Table of Contents

[Aims and Objectives: 1](#_Toc88151772)

[Survey of Literature: 2](#_Toc88151773)

[Requirements: 2](#_Toc88151774)

[Outline of Specification and Design: 2](#_Toc88151775)

[Planning and Timescales: 3](#_Toc88151776)

[Bibliography 5](#_Toc88151777)

# Aims and Objectives:

The primary aim of the project is to develop a keystroke dynamic based authentication that ensures users can securely use their systems without the possibility of malicious users taking advantage. The system will run in the background and provide constant protection of the user. The secondary aim of the project is to ensure that the systems performance is as good as possible so as not to hinder the user.

In today’s world, passwords aren’t enough in order to guarantee security. Many new methods have been tried with fingerprints and facial recognition being at the forefront of these. While these methods are good for one time login, they don’t prevent malicious users taking control after a legitimate user has logged in. This is where keystroke dynamics is useful. By analysing the keystrokes and building up a profile for the user, a keystroke dynamics-based system can easily ensure that the system is being used by a legitimate user.

Keystroke dynamics-based systems aren’t widespread and as such not many commercially available systems exist. My system is different to the few available because it’s designed to run continuously and has a particular emphasis on performance. Furthermore, I plan to see if I can improve any of the algorithms or systems currently in use.

The system is being primarily created in python 3.9 and has a key emphasis on security and performance. The key objectives are:

1. To produce a lightweight keylogger using python to log all inputs to the system accurately and securely
2. To create a graphical system in python that allows the user to register or login. This system will also allow the system to create and store profiles for that user
3. At an interval set by the user, the system will create a profile for the user based on the typing since the last interval and check this against the profile created in the registration system.
4. If the user doesn’t match, ask the current user to re-authenticate

The system will implement Fishers Linear Discriminant (FLD) which is a machine learning method in order to improve accuracy and reduce high-dimensional data into lower-dimensional data. Along with this, it will also implement feature fusion methods in order to try and combine all the features extracted from the data. In order to improve performance, the system will utilise clustering when storing the profiles to ensure that the system is lightweight and to ensure it’s not a burden when the user is using it.

# Survey of Literature:

To prepare to start this project, I’ve studied many different papers from lots of different areas.

To begin with and gain a basic understanding of how keystroke dynamics authentication works, I read *Time-frequency analysis of keystroke dynamics for user authentication* [1] which presented a very basic system that uses keystroke dynamics along with pure maths-based approach in order to authenticate the user. Unfortunately, this system wasn’t a continuous system, but the paper still provided some value explaining the topic and allowing me to get a basic understanding of how it works.

The paper *Keystroke dynamics-based authentication service for cloud computing: Keystroke Authentication Service* [2]showcase a different approach from the one above in that it makes use of more of a machine learning approach which is different from the previous papers purely maths-based approach. This paper also presents a continuous system which was useful when deciding to create my own.

Research into open-source examples was also undertaken with a look at a keystroke analysis system created by Nikolai Janakiev [3]. This system was very interesting and provided a practical look at how some had implemented a theoretical system using machine learning. The system shown was accurate but at the cost of performance. This system was based upon the paper *Comparing anomaly-detection algorithms for keystroke dynamics* [4]which was a very useful paper on neural network-based keystroke dynamic based authentication.

The paper *Keystroke Dynamics-Based Authentication Using Unique Keypad* [5]presents a system that is only used when logging in with a unique keypad which although not relevant in my case presented another way to make a keystroke dynamics authentication system.

I believe that my current research has given me a good understanding of already existing keystroke dynamics authentication approaches and will allow me to produce a working system, but I believe that I will have to do further research when I implement my secondary aim which is to ensure that the system is lightweight and not a hinderance to the user.

# Requirements:

# Outline of Specification and Design:

# Planning and Timescales:

The system will be written exclusively in python 3.9.1. This is because python provides a wide range of abilities which this project will need such as a UI along with capability to do advanced math which the NumPy package provides the ability to do. The program is designed to be running in the background most of the time and as such, performance was a huge factor when choosing a language as the user ideally wouldn’t even feel it running in the background. Using python along with packages like NumPy ensures that the program runs as lightweight as possible.

The program has four separate components. These are:

* Keylogger
* Login/Registration UI
* Backend Maths Component
* Database

A diagram of how each component will interact with each other is detailed in Figure 1.

Diagram

Description automatically generated

Figure : High Level Architecture of the system

Figure 1 illustrates the high-level view of my system. The system will start with the registration UI where users enter their personal details, and the system learns what the user types like. These personal details are stored inside a secure database which will be implanted in SQLite due to its secure and lightweight nature. The registration system is written in python and will consist of a simple registration screen consisting of a username, password and email input. Then the user will be prompted to type some text so that the system can then learn how the user types. This text will be captured using the keylogger which is once again written in python and go into the maths backend which is explained below. Once a profile has been generated for the user, then this is then stored inside the data storage. The system will then return to the keylogger where at an interval set by the user, the system will once again feed back to the maths backend. If the maths backend determines that the user is different than the profile stored, it will return the user to the login UI where the user will be prompted to enter their username and password. These will be checked against the values stored in the database. If they match, the new profile will be stored in the data storage and the system will return to it the keylogger state and then repeat the process at the interval set by the user.

