KEYSTROKE AUTHENTICATION

BSC (HONS) COMPUTER SCIENCE

ORE BENSON

CANDIDATE NUMBER: 219685

SUPERVISOR: PROFESSOR PETER CHENG

DEPARTMENT OF INFORMATICS

ACADEMIC YEAR 2022/23

Statement of Originality

This report is submitted as part requirement for the degree of BSc (Hons) Computer Science at the University of Sussex. It is the product of my own labour, except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Acknowledgements

I would like to thank my supervisor, Peter, for providing me the opportunity to continue this research, and encouraging the process. I also thank Ibrahim, for supporting me throughout the course of writing this dissertation.

I would also like to say thanks to my family, for their moral support and belief.

Abstract

User authentication is the process of identifying and granting an individual access to a system. Different forms of it play major roles in our day to day lives, from using keys to enter a house, to entering a pin to access a bank account. The most common application is in data protection, which is becoming increasingly crucial as computer systems have become pivotal to society.

Biometric methods, such as face and fingerprint scanners, use natural biological data to identify a user, and are commonly used where password is not sufficient.

This project investigates the use of an individual's keystrokes as a biometric for user authentication, analysing keyboard inputs, finding habitual patterns, and typing rhythms. To investigate this, a software that accurately records and examines keystrokes was developed, the details of which are specified in this report.

Contents

Project Aim

In today's digital age, authentication is vital, as the lack of a secure system leads to unauthorized access to sensitive data. Current authentication methods such as passwords and PINs have many limitations, such as security, convenience, and user error. Keystroke dynamics offer a potential solution to these limitations, as they are difficult to replicate, and more unique to the user.

The aim of this project is to investigate the use of keystroke dynamics as a method of user authentication. This requires creating unique biometric profiles from user's typing patterns, and finding an algorithm that can accurately identify imposters vs genuine users. Potential benefits and limitations of this method are explored, in this report, discussing real-world applications.

This project outlines the development of a software that accurately records such data, and tests the performance of various classification algorithms, investigating the question; can a secure, reliable system for user authentication via keystroke dynamics be implemented?

Professional Considerations

The research in this project required the use of several participants, whose data was recorded and analysed. The BCS Code of Conduct in relation to this project is outlined below.

Public Interest

Public privacy and security were upheld during conduct. On entering the website, users are required to read extensive information detailing the use of their data for research purposes, data confidentiality, and the approval of the study from the university with contact details. There is also a consent form on the site, which is required to be filled out before any data can be entered. The website is securely hosted on the university servers, sending data through HTTPS to a backend hosted on an Amazon Web Services (AWS) instance. The data is securely stored this password protected instance, which also has a firewall to prevent unauthorized access.

This investigation was carried out without any form of discrimination. The website link for this project was shared indiscriminately to participants above the age of 18, ensuring a competent understanding of how their data will be used.

Professional Competence and Integrity

The competence for this project was developed throughout the author’s Computer Science BCs degree at the University of Sussex and is in line with the Data Protection Act and the BCS Code of Conduct. Criticisms and alternative viewpoints will be valued and met with respect.

Duty to Relevant Authority

This project is in alignment with the Data Protection Act and the ethics code outlined by the University of Sussex. An ethical review was applied for and approved by the university. It describes how the data is used, and the participants are involved in the investigation. A consent form and information sheet were developed during this review and are required to be read and filled out by each participant.

Duty to the Profession

The integrity and reputation of the profession were upheld during this investigation, with all relevant legislations and ethical considerations being adhered to.

Background

As society becomes increasingly reliant on computers, the need for secure authentication methods is more important than ever. However, traditional methods such as passwords are becoming outdated, as users continue to create weak and easily guessable sequences. Many users are uneducated on how to choose reliable passwords, leading to an increased vulnerability to cyber-attacks. As a matter of fact, a dictionary of 250,000 words can be cracked in under five minutes on a Unix-based system using DES encryption [9] on a single laptop workstation. This demonstrates how passwords can be inadequate in many cases and highlight the need for other forms of authentication, explored in this report.

Furthermore, the need for passwords to be memorable often leads to people reusing the same password across multiple accounts, further increasing the security risk. In a survey of 3000 adults taken in 2019, 65% either reused the same password across multiple accounts or used one password for all accounts [22]. These issues led to the decision to research a more efficient method of authorization that is reliable, regardless of user error.

Biometrics are proposed as an alternative to passcodes and pins, using unique attributes of a person for improved security in authorization [21]. These present a potentially infallible method of identification, as these are attributes that are not easily altered or tampered with. Biometric data includes unique attributes of an individual, such as fingerprints, facial structure, and voice patterns. Biometric authentication systems work in two stages: first, a reference profile is created using devices such as face and fingerprint scanners or microphones, which capture biological data and create a digital representation. In the second stage, when the individual seeks verification, the system validates the presented biometric data against the existing profile [8].

Biometric systems are already being implemented in various contexts, including airports nationwide, where the Schengen Visa Information System (VIS) utilizes biometric data, primarily fingerprints, for identification purposes across EU countries [20]. These metrics are near impossible to imitate, showing the extent of its reliability, and justifying biometrics as a robust means of verification.

Keystroke dynamics is a behavioural biometric [25], first used in the second world war by military intelligence to identify operators via the rhythm of their morse code, known as *The Fist of The Sender*. Behavioural biometrics focus on features produced by a person, for example voice tune and typing pattern [1], as opposed to physiological biometrics, which include unique attributes of an individual, such as fingerprints and face recognition. They are also less intrusive data, making it a suitable choice for research as more participants are willing to share this information.

In this case the focus is on keystroke dynamics for verification. This assumes that each person's typing patterns differ, sufficiently enough for reliable validation. There is a multitude of literature researching the use of keystroke dynamics [2],[18],[10], showing that large samples of keystrokes per user, in combination with an appropriate classification algorithm, can be used for reliable authentication with error rates less than 5%. This led to the decision to base this report keystroke dynamics. The research involves recording large samples of inter-key times and comparing a few pattern recognition algorithms applied to those times, finding the most accurate.

To utilize keystroke dynamics in cyber security, there must be a reliable method of recording keystrokes, using the appropriate metrics to identify a genuine user vs an imposter. Keystrokes consist of two events: key-down and key-up [17]. The time between these two events is the intra-key time, the time which a key is held for. The time between a key-up and a subsequent key-down is the inter-key time, the time between keystrokes. As a sentence is typed, a sequence of these events is caused. These timings will differ from user to user and can be compiled to create a biometric profile with which to validate against. However, for this to be reliable, many samples must be collected per user, as typing patterns can vary even for one person.

Pattern Recognition

Pattern recognition is necessary for keystroke-based authentication. There is literature covering several methods using keystroke dynamics [11], most of which use a form of statistical classifier [12]. These tend to either use a simple, linear statistical comparison, or a machine learning model. This report focuses on implementing four, very different methods of classification, to have a range of models to compare against. If time allowed, more models could be implemented. Some of these use machine learning, requiring the data to be split into training and testing data.

Mean vector & standard deviation (MSD)

This is a simple method of classification, utilizing the mean vector and standard deviation of each inter-key feature. The training data split is used to calculate these values, before applying them to the test data.

Once the mean vector of each feature is calculated, standard deviation can be used as a measure of tolerance [2]. Each test feature is compared to the mean value for its corresponding feature in the training data. If the test feature is within a certain multiple of standard deviations from the mean, it is then accepted. Recursively applying this process to each feature creates a list of accepted vs non-accepted features, providing a total score for the test, which must be above an acceptance threshold for the sample to be authorized.

$$Score(s) = \sum_{i=0}^{n} \{f_i(s) \mid (|f_i(s) - f_i(\bar{x})| < f_i(m) f_i(\sigma))\}$$

In the equation above, f(s) is the features for each sample, and f(x) is the mean vector for each feature. Through testing, two values must be optimized for this algorithm to be effective: the standard deviation multiple, m, for each feature in the training set, and the threshold score for acceptance.

This method offers a simple form of classification, requiring minimal processing power. This enhances deployment capabilities, increasing suitability for devices such as phones and tablets. However, when applied on a larger scale, this approach can be disadvantageous. As the number of system users increases, relying solely on the feature mean and standard deviation will be insufficient. The mean vectors will differ by fractions of a second, resulting in an increase in false positives, presenting a significant security risk. Therefore, this method may be suitable only for small-scale applications, which is why it is a fit for this project, as there is a limited number of users, each with many samples. This will increase the accuracy of the mean, without overloading the algorithm with comparison profiles.

Multi-layer Perceptron (MLP)

A multi-layer perceptron is a type of neural network, using supervised machine learning on a set of labelled training data to build a model for classification or regression [14]. Neural networking is a field which uses simple interpretations of biological brains to solve complex computation problems. These consist of a multi-layered network of neurons, each taking inputs from the previous layer, producing an output dependant on learned weights. This proposes a more sophisticated technique for classification, as complex non-linear relationships between input and output data can be learned.

Multi-layer perceptron algorithms learn a function $f(\cdot): R^m \rightarrow R^o$ through training on a dataset, where m is the number of input dimensions and o is the number of output dimensions [29]. The training data set consists of a set of features and their labels, which is then used to produce the weights for each layer of neurons. The layers of neurons between the input and output are known as the hidden layers.

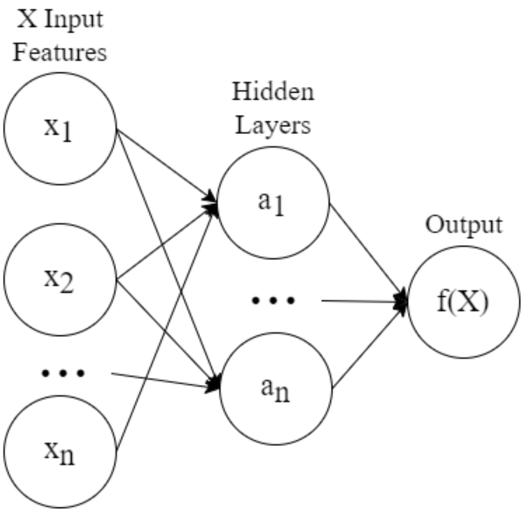


Fig. 1. MLP with one hidden layer

Once the MLP has been learned on the training data, it can be applied to a test sample for classification. At the input layer, each neuron represents a feature from the test sample {$x_{1,}$ x_2, x_3, .. x_i}. Each hidden layer consists of neurons {a_1, a_2, .. a_n} that transform inputs and output the result to the subsequent layer. The final output is the result of the transformed values from the last hidden layer.

The weights for each neuron are initialized randomly [15], then incrementally optimized using testing data. This uses a stochastic learning algorithm [16], making careful use of randomness within bounds to increase time efficiency when training a model. Through backpropagation, error is proportionally assigned to each weight, gradually training the model.

The capability to solve complex non-linear classification problems makes this method suitable for this project. Optimization is automated, and the large amounts of sample data collected are handled effectively. Only a few hyperparameters require tuning such as the number of hidden neurons, layers, and iterations [29]. Still, the use of random weight initialization means that different versions of the same model can produce varying validation accuracy.

K-Nearest Neighbours (KNN)

K-nearest neighbours (KNN) is a type of supervised machine learning algorithm, used for classification. It is trained on labelled data and classifies sample data points based on their proximity to the existing data [18]. It is used in a wide range of applications, such as speech recognition and text classification. Unlike MLPs, it presents a simple method of classification.

In KNN, data is represented on a d-dimensional plane, where d is the number of features per datapoint, and each record is represented as a vector of values that correspond to these features. The goal is to find the nearest datapoints to any given sample on this plane. The ‘k’ represents a hyperparameter [27], which determines how many neighbours are considered during class selection, which is carried out using a majority vote. The class with the largest number of votes among the k-nearest points is assigned, which can lead to a tie if k is even, in which case a random class is assigned. This means that k must be even to ensure accuracy. In the case of keystroke authentication, the dataset may contain more outliers due to variations in typing behaviour among different users. As a result, a higher k value is suitable.

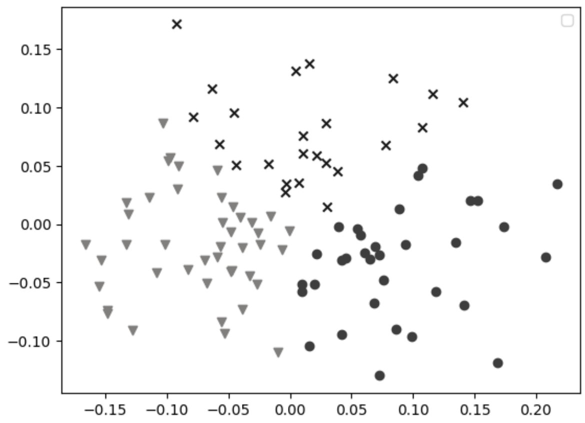


Fig. 2. 2-dimensional graph showing data representation of k-nearest-neighbours

To determine the distances, KNN calculates the Euclidean distance between the sample and each of the k-nearest datapoints, which is calculated by taking the square root of the sum of the squared differences between them. This is shown in the equation below, where x and y are the vector representations of the corresponding data points.

