd International Conference on Intelligent Engineering and Management (ICIEM)*, pp. 138-144. doi: 10.1109/ICIEM54221.2022.9853050.

Dyouri, A. and Hancox K. (2020) *How To Make a Web Application Using Flask in Python 3.* [online] DigitalOcean. Available at: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3 [Accessed 17 July 2024].

Nnamoko, N., Korkontzelos, I., Barrowclough, J. and Liptrott, M. (2022). 'CyberSignature: A user authentication tool based on behavioural biometrics.' *Software Impacts*, 14, p.100443. doi:https://doi.org/10.1016/j.simpa.2022.100443.

Naidu, G., Zuva, T. and Sibanda, E.M. (2023) 'A Review of Evaluation Metrics in Machine Learning Algorithms', *Artificial Intelligence Application in Networks and Systems*. 724. https://doi.org/10.1007/978-3-031-35314-7_2

Xi, T., Kuzminykh, I., Ghita, B. and Bakhshi, T. (2023) Evaluating Learning Algorithms for Keystroke Based User Authentication. *2023 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 283-288, doi: 10.1109/BlackSeaCom58138.2023.10299695.

Nnamoko, N., Barrowclough, J., Liptrott, M. and Korkontzelos, I. (2022) *Behaviour Biometrics Dataset.* DOI: 10.17632/fnf8b85kr6.1. Available at: https://data.mendeley.com/datasets/fnf8b85kr6/1 [Accessed 17 July 2024].

Scikit-learn (2019). *scikit-learn machine learning in Python.* Available at: https://scikit-learn.org/stable/. [Accessed 17 July 2024].

Scikit-learn. (2023) *1.4. Support Vector Machines* Available at: https://scikit-learn.org/stable/modules/svm.html [Accessed 17 July 2024].

Scikit-learn. (2023) *SVC* Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html [Accessed 17 July 2024].

Scikit-learn. (2023) *Random Forest Classifier* Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [Accessed 17 July 2024].

Scikit-learn. (2023) *Gradient Boosting Classifier* Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html [Accessed 17 July 2024].

Scikit-learn. (2023) *1.10. Decision Trees* Available at: https://scikit-learn.org/stable/modules/tree.html [Accessed 17 July 2024].

R. Aswani (2023) *Keep Data Safe: 2023 Guide to Secure Communication Systems*. Available at: https://www.kumospace.com/blog/secure-communication. [Accessed 17 July 2024].

Singh, A., Thakur, N. & Sharma, A. (2016). 'A review of supervised machine learning algorithms.' *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 1310-1315.

PyPi. (2024) *Pandas 2.2.2* Available at: https://pypi.org/project/pandas/. [Accessed 17 July 2024].

Python Software Foundation (2023) *Math — Mathematical Functions* Available at: https://docs.python.org/3/library/math.html. [Accessed 20 July 2024].

Millikin, J. (2010) *jsonlib 1.6.1* Available at: https://pypi.org/project/jsonlib/ [Accessed 21 July 2024].

PyPi. (2024) *matplotlib 3.9.1* Available at: https://pypi.org/project/matplotlib/. [Accessed 21 July 2024].

NumPy (2022) *NumPy documentation* Available at: https://numpy.org/doc/stable/. [Accessed 21 July 2024].

Salunke, S.V. and Ouda, A. (2023) 'Ensemble Learning to Enhance Continuous User Authentication For Real World Environments,' *2023 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. pp. 102-108, doi: 10.1109/BlackSeaCom58138.2023.10299704.

Teh, P. S., Zhang, N., Teoh, A. B. J., & Yue, S. (2016). 'A survey on touch dynamics authentication in mobile devices', *Computers & Security*, 59(C), pp. 210-235.

European Information Technologies Certification Academy (EITCA) (2023) *What are some advantages of using support vector machines (SVMs) in machine learning applications?* Available at: https://eitca.org/artificial-intelligence/eitc-ai-mlp-machine-learning-with-python/support-vector-machine/support-vector-machine-introduction-and-application/examination-review-support-vector-machine-introduction-and-application/what-are-some-advantages-of-using-support-vector-machines-svms-in-machine-learning-applications/#:~:text=Effective%20in%20high%2Ddimensional%20spaces,than%20the%20number%20of%20samples [Accessed 17 July 2024].

Pelto, B., Vanamala, M. and Dave, R. (2023) 'Your Identity is Your Behavior - Continuous User Authentication based on Machine Learning and Touch Dynamics', *3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. pp. 1-6. doi: 10.1109/ICECCME57830.2023.10252828.

Dhaka, M., Rao, U. P., and Chaurasia, A. (2022) 'User Authentication using Behavioural Biometrics with Machine Learning' *IEEE International Conference on Advanced Networks and Telecommunications Systems*, pp. 255-260. doi: 10.1109/ANTS56424.2022.10227806.

Shaik, A.B. and Srinivasan, S. (2019) 'A Brief Survey on Random Forest Ensembles in Classification Model' *International Conference on Innovative Computing and Communications*, 56, pp 253–260.

Bahad, P., Saxena, P. (2020) 'Study of AdaBoost and Gradient Boosting Algorithms for Predictive Analytics.' *International Conference on Intelligent Computing and Smart Communication Algorithms for Intelligent Systems*. pp 235–244 https://doi.org/10.1007/978-981-15-0633-8_22

Namnaik, P., Kurale, R. and Mahindrakar, S. (2018) 'Keystroke Dynamics For User Authentication', *International Research Journal of Engineering and Technology (IRJET)*, 5(4)

Ahmed, A. A. E. and Traore, I. (2007) 'A New Biometric Technology Based on Mouse Dynamics', in *IEEE Transactions on Dependable and Secure Computing*, 4(3), pp. 165-179. doi: 10.1109/TDSC.2007.70207.

Monrose, F. and Rubin, A.D. (2000) 'Keystroke dynamics as a biometric for authentication', *Future Generation Computer Systems*, 16(4), pp.351-359.

Salunke S.V. (2022) 'Behavioral Biometrics-based Continuous User Authentication', *Electronic Thesis and Dissertation Repository*, 9073, pp.1-92.

Deutschmann, I., Nordström, P. and Nilsson, L. (2013) 'Continuous Authentication Using Behavioral Biometrics', *IT Professional*, 15(4), pp.12-15. doi: 10.1109/MITP.2013.50.

Dybczak, J. and Nawrocki, P. (2022) 'Continuous authentication on mobile devices using behavioral biometrics' *22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pp. 1028-1035. doi: 10.1109/CCGrid54584.2022.00125.

Mondal, S. and Bours, P. (2015) 'Continuous Authentication in a real world settings' *2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR)*, pp. 1-6. doi: 10.1109/ICAPR.2015.7050673.

Chen, L., Zhong, Y., Ai, W. and Zhang, D. (2019) ‘Continuous Authentication Based on User Interaction Behavior’, *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1-6. doi: 10.1109/ISDFS.2019.8757539.

Salau, A.O. and Jain, S. (2019) 'Feature Extraction: A Survey of the Types, Techniques, Applications'. *2019 International Conference on Signal Processing and Communication (ICSC)*, pp. 158-164. doi: 10.1109/ICSC45622.2019.8938371.