23/11/2021

Francis, Jack T.

School of Computing and Mathematical Sciences, University of Leicester

Interim Report

Using Keystroke Dynamics to Authenticate a User Based On their Typing

CO2301 Computer Science Project

Table of Contents

[Aims and Objectives: 2](#_Toc88570822)

[Survey of Literature: 2](#_Toc88570823)

[Requirements: 4](#_Toc88570824)

[Outline of Specification and Design: 4](#_Toc88570825)

[Planning and Timescales: 9](#_Toc88570826)

[Bibliography 11](#_Toc88570827)

Text, letter

Description automatically generated

# Aims and Objectives:

The primary aim of the project is to develop a keystroke dynamic based authentication system 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 3 different algorithms which will all work together in order to ensure that the system is functioning correctly. These are:

* Dynamic Time Warping
* Wigner’s Distribution
* 2D Correlation Co-efficient

The system will also implement clustering which will improve performance. More on these are explained in the Outline of Specification and Design section below.

# 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. I decided to rely on this paper to base my system off because it was well-cited, researched, presented a solid approach I couldn’t fault, and it was published recently and built on many others attempts at a similar system.

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 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. After reading and understanding this paper, I decided not to continue with a machine-learning based approach and as such this paper wasn’t that useful to me and I couldn’t rely on it.

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. This unique keypad element of this paper wasn’t useful due to my system not using a unique keypad. However, the approach they used under the unique keypad was useful and could have been used in any system, with or without a unique keypad. I didn’t end up relying on this paper because I judged the approach type to be too theoretical in nature and not capable of being translated easily into my continuous system due to its performance drain and machine-learning based nature.

*The Wigner Distribution: A Time-Frequency analysis tool* [6]was a good resource for further understanding the Wigner distribution which is a key algorithm in the system. The paper goes into extreme detail about this algorithm and as such was very useful in combination with the paper which originally presents the approach using the algorithm. Although the source is old, and as such may have been eclipsed by new sources, it was detailed and cited in a few other papers I read on this topic. Therefore, I’m going to rely on it when doing that part of the project.

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:

Currently, I’ve been using the waterfall methodology in order to develop this project, but after some thought, I’ve decided to make the switch to the Scrum methodology instead. This is because in the waterfall methodology, the process is sequential and there is no reflection on the work done so far. Furthermore, you can also not move backwards in this methodology which I feel will become an issue in the testing section of my plan where I test out elements of my systems and make changes if need be. If I was to continue using this methodology, I would be forced to restart the entire project. In Scrum this is not the case and due to its iterative and self-reflective nature would increase the quality of the system. Scrum employs two-week sprints. At the start, you choose issues to work upon that sprint. At the end of the sprint, you once again choose some new issues to work upon. If at the end of the sprint, work remains to be done in the issues you chose at the start, then you continue and put issues not yet started in the spring backlog.

Requirements for this project are listed here:

1. The system should be able to tell genuine and imposter users apart with a high degree of accuracy
2. The system should learn how the user types inside the registration phase
3. If the system detects a new typing style and they authenticate correctly using correct login details, the system should change the profile stored for that user.
4. The system should allow the user to securely login
5. The system should be able to store the user’s login details securely and efficiently
6. The systems should have good performance and not impact the user
7. The system should run in the background and eject imposters if they do not exceed or match the typing threshold set.
8. The system should have a reasonable typing threshold to ensure that a genuine user is not incorrectly locked out.

These cover the core success requirements of my system and provide a reasonable basis. On top of this are some extra functionalities which is not essential to the success of the system:

1. The system could use clustering to improve performance
2. The system could let the user adjust the interval between when the system checks the typing

# Outline of Specification and Design:

The system will be written exclusively in python 3.9.1. This is because python is a versatile language 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 six separate components. These are:

* Keylogger
* Login
* Registration System
* Backend Maths Component
* Database
* Profile Data Storage

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 system 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 MySQL 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 keylogger 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.

I haven’t yet made a concrete decision on the structure of the profile storage but it’s likely this will take the form of table in a secure MySQL database. MySQL was chosen because of it’s ease of install and ease of use. Furthermore, it’s incredibly performance minded and supports clustering. All details of the user will be encrypted using SHA256 and a hashing function and then that stored in a secure database with the structure shown in Figure 2.

Text

Description automatically generated with low confidence

Figure : Database structure

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 3.

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. Clustering will also be used to store these profiles which will improve performance which once again is a huge part of the system. Clustering is a technique in databases where all similar data is stored next to one another so when the system for something within the database it can jump straight to different sections rather than having to scan the entire database. The impact of this can be felt in large datasets which this system should unfortunately generate.

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 4.

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 which will make spotting imposter samples easier. 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. This is known as the typing threshold. A large part of the project will be determining at what values the system thinks a user is genuine and at what values a user is an imposter. For example, setting a value too high such as 80% similarity would lead to genuine users being ejected too many times but a value too low such as 20% would lead to imposters getting let in too many times.

I plan on determining this value in the testing phase in my plan. To determine the correct value that strikes a balance between too high or too low, I plan on doing extensive testing using genuine and imposter users to do so.