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

An advantage of using KNN for keystroke analysis is that it is a simple and easy-to-implement algorithm and can handle data with complex distributions. Additionally, KNN can be used for both binary and multi-class classification tasks. However, KNN also has some limitations. One such limitation is that KNN can be sensitive to the choice of the k value, which requires careful optimization. Another disadvantage is that KNN can be computationally expensive for large datasets, as it requires calculating the distance between each data point and the new data point.

Support Vector Machines (SVM)

Support vector machines use supervised learning to classify data, requiring a set of labelled training data to learn from. It makes use of mathematical functions to construct hyperplanes, which are decision boundaries that SVMs uses to separate data points into different classes. [4] describes the use of linear and polynomial functions to construct these planes. Generally speaking, a larger margin between a hyperplane and the nearest datapoints for a group, provides a lower generalization error [28]. Therefore, the algorithm aims to make these planes as flat as possible. This is for increased accuracy, as it maximises the distance between two data points on either side of the plane.

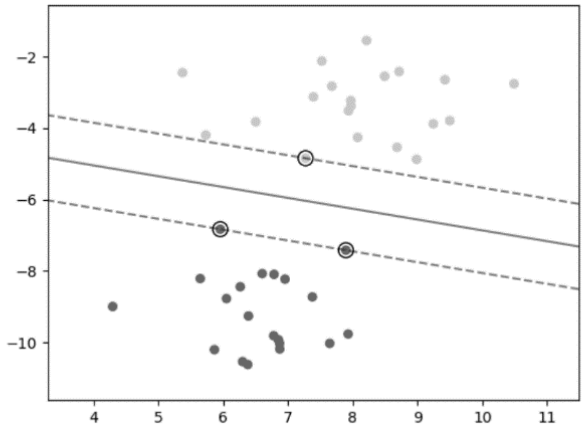


Fig. 3. 2-dimensional graph showing a single hyperplane for 2 sets of data [28]

The efficiency of the SVM algorithm is increased when the sample datapoints are spread further apart, making it easier for it to distinguish between classes. In the case of keystroke authentication, the performance is also influenced by other factors such as the quality of the data, the features extracted, and the specific tuning of the SVM algorithm.

There are several parameters that require tuning to improve the SVMs performance for this specific task. Firstly, the choice of kernel is important, as different kernel functions such as linear, polynomial, and radial basis function (RBF) all behave differently for varying forms of data [4]. This project uses the RBF model, as it tends to be more accurate for larger datasets, although it is more computationally expensive than linear models. The c-parameter controls the trade-off between maximizing the margin and minimizing the generalization error and can lead to overfitting the data if not optimized. Finally, the gamma-parameter controls the shape of the decision boundary, determining the area of influence of the support vector.

Objectives & Hypotheses

The primary objectives below serve as the focus of the project and are derived from the preliminary research and the project aim. Each primary objective has an associated hypothesis, predicting the outcomes of the research and providing a framework for testing the validity of the primary objectives. Additionally, they are supported by secondary objectives, which are necessary to prove or disprove the hypotheses presented.

Develop a reliable system for recording keystroke biometrics.

A reliable, easily accessible application must be developed, to collect a large dataset with numerous entries for each user. Some key characteristics of a reliable data system are:

- Data integrity: ensuring accuracy and consistency of the data entered into the system.
- Security: maintaining robust security measures on the database, preventing unauthorized access, modification, or deletion.
- Scalability: handling increasing amounts of data without compromising system performance or reliability.
- Usability: providing a system that is easy to use and navigate for both admins and end-users.

Collecting keystroke biometrics will require live tracking of user keystrokes for each sample entry. The keystroke biometrics in this case consist of a collection of inter-key times, represented in milliseconds between keystrokes.

Hypothesis - A reliable system making for recording a large sample of inter-key times for different users can be developed.

Secondary objectives

- Develop a front-end interface for data collection.
- Collect arrays of accurate inter-key times.
- Develop a secure back-end server for data storage.

Evaluate whether keystroke dynamics can be used to accurately authorize a user.

For a keystroke authentication system to be valid, the error rate must be below a certain threshold. This value should be determined by considering several factors. For example, ease of imitation. If the typing patterns of different users are too similar, the threshold for error must be lower. However, a lower value may lead to false rejections, where a legitimate user is denied access. On the other hand, a high threshold allows for more usability but may result on false positives, granting access to an imposter. A suitable value that produces the least false negatives and false positives must be calculated.

Hypothesis – An appropriate keystroke authentication algorithm can authorize users with a high level of accuracy, and an error rate below 5%.

Secondary objectives

- Implement and optimise various authorization algorithms.
- Compare accuracy and usability of authorization algorithms.
- Decide which algorithm is most appropriate for the application.

Requirements

This section outlines functional and non-function requirements, which expand upon the secondary objectives specified prior. There are many more detailed, technical requirements that are implemented in the application; however, these will not be covered here as a more technical deconstruction will be discussed in the implementation section below.

Functional Requirements

- The application must show the user information describing the nature of the research and the data usage.
- The application must require the user to fill in the consent form before entering data.
- The application must provide a means for the user to enter a username, password, and keystroke data.
- The application must create a new user if a user does not exist in the database.
- The application must not allow a user to enter data if the password is incorrect for the corresponding username entered.
- The application must display a sentence for the user to type.
- The application must prevent a user from entering incorrect data and inform them as such.
- The application must collect keystroke data as the user types.
- The application must inform the user of how many samples they have entered.
- The application must securely send each user's keystroke data to a remote server.
- The application must store samples for each user, in a relational database.
- The application must implement the authorization algorithm which has been determined to be most appropriate.
- The application must allow the user to test against the algorithm, informing them of whether they are accepted or denied.

Non-functional Requirements

- The algorithm should have an error rate below 5%.
- The application should be easily accessible and usable, able to be used without supervision.
- The recorded inter-key times should be accurate within 0.001 seconds.
- Security of the data stored on the server should be upheld, and the database should be regularly backed-up.
- The application should encourage users to enter many samples, for increased accuracy.
- The application should be always accessible, even during maintenance.

Design

This section details the experiment design and process for carrying out the investigation. It provides a step-by-step description of the procedures followed, in accordance with the primary and secondary objectives. It includes the approach, setting, and method used to select the participants, and explains the tools used for data collection and analysis.

Firstly, we discuss how participants are selected and approached. They must be from diverse background to ensure that results are representative of a wide range of individuals. This is because factors such as gender, age, education, and cultural background can influence typing patterns. Therefore, a variety of these is essential to ensure that the results can be generalized. They must also be of an age where they can understand how their data is being collected and used.

Participants should also have different levels of experience with computers, as more experienced people tend to have fast typing speeds. This would cause a strong bias in the results as data will be skewed towards this. To ensure diversity, participants are selected through family and friends with different levels of experience, who are in turn asked to share the experiment with their peers. Recruitment is carried out on social media, word of mouth, and direct invitations via email or phone to reach a wider audience.

All experiment procedures are carried out in accordance with the professional and ethical considerations previously discussed. It is conducted in compliance with ethical guidelines provided by the relevant professional associations and institutes. This includes obtaining informed consent, ensuring participant privacy and confidentiality, minimizing potential harm or discomfort towards the participants.

Each participant is first be presented with information explaining how their data will be collected, stored, and analysed in the experiment. This is to ensure full awareness of involvement and data usage. It also shows research approval from the appropriate institution, and provide contact details, so that they can email with any questions regarding the study. A consent form is required to be filled out before proceeding, confirming that all information has been understood, including the right to withdraw from the study at any point, without consequence.

After obtaining informed consent, participants are instructed to enter keystroke data, which is recorded accordingly. Clear instructions are provided, making sure that the task is understood and completed accurately. They are asked to complete a predetermined sentence a number of times, to collect large sample sizes for each participant.

Finally, participants' data is stored and analysed. The experiment can therefore be separated into two stages: data collection, and algorithms analysis. The data collected from the participants is stored securely, guaranteeing confidentiality, and personal data is de-identified, ensuring privacy. The results of the study are reported in a way which upholds anonymity of the participants.

Constructing the sentence

The sentence chosen to collect participants' keystroke data is:

"the quick brown fox jumps over the lazy dog"

This is a commonly used pangram, containing every letter in the alphabet at least once. This is important as it means that typing data can be captured across each letter on the keyboard. Non-alphabetical characters were not included in this sentence, as most keyboards require a shift key to be pressed to type special characters such as "@" or "&". It also does not include any capital letters, as this would require either a shift or caps lock key to be pressed, greatly affecting the inter-key times.

Additionally, keyboard configurations can vary a lot, such as containing number-pads, keyboard shortcuts, macros, and different languages. As this experiment is conducted on each participants' own workstation, this sentence maximises the similarities between their keyboard interfaces. Participants are also instructed during recruitment to carry out the experiment on a laptop as opposed to a mobile device, further maintaining consistency.

The length of the sentence is also important for several reasons. Participants are instructed to type the sentence 60 times or more, requiring it to be short and easily readable, so that mistakes can be minimized. The use of a large paragraph of text was proposed; however, this would lead to an increase in errors and would contain punctuation such as full-stops, which are avoided for the previously mentioned reasons regarding alphabetical characters. The sentence consists of short, simple words for this same reason, also taking into consideration participants with dyslexia.

The number of samples per participant is determined considering the sentence length, and total time for the experiment procedures. The decision to instruct the participants to enter the sentence 60 times or more was reached through preliminary testing and peer review. The experiment was tested to find a comfortable length of time that participants would be willing to dedicate. It takes the average participant around 10 minutes to enter the sentence 60 times, and they are encouraged not to complete this in one sitting.

Recording the data

Participant entries are captured via a text field with certain constraints. In order to maintain consistency between user samples, the participants are forced to only enter the correct characters. This means that errors such as backspaces and punctuation are not directly recorded, and users are informed when an incorrect character has been entered. These errors are instead interpreted as a larger inter-key time for that specific character pair, increasing until the correct character is entered.

Furthermore, the algorithms being implemented in the analysis stage are trained on data that requires the same number of characters across samples, which would differ in length if every error was recorded. In turn, this would reduce the effectiveness of these algorithms.

This investigation uses a website to carry out the experiment, as it is easily distributable through sharing a link. It can also be carried out without supervision allowing for a larger dataset to be collected, because individuals can participate regardless of their location. A website is used as opposed to an application, as it would require installation on each system, and compatibility across a range of devices, which is not feasible within the time constraints.

Lastly, a separate frontend website and backend server are necessary, separating data recording and data storage. This allows for a more reliable and secure system because it minimizes the risk of data corruption and data loss, as the frontend focuses on user interaction and data collection, while the backend focuses on data processing and storage.

Frontend

The frontend application is presented as a single page website, consisting of four distinct sections: participant information, consent form, experiment instructions, and data entry fields. This section of the report elaborates on the data entry fields, and how they collect keystroke data.

To maintain authenticity of user data, fields are provided for a username and password, in addition to the keystroke entry field. This prevents participants from entering false data for another user, which would skew the training data for the analysation algorithms. The keystroke entry field has several functions attached to it. These are for recording inter-key times, restricting character entry, and auto-submitting samples, and are executed on each keypress.

Inter-key times are recorded using a timer that resets after every sample entry. The timer is started once the first character of the sentence is typed, and a timestamp for every subsequent character is recorded in a list. The inter-key times are then computed as the difference between timestamps for every character pair, in milliseconds, to ensure the utmost accuracy.

$$\text{Inter-key-times} = \{k \mid k = timestamp(char_i) - timestamp(char_{i-1})\}$$

Character entry is restricted by recursively comparing the input text to the prepared sentence. On every iteration, a string consisting of all previous characters including the current character is compiled. This string is then compared to the prepared sentence, and if it does not match, the character is not accepted, and the user is informed of such. This function only allows characters to be typed when it matches the sentence, preventing errors and false data entry.

Once an entry is completed, a function sends a submission request to the backend, to store the user sample in the database. It is executed once the length of the character string matches the length of the prepared sentence, indicating that all keystrokes have been recorded. The request contains the username, password, and inter-key times for each sample, and waits for a response from the backend regarding the validity of the username and password. Once confirmation or rejection information is received, the user is informed of the result, and the text field is cleared to allow for the next entry.