The text in the registration system will be chosen randomly and will cover all edge cases in the way people type in order to avoid false positives or false negatives. This will involve text generation that uses most of the keyboard in order to accurately reflect the users typing. This will be done in python 3.9.1 due to its performance and ease of doing complicated mathematical equations that will be required to generate the text.

A more detailed diagram of the registration and how it interacts with the data storage is shown in Figure 2.

Diagram

Description automatically generated

Figure : Detailed look at the registration system

While the user is typing the keystrokes are captured by the keylogger. These keystrokes are never stored in order to improve security and alleviate data privacy concerns. Instead, the profile created by the maths backend is stored.

The maths backend consists of five sections:

1. Dynamic Time warping
2. Wigner Distribution
3. KDS
4. 2D correlation co-efficient
5. Similarity Measure

The way in which data flows through these is shown in Figure 3.

A picture containing text, sign

Description automatically generated

Figure : Maths Backend

All the data in the keylogger is fed through this system. The data recorded by the keylogger has several attributes which are:

* Keycode
* Motion (Key Up or Key Down)
* Timestamp this occurred

After even a short time, the amount of data collected is huge and as such it is then put through this system.

First, the data it put though an equation which transforms it into a KD signal (KDS). The value of KDS is at each time, the number of keys currently pressed. For example, when a new key is pressed the value of KDS increased by 1 and as such when a key is released then the value decreases by one. This function makes the use of the Heaviside deep function which will either output 0, 0,5 or 1 depending on the data put in. The biggest difference between any two KD signals is the length of them. The length of a KD Signal could be defined as the pressing of the first key and releasing of the last one.

This causes problems when measuring the similarity of two KD signals. In order to solve this, the KD signal is then put through Dynamic Time Warping (DTW) in order to normalise the data. The DTW makes use of an optimal warping path algorithm which is used to align the data. The rough steps of an optimal warping path algorithm are shown below:

1. Calculate a cost matrix.
2. While the number of rows and columns is greater than 1.
   1. If the number of rows is 1 then set the number of columns to the current value-1 and add both the number of rows and the number of columns to the path. And then return the path.
   2. If the number of columns is 1 and then set the number of columns to the current value-1 and add both the number of rows and the number of columns to the path and then return the path.
   3. Otherwise, if the value in the array at the current value of the number of rows-1 and columns. Compare this to the minimum value from:
      1. Current row-1 and current column
      2. Current row and current column -1
      3. Current row -1 and current column -1

If they match minus 1 from the current row. If they don’t either minus 1 from the current column or minus 1 from both the current row and the current column.

1. Return the current path at the end

Once this occurs, I could perform Euclidean distance calculation on the output in order to see how far apart they are but in order to improve the comparison, a Wigner Distribution is then applied to the output of the data output by DTW.

A Wigner distribution (WD) is used to transform the data into the time-frequency domain. This is done because time frequency analysis using WD ensures that representation of genuine samples has few differences and that WD representations on genuine samples shouldn’t change dramatically. This is another measure that makes comparing the data way easier. While it is a further drain on the users’ computers resources, it’s worth it to improve accuracy.

Finally, in order to compare two signals a 2D correlation co-efficient equation is used. Once again this is done to improve accuracy and ensures that the system can easily spot imposters. Once again, a less complicated measure could have been used instead which would have worked into the lightweight measure, but these can lack accuracy and as such aren’t as useful in a security context. This performance impact is something I will be exploring once the system is developed in order to either test out some other methods or trying to decrease the performance impact that the system has.

The value that the 2D returns a value that will measure how similar the two values are. The higher the value the more closely aligned they are. For example, if the equation returned 1 then that indicates a stronger relationship whereas a value of 0 returned indicates no relationship. In the systems case, a higher value when comparing the data from the current interval to the data in the training data shows that the same user is using the user’s computer whereas a lower value indicates an imposter.

# Bibliography

|  |  |
| --- | --- |
| [1] | R. Toosi and M. A. Akhaee, “Time-frequency analysis of keystroke dynamics for user authentication,” *Future Generation Computer Systems,* vol. 115, pp. 438 - 447, 2021. |
| [2] | A. Abo-alian, N. L. Badr and M. Tolba, “Keystroke dynamics-based user authentication service for cloud computing,” *Concurrency and computation,* vol. 28, no. 9, pp. 2567 - 2585, 2016. |
| [3] | N. Janakiev, “Biometric Prediction on Keystroke Dyanamics,” 7 June 2018. [Online]. Available: https://github.com/njanakiev/keystroke-biometrics. [Accessed 10 November 2021]. |
| [4] | K. S. Killourhy and R. A. Maxion, “Comparing anomaly-detection algorithms for keystroke dynamics,” in *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, Lisbon, 2009. |
| [5] | M. Choi, S. Lee, M. Jo and J. S. Shin, “Keystroke Dynamics-Based Authentication Using Unique Keypad,” *Sensors,* vol. 21, no. 6, p. 1424, 2021. |