# Planning and Timescales:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Start Date | End Date | Semester | Status |
| Research into existing approaches | 06/10/2021 | 10/11/2021 | 1 | Completed |
| Familiarising myself with and understanding the mathematical element | 20/10/2021 | 12/11/2021 | 1 | Completed |
| Creating a basic Keylogger prototype | 15/11/2021 | 19/11/2021 | 1 | Completed |
| Creating a more advanced Keylogger prototype | 22/11/2021 | 06/12/2021 | 1 | In Progress |
| Implementing profile storage | 07/12/2021 | 22/12/2021 | 1 | In Progress |
| Implementing secure login details storage | 23/12/2021 | 31/12/2021 | 1 | Not started |
| Implementing the mathematical basis for the system | 03/01/2021 | 20/01/2022 | 1 | Not started |
| Implementing the Dynamic Time Warping Algorithm | 21/01/2022 | 02/02/2022 | 2 | Not started |
| Implement Wigners Distribution | 03/02/2022 | 18/02/2022 | 2 | Not started |
| Implementing the similarity measure | 21/02/2022 | 03/03/2022 | 2 | Not started |
| Implementing the registration system | 04/03/2022 | 16/03/2022 | 2 | Not started |
| Implementing the login system | 17/03/2022 | 25/03/2022 | 2 | Not started |
| Optimisation | 28/03/2022 | 06/04/2022 | 2 | Not started |
| Refactoring | 07/04/2022 | 18/04/2022 | 2 | Not started |
| Testing and Hand In | 19/04/2022 | 05/05/2022 | 2 | Not started |

This is my list of all the tasks that need to be completed before the final version of the system. Some tasks such as implementing the mathematical basis for the system I’ve allocated more for because this is the step in which most of the system is based upon and as such it’s important to get it working efficiently and to get it 100% correct. Furthermore, I decided that having a large optimisation, refactoring and testing stage is important because the way in which the system is designed makes testing difficult with only one person and as such in these stages I will need to make use of other people which will take more time. Furthermore, as performance of the system is a large success condition of the system, so I wanted a lot of time to get as much performance out of it as possible.

The general goal for semester 1 is to develop a good strong foundation for the rest of the project whilst semester 2 has the goal of building on top of this and adding functionality. Looking at the plan for the current year. I would estimate that everything I am around 70% through the workload for this semester which is exactly where I want to be for this semester.

In figure 5, you can see the same data represented in a Gantt chart with the prerequisites shown. In figure 6, is the Gantt chart for just semester 1 and in figure 7 just the tasks for semester 2. A cleaner version of these is stored inside the GitLab inside /Notes.

A picture containing chart

Description automatically generated

Figure : Gantt chart of both semesters

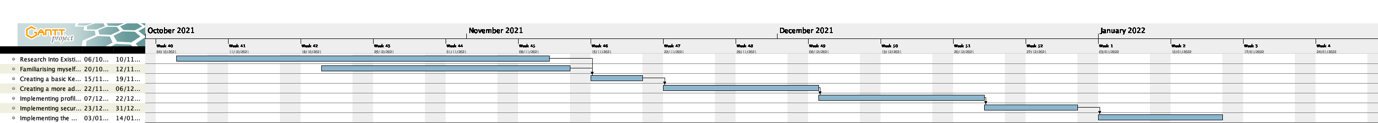


Figure : Gantt chart for Semester 1

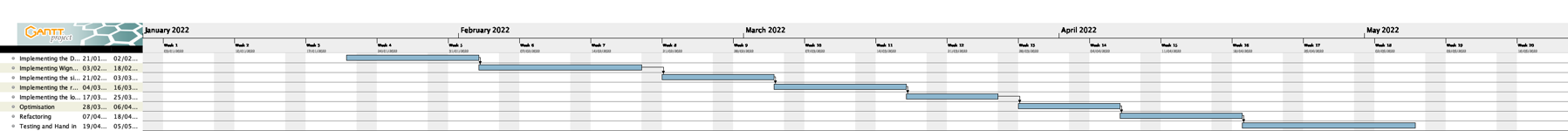


Figure : Gantt Chart for Semester 2

When looking not only at the work already completed but the work coming up, I feel as though I’ve managed to balance my time well and have come up with a good plan to continue doing so. When doing a project such as this, reading and understanding all the approaches is essential and therefore I dedicated so much time to reading and understanding current approaches.

A lot of time is also used implementing the algorithms that are essential to the project. In total, around a month and a half is scheduled to be used implementing 3 algorithms. The reason for this is because as above, these algorithms are essential to the success of the system and any delay caused by an incorrect implementation of these algorithms would significantly affect the success of the project.

Out of all the tasks, the key milestones are:

1. Creating an advanced keylogger
2. Implementing the similarity measure
3. Optimisation

The reason these are the milestones I have chosen is because at each milestone a key part of the project id done. For example, once the creating an advanced keylogger milestone is complete then I know for a fact that I

If at these three stages, I am where I want the project to be, then I would believe that the project is not only on time but a success.

# 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. |
| [6] | A.-H. Y. NAJMI, “THE WIGNER DISTRIBUTION: A TIME-FREQUENCY ANALYSIS TOOL,” *Johns Hopkins APL Technical Digest,* vol. 15, no. 4, pp. 298 - 305, 1994. |