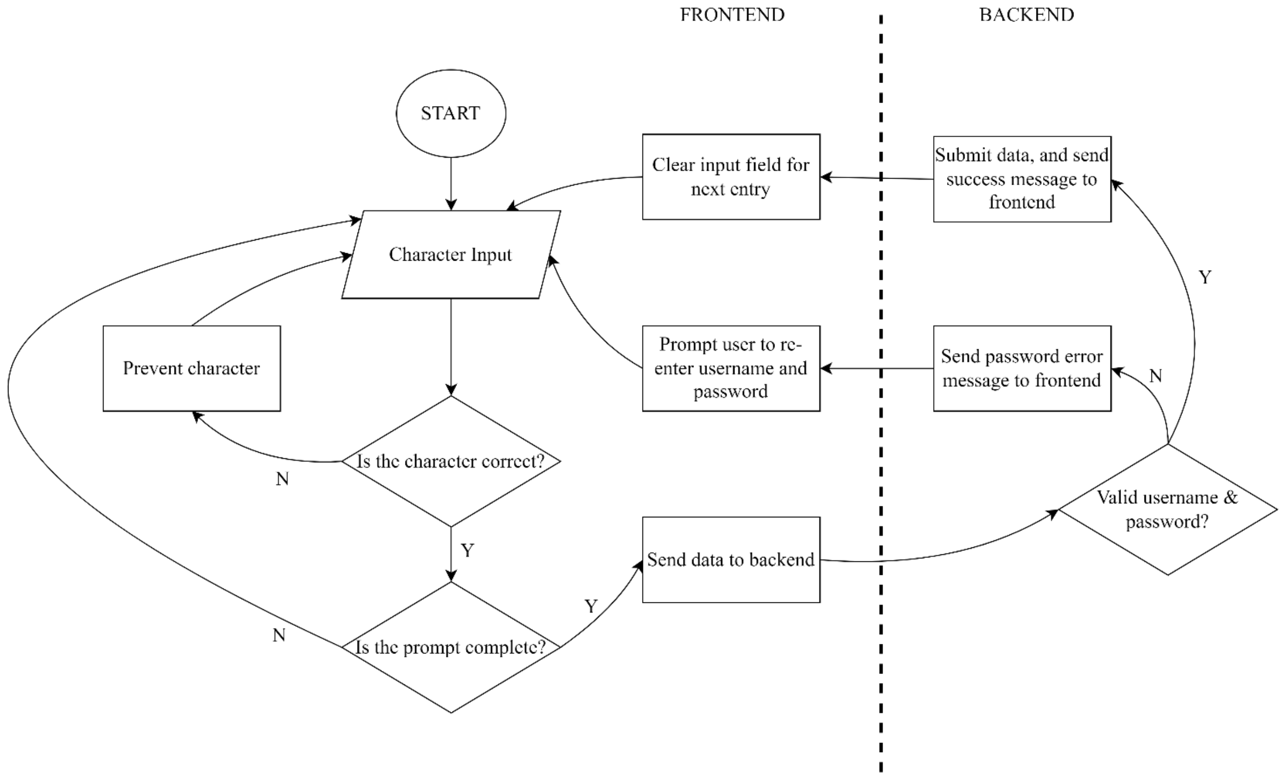

FRONTEND
BACKEND
START
Clear input field for next entry
Submit data, and send success message to frontend
Character Input
Y
Prevent character
Prompt user to re-enter username and password
Send password error message to frontend
N
Is the character correct?
N
Valid username & password?
Y
Send data to backend
Y
N
Is the prompt complete?

Fig. 15. Flowchart showing frontend logic and communication with the backend.

Backend

The backend architecture is comprised of a database and an application that handles submission requests received from the frontend. It presents a simple interface which the frontend can submit data through, accepting an array containing the username, password, and inter-key times of each sample. Once a request is received, the application carries out a series of steps to handle it and respond appropriately.

The first step involves checking whether the user already exists in the database. If it exists and the password does not match, an error message is returned to the frontend and the sample is not stored. If the user does not exist, a new user is created, and the sample times are stored. This approach eliminates the need for a separate registration and login system, streamlining the data collection process for a large number of participants.

If the user exists in the database and the password matches, the sample is stored and assigned to that user. The database uses a many-to-one relation, with one table for usernames and passwords, and another for samples. The user id serves as a foreign key in the samples table, linking the two together. This structure is used for data storage as it simplifies the process of retrieving data for a particular user through database queries. Additionally, it increases security since the analysing algorithms only have access to the samples table therefore being incapable of retrieving user passwords from the user table.

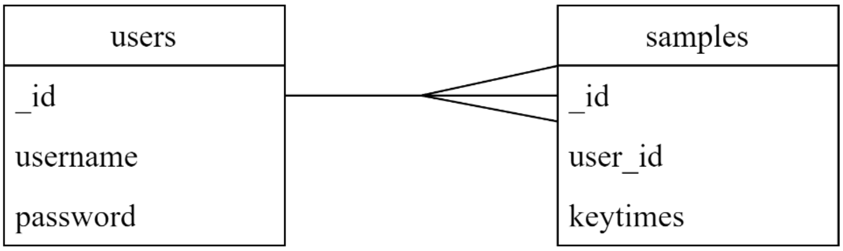


Fig. 14. Many-to-one relation between user and samples tables

The backend server is deployed separately from the frontend and is protected by a firewall that prevents attacks from any other locations without a key file. This increases security by reducing the area for potential attacks as it can only be accessed by authorized individuals or systems that possess the necessary credentials.

Algorithms analyses

At the end of the experiment, all collected data is subjected to a series of machine learning algorithms designed to detect and authenticate the unique typing patterns of each user. These algorithms consider various metrics, such as typing speed and average times per keystroke.

An in-depth statistical analysis of each algorithm is carried out using a combination of graphs, matrices, and tables. This provides detailed information about the performance of each, evaluating their performance and efficiency.

The results section of this report details the plan and execution of this analysis, determining their strengths and weaknesses. It also presents recommendations for improving the authentication process, based on the accuracy, and practicality of each.

Implementation

The focus here is on the practical implementation of specific design features and satisfying requirements, presented as a series of annotated screenshots and code snippets. The website is designed to provide a user-friendly interface, being kept minimal and intuitive for ease of use for participants. The backend is designed to be fast and responsive, handling scaling for concurrent data entries effectively. The implementation of the machine learning algorithms involved training them on the datasets of inter-key data and testing them recursively for each user.

Technologies

Various technologies and tools have been used for the website, chosen based on their suitability and popularity for similar tasks. The frontend is constructed using HTML, CSS, and JavaScript, as they are the obvious choices for a simple website design.

The backend application is constructed using Fastie [26], a web framework for building APIs with Python. It was selected because of its simplicity and performance. It is designed to be easy to use and develop, with high scalability, which is key for this project because it needs to handle large amounts of user data from different clients being received simultaneously. It also supports asynchronous programming, making it suitable for this project. FastAPI also provides automatic validation of input data and simple integration of SQL databases, easing the processing and storage of sample data received.

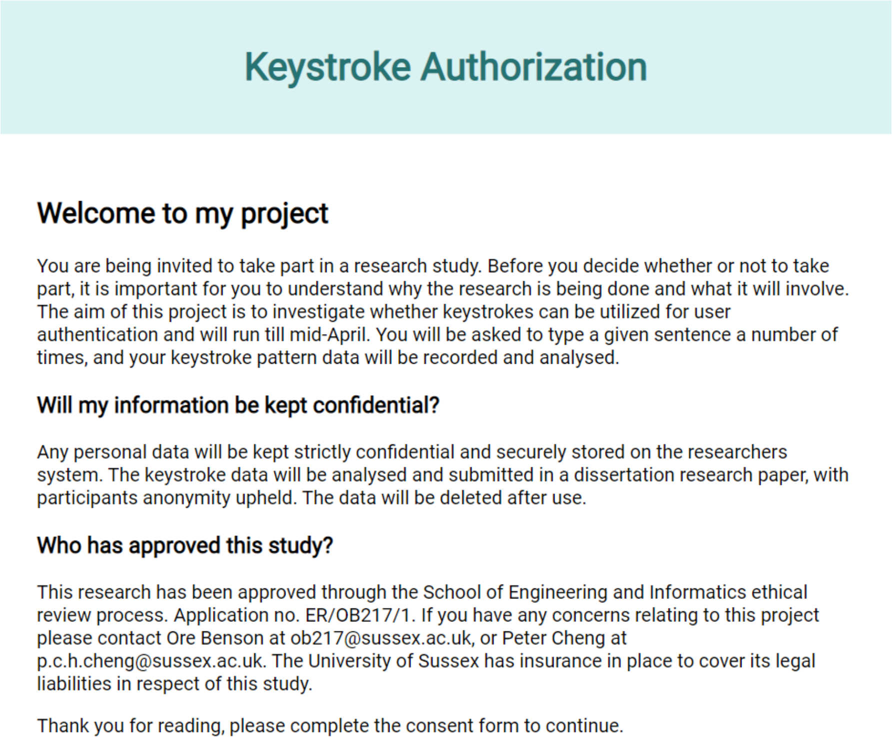
The database used in tandem with the FastAPI framework is SQLite. It is chosen as it requires no setup or administration, and can be directly constructed using schemas, coded in Python. It supports standard SQL syntax, which is the most common language used for relational database management, meaning there is a wide range of support, and is appropriate for this project. SQLite is self-contained and serverless, and its transactions are atomic, consistent, isolated, and durable (ACID). This is important as it guarantees data integrity and reliability, in accordance with the objectives.

The backend is deployed on Amazon Web Services (AWS), a secure, and reliable cloud system. It has built in security features, which are used to prevent unauthorized access to the backend database. Furthermore, it runs on a Unix-based operating system, supporting nginx and python for server management and operation.

The analysis uses the scikit-learn library [27]-[29]. a Python library for statistical inference and hypotheses testing, capable of analysing large datasets. It provides many statistical functions, including the classification functions used in this project. It suits this project as it is compatible with Matplotlib [23], a library for creating data visualisations in Python.

Frontend

Displayed below are screenshots from the website, showing the implementation of the design features. The emphasis is on clarity and usability, making the information and instructions easy to understand and carry out. Also shown is an example of one of the functions in the JavaScript code, the rest of which are in the attached source code for the site.



Keystroke Authorization

Welcome to my project

You are being invited to take part in a research study. Before you decide whether or not to take part, it is important for you to understand why the research is being done and what it will involve. The aim of this project is to investigate whether keystrokes can be utilized for user authentication and will run till mid-April. You will be asked to type a given sentence a number of times, and your keystroke pattern data will be recorded and analysed.

Will my information be kept confidential?

Any personal data will be kept strictly confidential and securely stored on the researchers system. The keystroke data will be analysed and submitted in a dissertation research paper, with participants anonymity upheld. The data will be deleted after use.

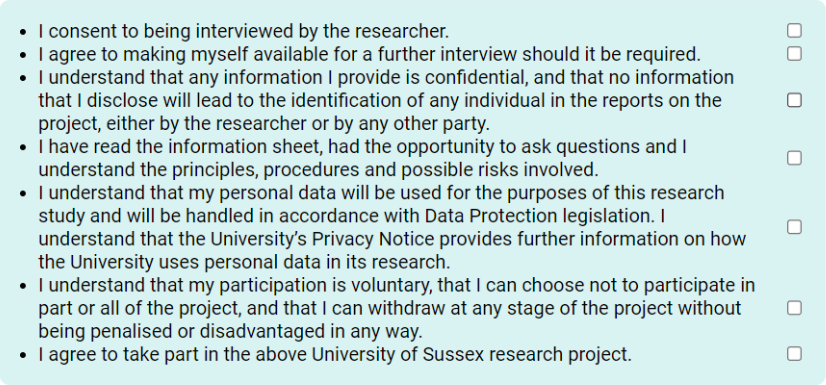
Who has approved this study?

This research has been approved through the School of Engineering and Informatics ethical review process. Application no. ER/OB217/1. If you have any concerns relating to this project please contact Ore Benson at ob217@sussex.ac.uk, or Peter Cheng at p.c.h.cheng@sussex.ac.uk. The University of Sussex has insurance in place to cover its legal liabilities in respect of this study.

Thank you for reading, please complete the consent form to continue.

Fig. 4. Participant information

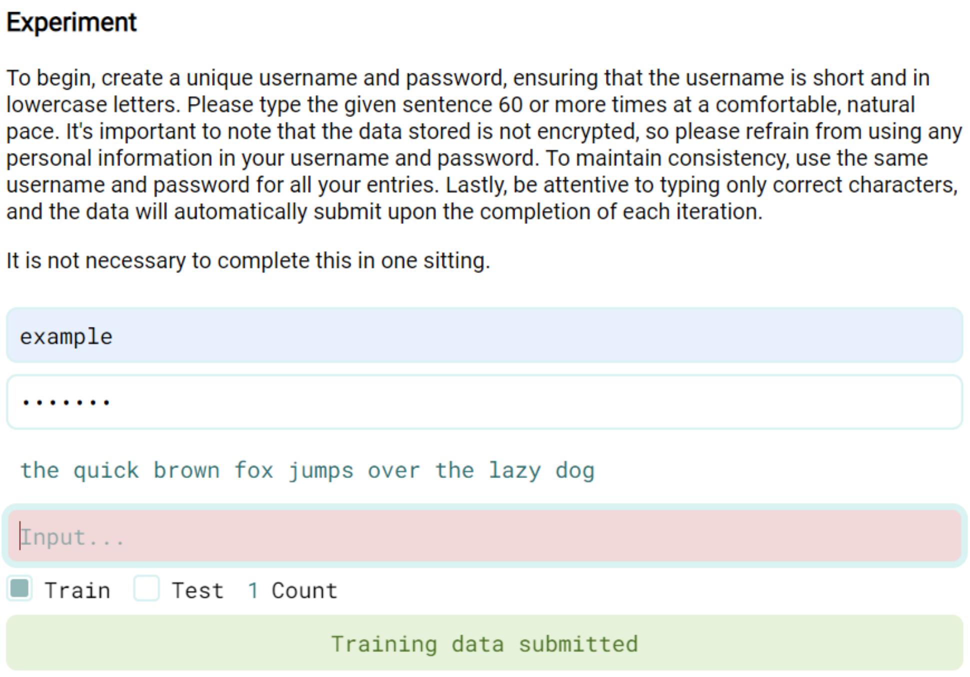
The screenshot shown in fig. 4. is the first section of the website, welcoming and briefing participants on the purpose of the study and providing information on contact details and approval of the study from the University of Sussex. Large, easily readable font is used, with simple grammar, accommodating for a range of participants. The information is separated into the relevant sections, so that it is clear and concise.



- I consent to being interviewed by the researcher. □
- I agree to making myself available for a further interview should it be required. □
- I understand that any information I provide is confidential, and that no information that I disclose will lead to the identification of any individual in the reports on the project, either by the researcher or by any other party. □
- I have read the information sheet, had the opportunity to ask questions and I understand the principles, procedures and possible risks involved. □
- I understand that my personal data will be used for the purposes of this research study and will be handled in accordance with Data Protection legislation. I understand that the University's Privacy Notice provides further information on how the University uses personal data in its research. □
- I understand that my participation is voluntary, that I can choose not to participate in part or all of the project, and that I can withdraw at any stage of the project without being penalised or disadvantaged in any way. □
- I agree to take part in the above University of Sussex research project. □

Fig. 5. Consent form

The consent form section shown in fig. 5. has multiple points which must all be accepted for the participant to proceed with the experiment. They confirm that the participant has read and understood the information, and the rights that they have in relation to their data and participation. It separates these into bullet points, for simplicity and easy comprehension, and uses checkboxes to accept each point as opposed to a signature, so that they can be accepted on a laptop or PC.



Experiment

To begin, create a unique username and password, ensuring that the username is short and in lowercase letters. Please type the given sentence 60 or more times at a comfortable, natural pace. It's important to note that the data stored is not encrypted, so please refrain from using any personal information in your username and password. To maintain consistency, use the same username and password for all your entries. Lastly, be attentive to typing only correct characters, and the data will automatically submit upon the completion of each iteration.

It is not necessary to complete this in one sitting.

example

•••••••

the quick brown fox jumps over the lazy dog

Input...

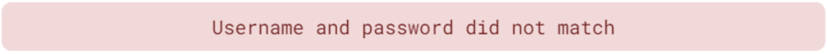
☑ Train □ Test 1 Count

Training data submitted

Fig. 6. Experiment instructions and data entry fields

The data entry section is displayed in fig. 6. First, it instructs the participant on how to carry out the experiment, showing the points discussed in the design, such as the number of samples to enter, and accurate character entry. It recommends that all the samples are not entered in one sitting, as this may take longer for less experienced participants, and reduces possible bias caused by a user learning the prompt thereby increasing in typing speed.

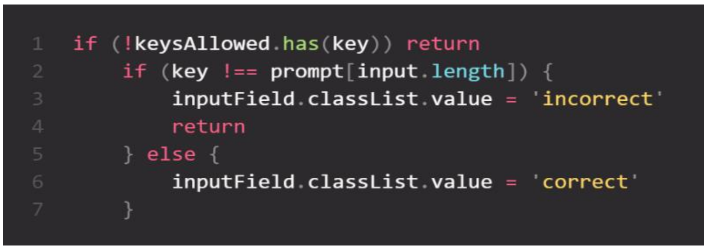
Fields such as the username, password, and prompt, use a monospace version of the font, making the prompt easier to read and copy. The red highlight of the input section demonstrates how the user is informed that an incorrect character is currently being typed, changing to green when characters are accurate, and a count is shown to keep track of the number of entries for this session. Data entry is prevented if the consent form has not been filled, and if a username and password have not been entered, by blocking access to the input field.



Username and password did not match

Fig. 7. Password verification

The area underneath the input field, shown at the bottom of fig. 6., and in fig. 7., is used to display information received as a response from the backend, such as submission of training data vs testing authentication, and user verification. The highlight used is similar to the input field, using red for error messages, and green for success messages, further increasing the intuitivity of the system, making data entry a simple process for participants.



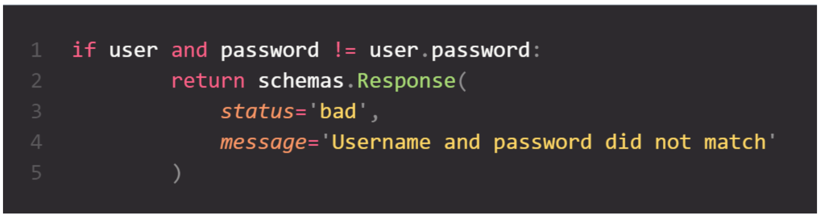
```
if (!keysAllowed.has(key)) return
    if (key !== prompt[input.length]) {
        inputField.classList.value = 'incorrect'
        return
    } else {
        inputField.classList.value = 'correct'
    }
```

Fig. 8. Character comparison function

The code snippet in fig. 8. shows the string comparison function that determines whether the current character being typed is correct and sets the value for the CSS class of the input field, determining the colour. It is executed recursively, comparing the entry to the prompt with each keypress.

Backend

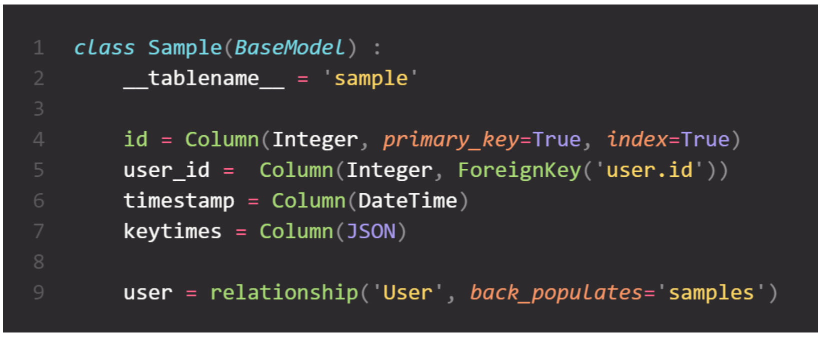
Displayed below are screenshots from the backend implementation of the design features such as the database initialization, and the deployment. The aim of the backend development was to maintain a high level of security, and a high speed of interaction with the frontend.



```python
if user and password != user.password:
        return schemas.Response(
            status='bad',
            message='Username and password did not match'
        )
```

Fig. 12. User verification response handling

Fig. 12. Is an example of the frontend response handling. It shows the case in which the user exists, and the password doesn't match, returning an status, and appropriate error message, to inform the participant.



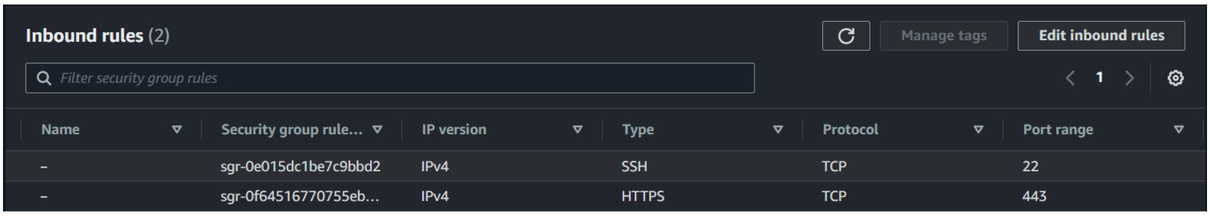
```python
class Sample(BaseModel) :
    __tablename__ = 'sample'

    id = Column(Integer, primary_key=True, index=True)
    user_id =  Column(Integer, ForeignKey('user.id'))
    timestamp = Column(DateTime)
    keytimes = Column(JSON)

    user = relationship('User', back_populates='samples')
```

Fig. 9. Sample table initialization

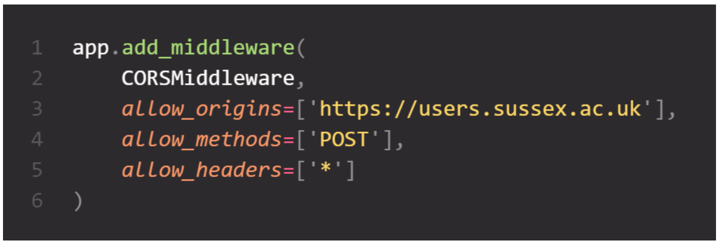
The sample table of the database consists of the sample id, user id, timestamp, and inter-key times array, as specified in the design. The user id is a foreign key and is what the algorithms see when the data is pulled, maintaining user anonymity.



Name	Security group rule...	IP version	Type	Protocol	Port range
–	sgr-0e015dc1be7c9bbd2	IPv4	SSH	TCP	22
–	sgr-0f64516770755eb...	IPv4	HTTPS	TCP	443

Fig. 10. AWS security group

Fig. 10. shows the security groups for accessing the server. It only allows two types of connections, SSH, which requires a specific key file, and HTTPS, using a certificate verified by the Let's Encrypt [24] certificate authority.



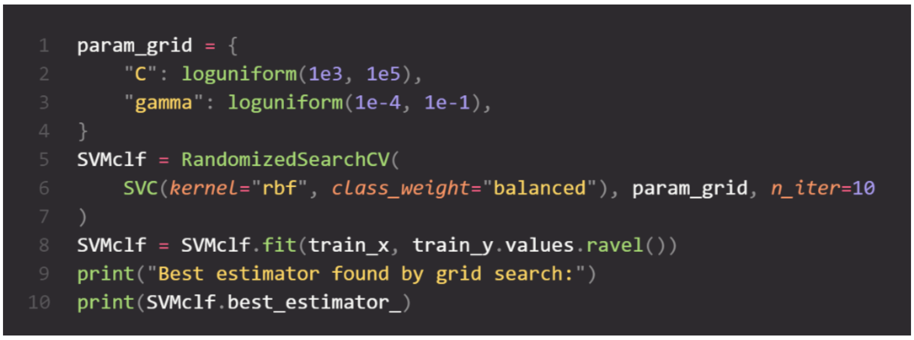
```
app.add_middleware(
    CORSMiddleware,
    allow_origins=['https://users.sussex.ac.uk'],
    allow_methods=['POST'],
    allow_headers=['*']
)
```

Fig. 11. CORS settings for database access

The snippet in fig. 11. demonstrates the use of Cross-Origin Resource Sharing (CORS) to only allow access from the University of Sussex domain. In combination with the security groups in fig. 10., this ensures that a high level of security is upheld, protecting potentially sensitive participant data.

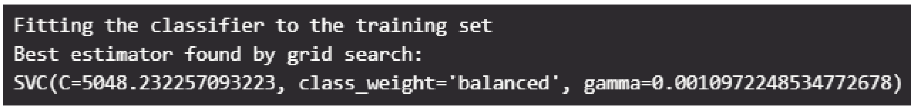
Analysis

The code snippet below shows an example of the initialization and optimization of one of the classifier models being used in this report.



```
param_grid = {
    "C": loguniform(1e3, 1e5),
    "gamma": loguniform(1e-4, 1e-1),
}
SVMclf = RandomizedSearchCV(
    SVC(kernel="rbf", class_weight="balanced"), param_grid, n_iter=10
)
SVMclf = SVMclf.fit(train_x, train_y.values.ravel())
print("Best estimator found by grid search:")
print(SVMclf.best_estimator_)
```

Fig. 12. Optimizing SVM classifier



```
Fitting the classifier to the training set
Best estimator found by grid search:
SVC(C=5048.232257093223, class_weight='balanced', gamma=0.0010972248534772678)
```

Fig. 13. SVM values

Fig.12. shows an example of classifier optimization. In this case the SVM based classifier is being optimized, by finding the optimal C, and gamma values. The output is displayed in fig. 13. This form of optimization is applied to each classifier.

Testing

To test the system during development, the website was run locally, and used by a small group of participants. They were asked to enter their data 50 times and were timed to determine the time taken to complete the experiment. After each testing phase, the users were asked about the feel of the website, describing how easy it was to understand the information presented and follow the instructions. Their feedback was used to make changes to the user interface and improve the user experience.

The backend application was also hosted locally during testing, to demonstrate how it received and handled data from the frontend. This testing stage was as important step in ensuring the reliability and usability of the system before deployment.

Experiment

The execution of the experiment in agreement with the design procedures, is explained in this section. The experiment was carried out over a period of two weeks, and participation requests containing the website link were distributed through various channels, such as WhatsApp, Instagram, and email. Participants were also asked to share the link with their peers to collect a wider range of data. The invitation link provided participants with a comprehensive explanation of the experiment, as shown in the fig.4. Data collected during the experiment was anonymized to ensure participant privacy, complying with all relevant data protection regulations, discussed prior.

To avoid potential bias in the data, care was taken not to recruit a significant number of participants from the researcher's Computer Science course. This decision assumed that these individuals would have more experience using computers, leading to a skew in typing speeds.

Participants were instructed to use a laptop for the experiment to ensure that the data collected was consistent with the design specifications. This was done to minimize the potential for inaccuracy in the data due to the use of different devices and maintain consistency.

The backend server was monitored regularly throughout the experiment, to make sure that it was functioning smoothly. There were a number of instances where the server had to be restarted, however the database was regularly backed up and kept secure, to prevent data loss in the event of a server crash.

Results

Analysis plan

In order to compare the algorithms outlined previously, they each need to be implemented and trained on the recorded data. This is carried out using the scikit library [27]-[29], in combination with matplotlib to display the results.

Firstly, the raw keystroke data must be obtained from the experiment must be pre-processed, filtering out noise and outliers that would affect the performance of the algorithms. Then, feature extraction is performed to obtain the relevant values from the data, and the features are scaled within -1 and +1 for the classifiers. The resulting data is split into a training set, and a testing set, with a 3:1 split, to prevent data being classified on itself, which would lead to a skewed accuracy. Each algorithm must be optimized on this training data, finding values for the hyperparameters that allow each to perform to its maximum potential.

For this study, four algorithms were chosen to find the most accurate method of authentication via keystroke biometrics. The pattern recognition section of this report contains in-depth explanations of these. The first algorithm is the Multi-layer Perceptron (MLP), a neural network algorithm, chosen because of its ability to compute complex, non-linear relationships between input features, without a high computational cost.

The second algorithm is K-nearest neighbours (KNN), a machine learning algorithm that uses vector distance to classify sample data and is suitable because of its efficiency. However, it is less complex than MLP and SVM algorithms.

The third algorithm is the Support Vector Machine (SVM) algorithm, which also uses machine learning. It is a unique algorithm and is popular for similar classification tasks.

The final algorithm uses mean & standard deviation (MSD), and is a simplistic, statistics-based algorithm. It accepts a sample based on the number of inter-key times that are within a standard deviation of the mean for each character pair.

To help visualise the performance of each algorithm, a score matrix heatmap will be used, showing the number of acceptances of each sample compared to each user, making it clear to see trends of acceptance rates for each. The score for each is the number of samples predicted to be of that class.

The metrics used for comparing each algorithm are the accuracy, precision, recall, and F1 score, with the formulas for each shown below:

$$\text{Accuracy} = \frac{(\text{True Positive} + \text{True Negative})}{(\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative})}$$

$$\text{Precision} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Positive})}$$

$$\text{Recall} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Negative})}$$

$$\text{F1 score} = \frac{(2 * \text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

To decide on the best algorithm for this study, runtime, and accuracy for each must be compared. The preferred algorithm should have a high accuracy and low runtime, for authentication error, and minimal computational cost.

Summary

Over the course of the experiment, a large dataset was formed, with users entering over 30,000 keystrokes in total. This is a satisfactory amount of data for the stats analysis, with an average of over 1800 keystrokes per user.

Total users	18
Total samples	790
Total keystrokes	33970

Fig. 19. Summary of data

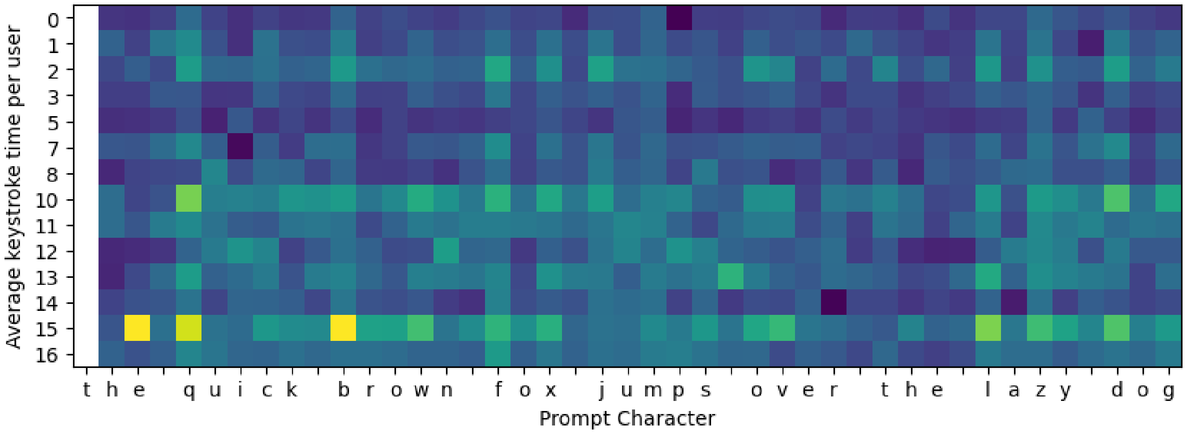


Fig. 16. Graph showing heatmap of average inter-key times per character.

Fig. 16. Is a heatmap of average inter-key times per user, with the higher values being a lighter shade of yellow, and the lower values being a darker shade of blue. The reason for user 4, 6, and 9 not being shown I because they were removed in pre-processing, which was necessary as these users either didn’t enter enough data, or their data was entered incorrectly during the experiment. From this graph, it can be deducted that certain participants, such as no.15 and no.10, are slower at typing, on average, than participants such as no.0 and no.7, as they have a number of brighter yellow squares. In the case

of no.15 especially, there are solid yellow boxes on the first ‘e’, ‘q’, and ‘b’ characters. This either means that they struggled repeatedly on these in particular or made a large enough error on one sample to significantly skew their average.

Taking a closer look at certain characters reveals trends across all the participants, showing bright vertical lines above characters at the start of words like ‘q’, ‘f’, and ‘l’ tend to take more time to enter. Other characters such as ‘x’, and ‘z’ also show this characteristic, which is most likely because of how uncommon they are to type, and their location at the bottom of the keyboard. The ‘t’ at the start of the graph has no entries, as there is no character entered before it, so it starts at 0.

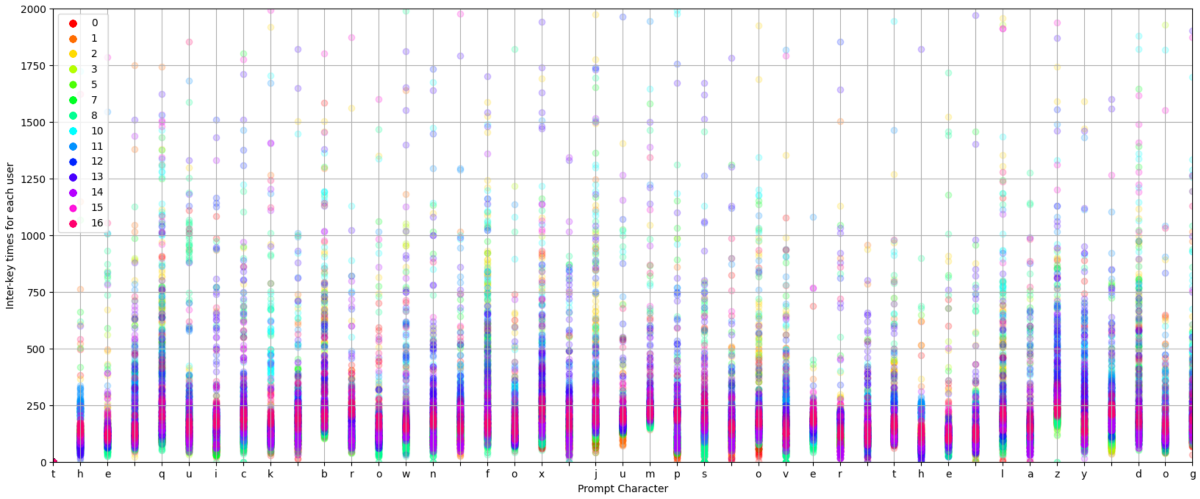


Fig. 24. Graph showing all keystrokes.

Fig.24 shows a graph of every keystroke from every user against the characters in the prompt sentence. These are represented as transparent, coloured points with each user being assigned a different colour. It shows that in general, users had inter-key times below 500 milliseconds, skewed toward the 250-millisecond mark. This implies that users had relatively similar keystrokes, which affects the MSD algorithm’s performance as it depends simply on the mean and standard deviation.

It also shows the fastest speeds for each character, which keys such as ‘b’, ‘u’, ‘m’ and ‘z’ having the slowest speeds, whereas keys such as the ‘p’ in ‘jumps’ and the ‘r’ in ‘over’ having minimum values almost equal to zero. This is interesting as the ‘r’ also has one of the narrowest spreads of datapoints, which is caused by its proximity on the keyboard to the previous character ‘e’. The keystrokes with the largest spreads tend to be the first letters of each sentence.

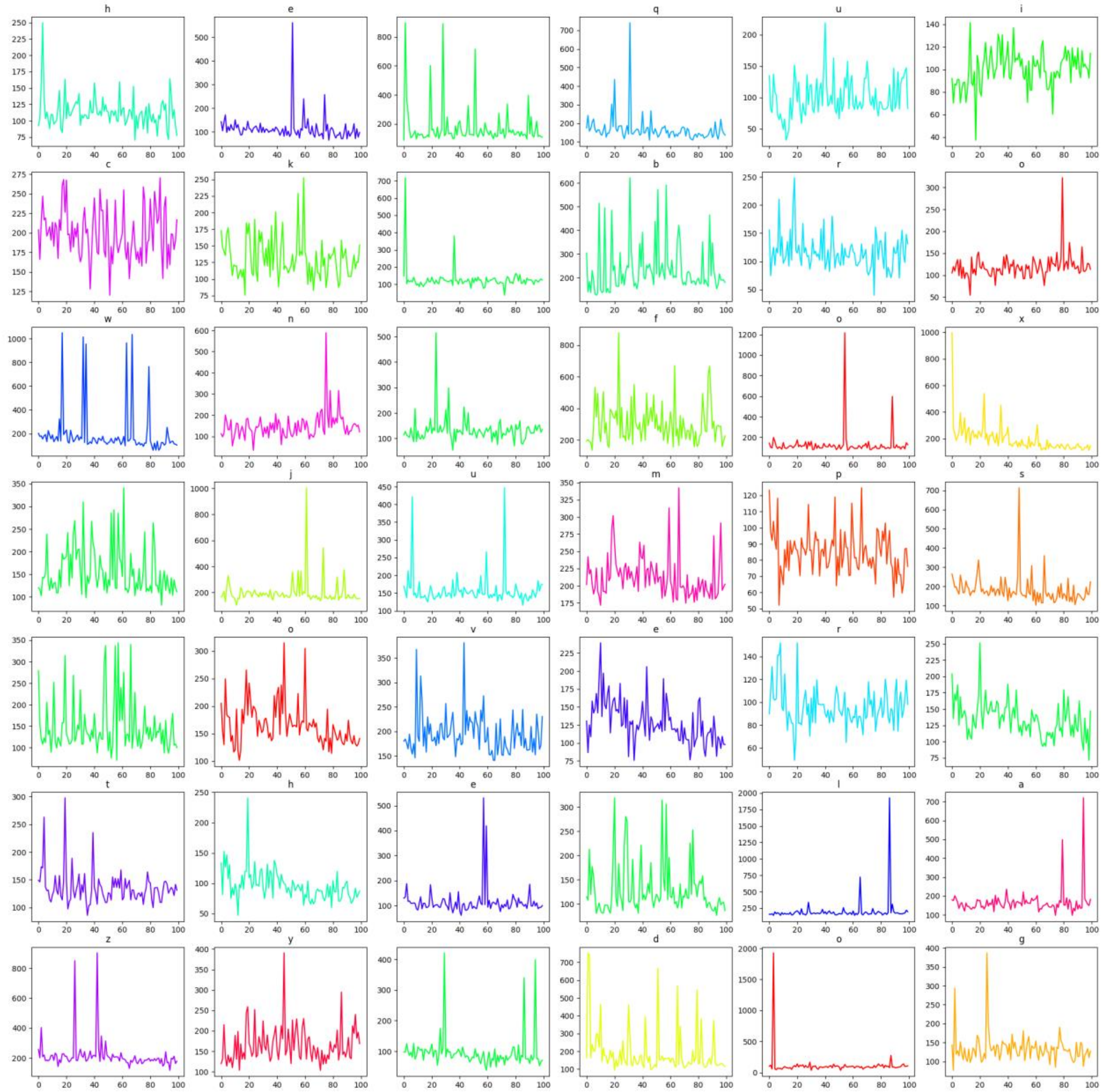


Fig .17. Graph showing inter-key times over number of entries from one user, for each character.

A user with a large number of entries was selected, and their keystrokes were further analysed here using the series of graphs shown above. Each graph shows the change in keystroke times for a given letter, over the number of entries entered. This gives a look into whether typing the same words repeatedly affects learning and typing speed.

Most letters show a consistent speed, with occasional spikes and dips. The most consistent of these is the final ‘o’, with only one major spike near the start. This is similar to the behaviour of the ‘o’ in ‘fox’ as well, being the fastest typed letter among all of them. Surprisingly, there does not seem to be any upwards trend in any of the data, meaning that learning the sentence did not have an affect on typing speed for this user.

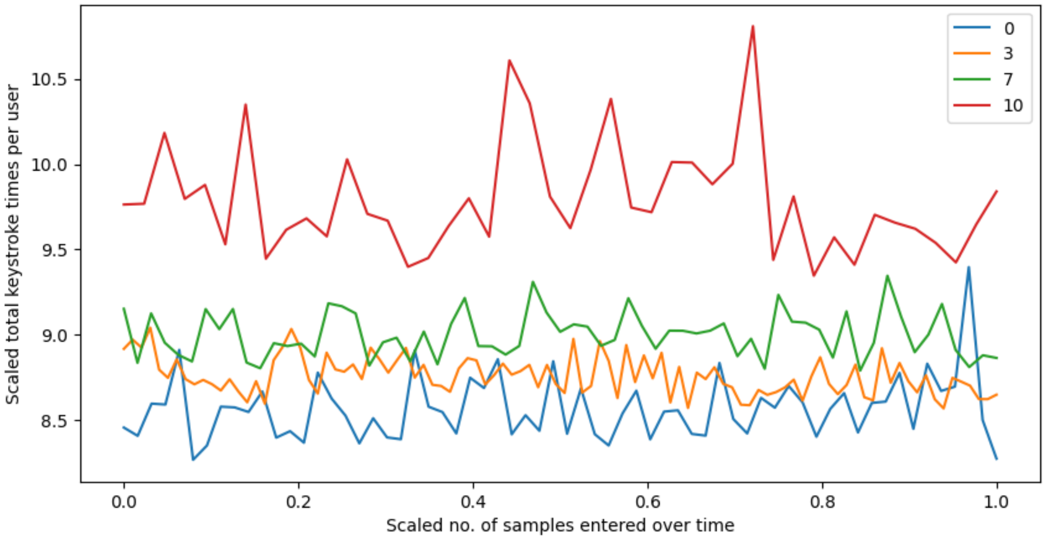


Fig. 13. Graph showing average key times for select users.

Four users were chosen at random to compare data in a similar way to the previous graph, represented instead as the total time taken to type the sentence with each repetition. It further demonstrates that repeating the sentence does not necessarily improve typing speed, as the data for all users stays relatively consistent. This graph in fig.13. clearly shows the difference in typing patterns among these users however, implying suitability for use in a keystroke authentication as they are unique, with participant no.10 being slower than participants 0, 3, and 7.

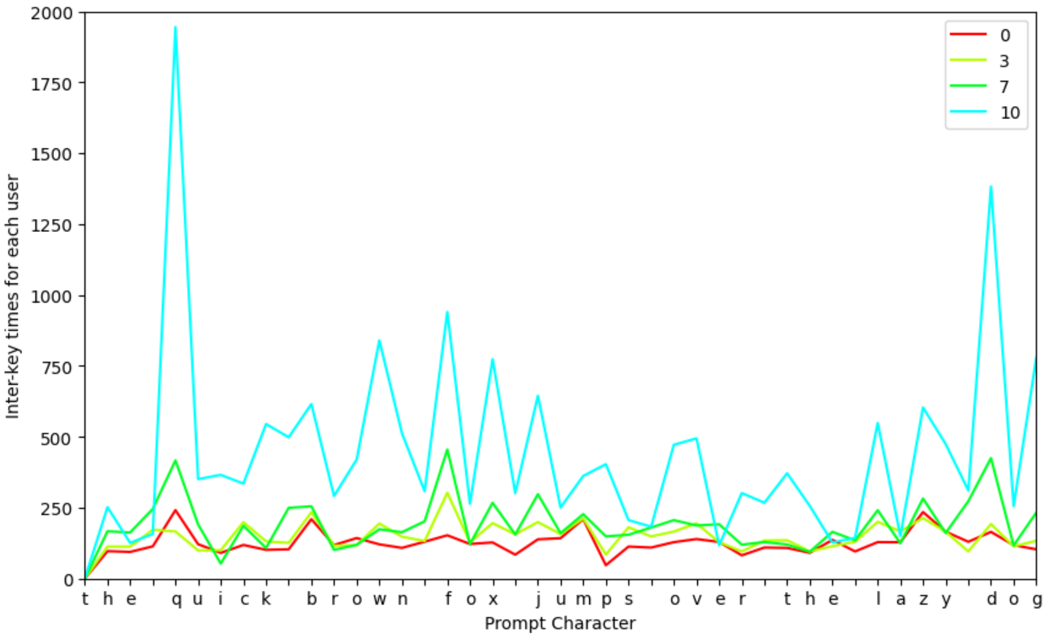


Fig. 31. Graph showing average key times for each character for select users.

Fig. 31. Shows these same four users, compared instead by their average inter-key times for each character in the prompt. This shows a clear distinction between no.10 and the other participants, showing large spikes in keystroke times, especially on ‘q’. Participants 0,3 and 7 seem to have very similar patterns, but with a closer look a few key differences are clear. Participant no.7. has more spikes than the others, albeit on a smaller scale than participant no.10. Also, no. 0. Has lower dips than the others, showing the fastest typing speed out of this selection.

Score matrices

The score matrixes below were computed by running each classifier on every user pair and returning the number of true and false predictions for each. The brighter shades represent a higher number of positive predictions, whereas the darker shades represent negative predictions. They are each followed by a brief summary and will be compared in the evaluation below.

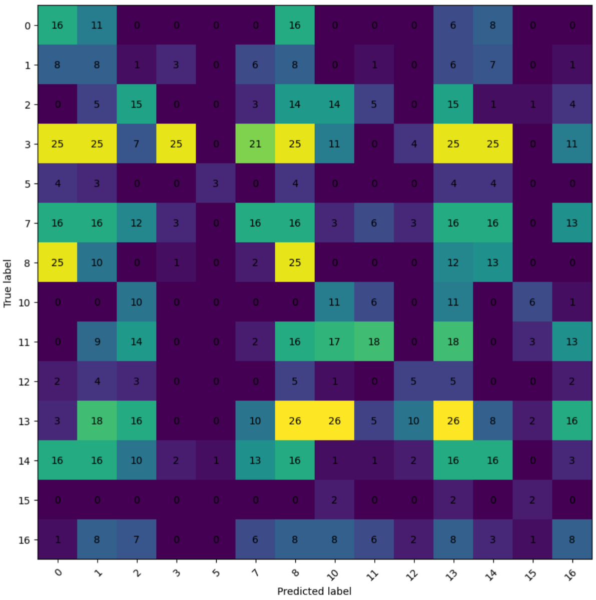


Fig. 19. MSD score matrix

The score matrix in fig.19. was produced by the MSD algorithm. It clearly shows a wide spread of predictions, which a vague diagonal trend, which was further revealed when tested with a lower acceptance threshold. However, this also led to many more false negatives, decreasing the accuracy. Certain comparisons seem to be predicted more, such as no.3 and no. 13.

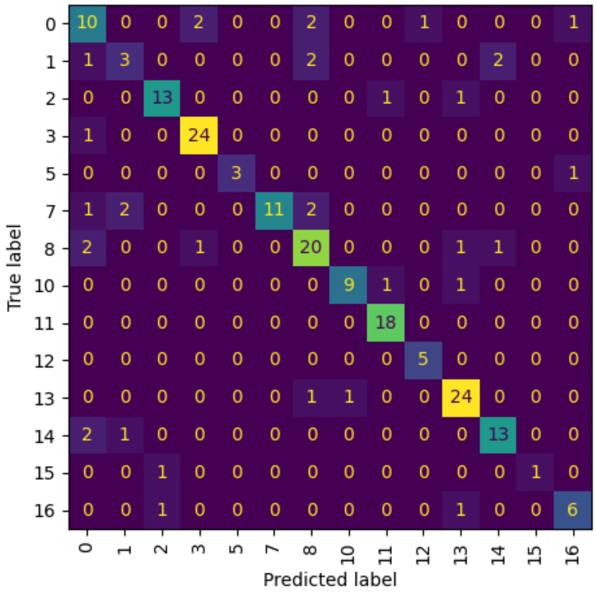
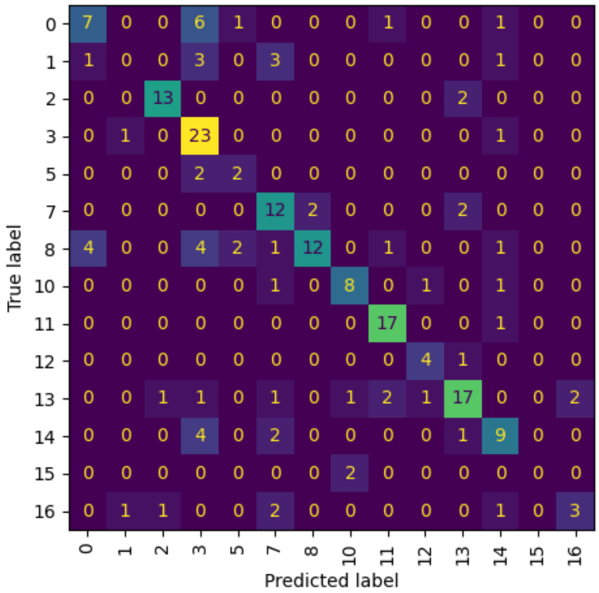


Fig. 20. MLP score matrix.

Fig 20. Shows the score matrix for the MLP classifier, demonstrating the opposite of the MSD classifier, with a strong diagonal correlation, giving a high ratio of correct predictions. Similar to

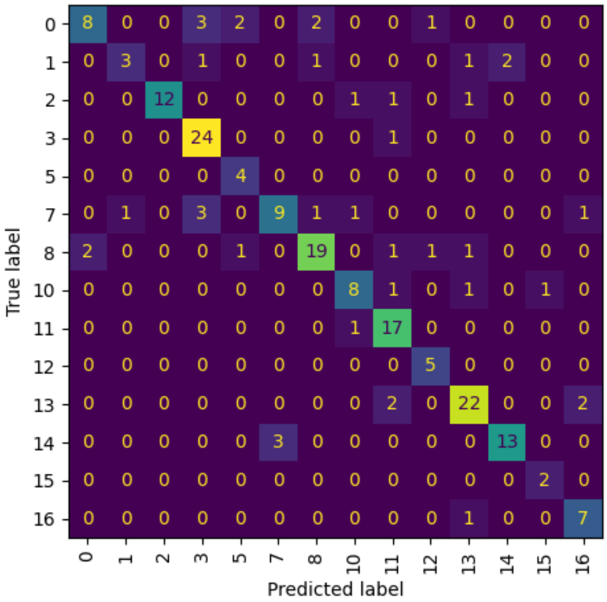
MSD, however, is the number of correct predictions for no.3 and 13., showing to be the most effective on these datapoints.



	0	1	2	3	5	7	8	10	11	12	13	14	15	16
0	7	0	0	6	1	0	0	0	1	0	0	1	0	0
1	1	0	0	3	0	3	0	0	0	0	0	1	0	0
2	0	0	13	0	0	0	0	0	0	0	2	0	0	0
3	0	1	0	23	0	0	0	0	0	0	0	1	0	0
5	0	0	0	2	2	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	12	2	0	0	0	2	0	0	0
8	4	0	0	4	2	1	12	0	1	0	0	1	0	0
10	0	0	0	0	0	1	0	8	0	1	0	1	0	0
11	0	0	0	0	0	0	0	0	17	0	0	1	0	0
12	0	0	0	0	0	0	0	0	0	4	1	0	0	0
13	0	0	1	1	0	1	0	1	2	1	17	0	0	2
14	0	0	0	4	0	2	0	0	0	0	1	9	0	0
15	0	0	0	0	0	0	0	2	0	0	0	0	0	0
16	0	1	1	0	0	2	0	0	0	0	0	1	0	3

Fig. 21. KNN score matrix.

The KNN matrix shown above also shows a strong diagonal correlation. It does not have as many positive predictions on the diagonal as MLP, but also does not have a high number of false positives outside of that. This will be reflected in the precision and recall values below.



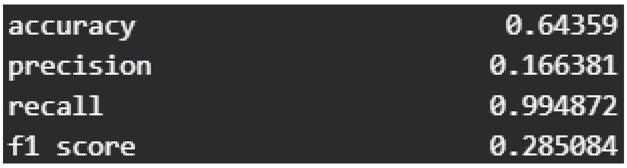
	0	1	2	3	5	7	8	10	11	12	13	14	15	16
0	8	0	0	3	2	0	2	0	0	1	0	0	0	0
1	0	3	0	1	0	0	1	0	0	0	1	2	0	0
2	0	0	12	0	0	0	0	1	1	0	1	0	0	0
3	0	0	0	24	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	4	0	0	0	0	0	0	0	0	0
7	0	1	0	3	0	9	1	1	0	0	0	0	0	1
8	2	0	0	0	1	0	19	0	1	1	1	0	0	0
10	0	0	0	0	0	0	0	8	1	0	1	0	1	0
11	0	0	0	0	0	0	0	1	17	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	5	0	0	0	0
13	0	0	0	0	0	0	0	0	2	0	22	0	0	2
14	0	0	0	0	0	3	0	0	0	0	0	13	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	2	0
16	0	0	0	0	0	0	0	0	0	0	1	0	0	7

Fig. 22. SVM score matrix.

The SVM matrix is similar to the KNN and MLP matrices, with a strong diagonal correlation. It seems to have more true positives than KNN, but less than MLP. There is a clear bias towards no.3 as each matrix including this one has a high number of true positives for it.

Tables for each classifier reporting accuracy, precision, recall, F1 score

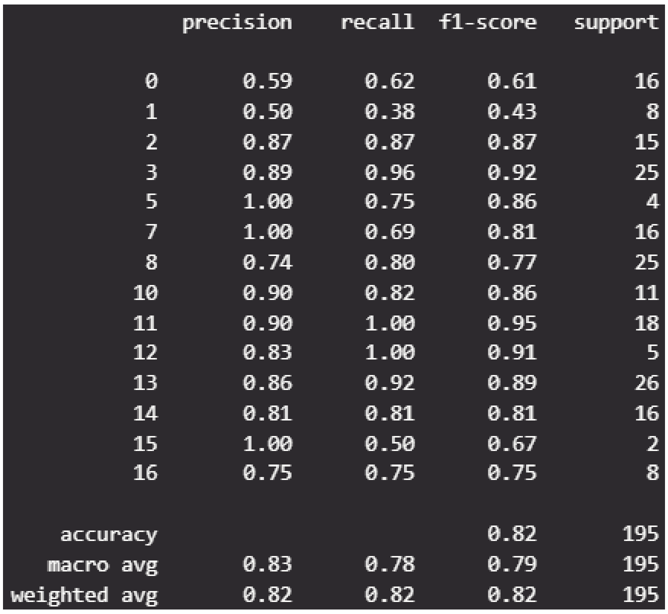
Below are tables for each classifier, showing the performance in terms of accuracy, precision, recall, and f1 score. The main focus is on accuracy; however, the other three metrics provide further insight to the bias of each algorithm.



```
accuracy          0.64359
precision        0.166381
recall           0.994872
f1 score         0.285084
```

Fig. 23. MSD performance summary

The first point noticed is the accuracy. It is very low and does not come close to the required 95%. The high recall means that there was a high level of true positives, which is good, however there was a trade-off with the low precision, meaning that there were a lot of false positives. The f1 score takes both the high recall and low precision into account, and further demonstrates the poor performance of this algorithm. The optimum f1 score is the closest to 1, and this algorithm has less than 30% of that, rendering it ineffective for this task. This could be improved with further optimisation, reducing the acceptance threshold.



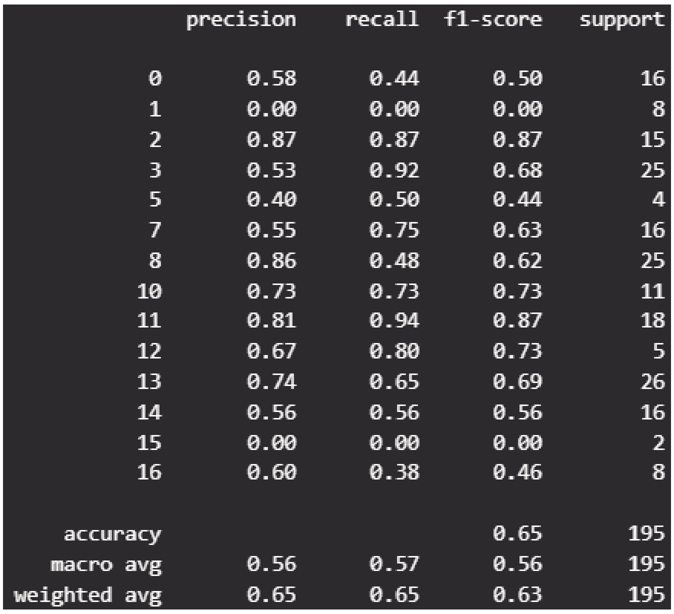
```
              precision    recall  f1-score   support

           0       0.59      0.62      0.61        16
           1       0.50      0.38      0.43         8
           2       0.87      0.87      0.87        15
           3       0.89      0.96      0.92        25
           5       1.00      0.75      0.86         4
           7       1.00      0.69      0.81        16
           8       0.74      0.80      0.77        25
          10       0.90      0.82      0.86        11
          11       0.90      1.00      0.95        18
          12       0.83      1.00      0.91         5
          13       0.86      0.92      0.89        26
          14       0.81      0.81      0.81        16
          15       1.00      0.50      0.67         2
          16       0.75      0.75      0.75         8

    accuracy                           0.82       195
   macro avg       0.83      0.78      0.79       195
weighted avg       0.82      0.82      0.82       195
```

Fig. 25. MLP performance summary

In comparison to the MSD metrics, the results shown for the MLP classifier in fig. 25. Are very promising. The accuracy is above 80%, and the highest f1 score is 0.95, with the lowest being 0.43. This is still higher than the total f1 score for the MSD algorithm. This high f1 score indicates a good balance between the precision and recall, maximizing both.



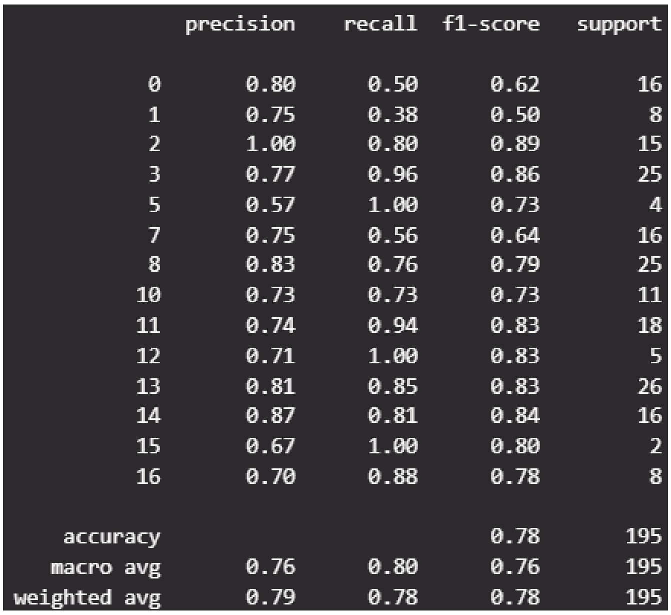
```
              precision    recall  f1-score   support

           0       0.58      0.44      0.50        16
           1       0.00      0.00      0.00         8
           2       0.87      0.87      0.87        15
           3       0.53      0.92      0.68        25
           5       0.40      0.50      0.44         4
           7       0.55      0.75      0.63        16
           8       0.86      0.48      0.62        25
          10       0.73      0.73      0.73        11
          11       0.81      0.94      0.87        18
          12       0.67      0.80      0.73         5
          13       0.74      0.65      0.69        26
          14       0.56      0.56      0.56        16
          15       0.00      0.00      0.00         2
          16       0.60      0.38      0.46         8

    accuracy                           0.65       195
   macro avg       0.56      0.57      0.56       195
weighted avg       0.65      0.65      0.63       195
```

Fig. 26. KNN performance summary

Fig. 26. Shows the performance of the KNN algorithm. The accuracy is significantly lower than the MLP performance, and the f1 score is average. However, interestingly, participants no.1. and 15. Have scores of 0 across the board, which could mean that they were predicted with 100% error rate. Surprisingly, the accuracy score is similar to MSD, differing by only 1%. The low performance may be due to a poorly optimized K- value, as a higher value was used for this test.



```
              precision    recall  f1-score   support

           0       0.80      0.50      0.62        16
           1       0.75      0.38      0.50         8
           2       1.00      0.80      0.89        15
           3       0.77      0.96      0.86        25
           5       0.57      1.00      0.73         4
           7       0.75      0.56      0.64        16
           8       0.83      0.76      0.79        25
          10       0.73      0.73      0.73        11
          11       0.74      0.94      0.83        18
          12       0.71      1.00      0.83         5
          13       0.81      0.85      0.83        26
          14       0.87      0.81      0.84        16
          15       0.67      1.00      0.80         2
          16       0.70      0.88      0.78         8

    accuracy                           0.78       195
   macro avg       0.76      0.80      0.76       195
weighted avg       0.79      0.78      0.78       195
```

Fig. 27. SVM performance summary

The SVM performance is decent, having a higher accuracy than KNN. The f1 score is also significantly higher than KNN, however does not reach the same peak as MLP. The lowest f1 score here is 0.5 however, which is higher than the lowest MLP score.

Comparing suitability

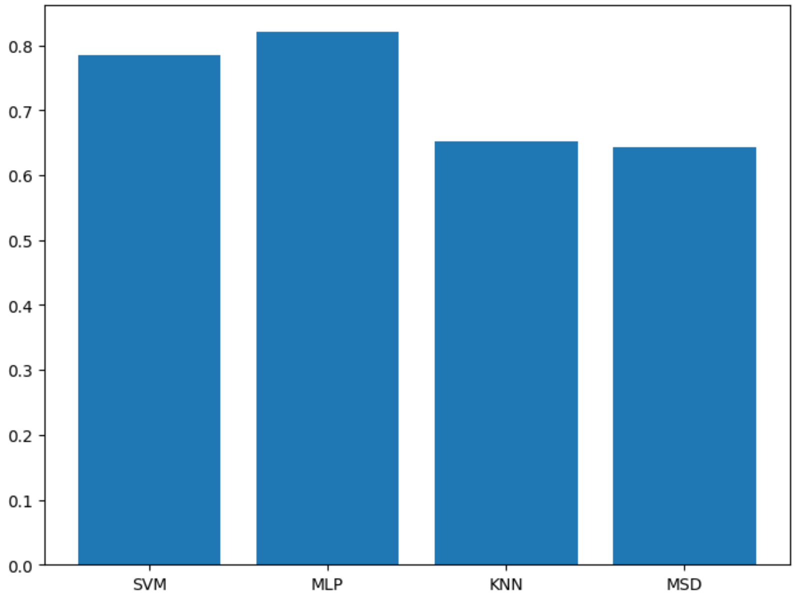


Fig. 28. Graph comparing accuracy of each algorithm.

Fig. 28. Shows the accuracy of each classifier, with a clear winner in MLP, but a close second with SVM. KNN and MSD are both in the bottom end of the accuracy, being less than 70%. SVM, KNN, and MSD, would not be suitable for this task, as they all have error rates over 20%, 4 times more than necessary.

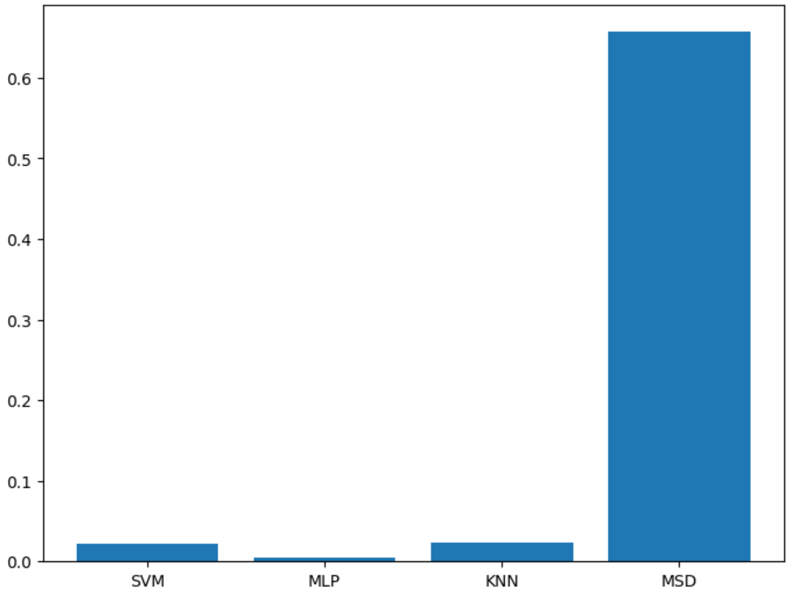


Fig. 29. Graph comparing runtime of each algorithm.

This graph shows the runtime of each algorithm in seconds. The runtime of MSD stands out as being over 10 times higher than the other classifiers, with a runtime of over 0.6 seconds. The order of efficiency mimics the accuracy, with MLP being the fastest, however SVM and KNN are closer in this case.

Discussion

Evaluation

After the experiment and evaluation was carried out, the objectives and requirements were referred to, making sure that the success criteria and hypotheses are satisfied.

Develop a reliable system for recording keystroke biometrics.

This objective describes 4 main characteristics to be satisfied in order to create a reliable system: Data integrity, data security, scalability, and usability.

Data integrity is upheld throughout the system, on both the frontend and backend. The frontend maintains accuracy and consistency via several restrictions on the input data. Functions that run on every key-press check for user errors, preventing the entry of incorrect characters therefore keeping consistency across samples. It also satisfies the secondary objective; ‘record arrays of keystroke data’, by recursively compiling the keystrokes entered. The backend also contributes, as it makes sure that user credentials are verified before accepting any data, preventing incorrect and inconsistent data entries.

Data security is mostly handled by the backend application through various methods. Once such method is hosting it on the AWS cloud, a remote, secure hosting service, as opposed to a local system. The first layer of security is CORS filter, that only allows connections from authorized domains, filtering out a range of threats such as phishing attacks and DDoS attacks, which both take advantage of publicly accessible addresses to access or reduce performance of a system.

The second layer of security is a firewall in the AWS instance, which only allows connections via HTTPS and SSH. SSH connections require a key file, only allowing direct server access to the owner. HTTPS prevents numerous vulnerabilities, such as Man-in-the-middle (MitM) attacks, which intercept traffic between the client and the server. As HTTPS encrypts this data, it is much more difficult to decrypt and interpret, should an attack occur. There is another layer of security that was implemented in the server routing files, which only allows access to the server via the domain name, re-routing any attempts to access via the direct IP address. This stops attackers from circumventing the firewall.

Scalability is implemented in both the backend and the frontend. As the frontend is a website hosted on the Sussex University server, it can be accessed from any number of devices at one time, allowing for many users to enter data at the same time. The backend database is designed to handle requests simultaneously with FastAPI, and SQLite enables structured storage of the large datasets necessary.

Usability is one of the main focuses of the frontend, having been tested with participants before deployment. The website was designed a webpage consisting of just one page, with large fonts and 3 distinct sections to follow. Clear messages utilizing colours to represent the status were also used to improve how intuitive the experiment is. AWS was used because it provides a simple interface for server management and maintenance.

In summary, all sections of this objective were fully completed, satisfying all secondary objectives and proving the hypothesis claiming that “a reliable system making for recording a large sample of inter-key times for different users can be developed.”

Evaluate whether keystroke dynamics can be used to accurately authorize a user.

The objective referenced here outlines what the algorithms analyses should aim to achieve. It describes how the chosen algorithm should have the lowest error rate and runtime, maximising efficiency, and scalability.

MLP is the clear choice because it has the lowest runtime, in combination with the highest accuracy overall. This is a result of its complexity and recursive optimization, iterating over the weights between each neuron layer, updating them till the accuracy is maximised. The score matrix also shows the highest diagonal trend, with the most correctly assigned labels out of any algorithms, although the KNN and SVM algorithms also had a similar trend, but with less true positives.

SVM came in close second, with the second lowest runtime and error rates. It also had a stronger diagonal trend than KNN on the score matrix. The f1 scores were similar to KNN but the accuracy was greater by over 10%, making it a stronger candidate for authorization.

KNN is in the middle ground, however it is stunted by its low accuracy, which was marginally better than MSD, at 65%. The score matrix has a strong diagonal trend however, showing potential given further optimization.

MSD was shown to be the least effective algorithm, with the highest runtime by a large margin. This is because it contains several loops, manually comparing each sample to find a score. This leads to an exponential runtime in relation to the size of the dataset. The score matrix also lacked a strong diagonal correlation, showing a wide spread of false positive predictions.

In conclusion, although MLP was the highest performing algorithm, the hypothesis was not met. It claims that an algorithm with an error rate below 5% could be identified, however the error rate of MLP was 18%, 3.6 times larger than the claim. This is unacceptable for an authentication system, which ideally should have a much lower error rate than even 5% for real-world use.

Existing methods

Statistics shown in [12] from similar studies report various accuracies for different authorization algorithms. One such study showed an error rate of 5.8%, also utilizing inter-key times to distinguish between users. In comparison to the values found here, the studies shown have lower error rates, however the sample sizes were much larger, some containing over double the number of participants. These studies use varying methods for recording inter-key times, such as large paragraphs of text being typed a small number of times.

Limitations

There are several limitations encountered in this project. One of the major constraints is the time limitation, where more time would allow for the development of more complex, efficient algorithms. However, there are other factors that can limit the accuracy of the system, including the number of samples available. To help overcome these limitations, it is possible to use a combination of algorithms to improve the accuracy of the system. Through this, the system can take advantage of the strengths of each approach and potentially overcome some of the downfalls associated with each individual algorithm.

The Future

Even though the hypothesis was not met, keystroke dynamics may still be a viable choice for user authentication. More features can be added, such as key pressure and typing styles. With the increasing amount of data being transmitted and stored electronically, the importance of new authentication measures grows. As biometrics evolve, there will be more opportunities to enhance our digital security and privacy. However, as with any new development, there will also be new challenges and vulnerabilities to address. Therefore, research and development in the field of computing will be critical in staying ahead of new threats, ensuring the safety of individuals and organizations in an increasingly interconnected world.

I hope that this research inspires the reader.

References

[1] Y. Zhao, “Learning User Keystroke Patterns for Authentication,” *International Journal of Computer, Electrical, Automation, Control and Information Engineering,* 2008.

[2] D. Umphress and G. Williams, “Identity verification through keyboard characteristics,” *International Journal of Man-Machine Studies,* 1985.

[3] E. H. Spafford, “OPUS: Preventing weak password choices,” *Computers & Security,* 1992.

21 A. J. Smola and B. Scholkopf, “A tutorial on support vector regression,” *Statistics and Computing,* 2004.

[5] G. Romain, M. El-Abed and C. Rosenberger, “Keystroke dynamics with low constraints SVM based passphrase enrollment,” in *IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems,* 2009.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research,* 2011.

[7] F. Monrose and A. D. Rubin, “Keystroke dynamics as a biometric for authentication,” *Future Generation Computer Systems,* 2000.

[8] S. Kloppenburg, “Securing Identities: Biometric Technologies and the Enactment of Human Bodily Differences,” *Science as Culture,* 2020.

[9] D. V. Klein, “Foiling the cracker: A survey of, and improvements to, password security,” *Programming and Computer Software,* 1992.

[10] D. Gunetti and C. Picardi, “Keystroke analysis of free text,” *ACM Transactions on Information and System Security,* 2005.

[11] R. Giot, M. El-Abed and C. Rosenberger, “Keystroke Dynamics Authentication,” *Intech Book on Biometrics,* 2011.

[12] H. Crawford, “Keystroke dynamics: Characteristics and opportunities,” in *Eighth International Conference on Privacy, Security and Trust,* 2010.

[13] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt and G. Varoquaux, “API design for machine learning software: experiences from the scikit-learn project,” in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning,* 2013.

[14] J. Brownlee, “Crash Course on Multi-Layer Perceptron Neural Networks - MachineLearningMaster.com,” 17 May 2016. [Online]. Available: https://machinelearningmastery.com/neural-networks-crash-course/.

[15] J. Brownlee, “Why Initialize a Neural Network with Random Weights? - MachineLearningMaster.com,” 14 August 2022. [Online]. Available:

https://machinelearningmastery.com/why-initialize-a-neural-network-with-random-weights/#:~:text=The%20weights%20of%20artificial%20neural,model%2C%20called%20stochastic%20gradient%20descent..

[16] J. Brownlee, “How to Manually Optimize Neural Network Models - MachineLearningMastery.com,” 4 December 2020. [Online]. Available: https://machinelearningmastery.com/manually-optimize-neural-networks/#:~:text=Optimize%20Neural%20Networks,-Deep%20learning%20or&text=Models%20are%20trained%20by%20repeatedly,stochastic%20gradient%20descent%20optimization%20algorithm..

[17] S. Bannerjee, Z. A. Syed and B. Cukic, “Keystroke Recognition,” *Encyclopedia of Biometrics,* 2009.

[18] IBM, “What is the k-nearest neighbors algorithm? | IBM,” 2023. [Online]. Available: https://www.ibm.com/topics/knn.

[19] European Commision, “Visa Information System,” 2023. [Online]. Available: https://home-affairs.ec.europa.eu/policies/schengen-borders-and-visa/visa-information-system_en.

[20] Schengen Visa, “VIS - The Schengen Visa Information System - SchengenVisaInfo.com,” 2023. [Online]. Available: https://www.schengenvisainfo.com/security-system/vis/.

[21] National Cyber Security Centre, “Understanding biometrics - NCSC.GOV.UK,” 2023. [Online]. Available: https://www.ncsc.gov.uk/collection/biometrics/understanding-biometrics.

[22] Google; Harris Poll, “Online Security Survey,” February 2019. [Online]. Available: https://services.google.com/fh/files/blogs/google_security_infographic.pdf.

[23] Mtplotlib, “Matplotlib - Visualization with Python,” [Online]. Available: https://matplotlib.org/.

[24] Let's Encrypt, “Let's Encrypt,” 2023. [Online]. Available: https://letsencrypt.org/.

[25] Biometric Solutions, “Keystroke Dynamics,” 2023. [Online]. Available: https://www.biometric-solutions.com/keystroke-dynamics.html.

[26] FastAPI, “FastAPI,” 2023. [Online]. Available: https://fastapi.tiangolo.com/.

[27] scikit learn, “1.6. Nearest Neighbors - scikit-learn 1.2.2 documentation,” 2023. [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification.

[28] scikit learn, “1.4. Support Vector Machines - scikit-learn 1.2.2 documentation,” 2023. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation.

[29] scikit learn, “1.17. Neural network models (supervised) - scikit-learn 1.2.2 documentation,” 2023. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html#classification.

Appendices

Participant information sheet & consent form

PARTICIPANT INFORMATION SHEET

Keystroke Authentication Research Project

Invitation paragraph

You are being invited to take part in a research study. Before you decide whether or not to take part, it is important for you to understand why the research is being done and what it will involve. Please take time to read the following information carefully.

What is the purpose of the study?

The aim of this project is to investigate whether keystrokes can be utilized for user authentication and will run till mid-April. You will be asked to type a given paragraph of text a number of times, and your keystroke pattern data will be recorded and analysed.

Do I have to take part?

It is up to you to decide whether or not to take part. If you do decide to take part, you will be given this information sheet to keep and be asked to sign a consent form. If you decide to take part, you are still free to withdraw at any time and without giving a reason. This will not impact your future marks or studies as a student.

Will my information in this study be kept confidential?[1]

Any personal data will be kept strictly confidential and securely stored on the researcher's system. The keystroke data will be analysed and submitted in a dissertation research paper, with participants anonymity upheld. The data will be deleted after use.

Who has approved this study?

This research has been approved through the School of Engineering and Informatics ethical review process. Application no. ER/OB217/1

Contact for Further Information

If you have any concerns relating to this project please contact Ore Benson at ob217@sussex.ac.uk, or Peter Cheng at p.c.h.cheng@sussex.ac.uk.

INSURANCE

The University of Sussex has insurance in place to cover its legal liabilities in respect of this study.

Thank you for reading this information sheet. A consent form is also attached.

16/02/23

CONSENT FORM FOR PROJECT PARTICIPANTS

Title of Project: KEYSTROKE AUTHENTICATION RESEARCH PROJECT

Name of Researcher and School: Ore Benson – School of Engineering and Informatics

C-REC Ref no: ER/OB217/1

Please tick box.

	YES	**NO**
• *I consent to being interviewed by the researcher*	□	□
• *I agree to making myself available for a further interview should it be required*	□	□
• I understand that any information I provide is confidential, and that no information that I disclose will lead to the identification of any individual in the reports on the project, either by the researcher or by any other party	□	□
• I have read the information sheet, had the opportunity to ask questions and I understand the principles, procedures and possible risks involved.	□	□

- I understand that my personal data will be used for the purposes of this research study and will be handled in accordance with Data Protection legislation. I understand that the University's Privacy Notice provides further information on how the University uses personal data in its research. □ □

- I understand that my participation is voluntary, that I can choose not to participate in part or all of the project, and that I can withdraw at any stage of the project without being penalised or disadvantaged in any way. □ □

- I agree to take part in the above University of Sussex research project □ □

Name:

Signature

Date: