VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

Authentication using Keystroke Dynamics

A project submitted

in partial fulfillment of the requirements for the degree of

Bachelor of Technology in Computer Science and Engineering

By:

Namit Nathwani (17BCI0113)

Satyaki Suman (17BCI0026)

Ishita Gupta (17BCI0084)

Njokosi J Kawunju (17BCI0186)

Course Instructor:

Dr. Ramesh Babu K.

Professor

Vellore - 632014, Tamil Nadu, India

November 2020

Undertaking

This is to declare that the project entitled “Authentication using Keystroke Dynamics” is an original work done by undersigned, in partial fulfillment of the requirements for the degree “Bachelor of Technology in Computer Science and Engineering” at School of Computer Science and Engineering, Vellore Institute of Technology (VIT), Vellore.

All the analysis, design and system development have been accomplished by the undersigned. Moreover, this project has not been submitted to any other college or University.

Namit Nathwani (17BCI0113)

Satyaki Suman (17BCI0026)

Ishita Gupta (17BCI0084)

Njokosi J Kawunju (17BCI0186)

Abstract

Passwords are an age-old authentication method. With the recent hacks and leaks, keeping a password secure is proving to be tougher day by day. A solution to this problem is to add other factors to authentication. The most non-invasive of which is typing patterns. This is called Keystroke Dynamics. It is the detailed timing information which describes exactly when each key was pressed and when it was released as a person is typing at a computer keyboard. This has a significant advantage over other methods since it is a biometric trait and requires no additional inputs from the user in terms of things like an OTP.

The behavioural biometric of Keystroke Dynamics uses the manner and rhythm in which individuals type characters on a keyboard or keypad. The keystroke rhythms of a user are measured to develop a unique biometric template of the user's typing pattern for future authentication. This project aims to build a customisable and extensible keystroke dynamics authentication system and dashboard, which is modular and easy to use in conjunction with existing authentication systems.

Table of Contents

1 Introduction

Over the years the requirement of a quicker methodology for the implementation of Two Factor Authentication has grown. With the growth of 2FA, the overall complexity in verification has also drastically increased. A lot of users do not want to go through the hassle each time and require a quicker method of authentication while maintaining a high level of security at the same time. Here is where Keystroke Dynamics comes in. We can effectively solve both the issue of speed and security in one. Technical accomplishments over the previous decade have brought about improved network administrations, especially in the zones of execution, reliability, and accessibility, and have essentially decreased operation costs because of their more productive and efficient use.

Keystroke Dynamics is a measure which authenticates the access to PCs by recognizing certain unique patterns in a user's typing rhythm and pattern while logging in. We feel that the use of keystroke dynamics is a natural and important choice for cybersecurity. This argument comes from the fact that similar neuro-physiological factors that make written signatures unique are also exhibited in a user's typing pattern. When a person types, the latencies between successive keystrokes, keystroke durations, finger placement and applied pressure on the keys can be used to construct a unique signature for that individual. For well-known, regularly typed strings, such signatures can be quite consistent. Furthermore, recognition based on typing rhythm is not intrusive, making it quite applicable to computer access security as users will be typing at the keyboard anyway.

2 Literature Survey

[1] The behavioural features such as voice patterns and keystroke patterns haven't yet led to stable user authentication systems due to lack of an acceptable level of accuracy. This paper proposes a method to limit that inaccuracy by extensive testing on 154 individuals and allowing the possibilities of human error such as typing errors without any major alterations to the authentication system. The paper uses 2 major factors to calculate the distance of two samples first, the duration for which the key is pressed down and second, the latency between two keystrokes.

[2] This paper talks about various evaluation techniques for different biometric authentication methods. It also throws light on the lack of details in the previous papers of literature and a dynamic database to test the accuracy of different keystroke authentication models. The GREYC- keystroke software incorporates all the necessary testing factors for evaluating a keystroke authentication algorithm, such as dependency to keyboard, computer operating system, knowledge of the password, size of the password, the content of the password, new and old passwords etc.

[3] The paper compares the performances of different keystroke anomaly detectors across different studies in the keystroke dynamics literature. The objective was to collect a data set, develop an evaluation procedure, and measure the performance of many anomaly detection algorithms on an equal basis. The paper found that the Mahalanobis (Nearest Neighbour detector) has the lowest error rates. It also provides a data set and evaluation methodology that can be used to assess new detectors and report comparative results.

[4] In this paper, the author gives a detailed description of C++ toolkit built using xview library routines helpful in the diagnosis of the system behaviour and generating graphs for MatLab and Gnuplot. The toolkit uses k nearest neighbours algorithm to cluster people comprising disjoint traits, this clustering helps in user identification. The accuracy of the toolkit for a user base of 63 was found to be 83.22%-92.14%.

[5] The paper attempts to provide a comprehensive survey of research on keystroke dynamics described in the last two decades. The techniques were categorized based on the features, feature extraction methods and classification methods employed and their performance has been discussed. It has been observed that most of the work employed keystroke duration, latency or digraph as features, and the combined use of these features leads to low False Alarm Rates and Imposter Pass rates.

[6] This paper describes how a continuous keystroke dynamics system needs to be evaluated. This system is one that, rather than denying the user access on the first erroneous attempt, it grades the user's access requests on a genuine confidence gradient. The key factor in the evaluation of a CKD system is a so-called "penalty-and reward" function that is designed to adapt the trust level of the system.

[7] The authors have investigated a solution that enables modelling an individual's keystroke dynamics while minimizing the used samples for the definition of the reference template. The size of each user reference would increase while using the system, to reach a maximum size equal to 10 thanks to the double serial mechanism. The paper also evolved a GA-KNN verification method to achieve better performances during the whole adaptation session. the weights from different distances for the KNN classifier, in addition to the GA optimization, are useful to minimize recognition errors. With regards to previous work, the suggested method shows great performance improvement.

[8] This paper makes use of non-conventional features like words per minute, error rate, and capslock usage to highlight the users' free-text keystroke. The main problem is solved using decision trees and support vector machines. The method allows for extra information to be extracted from user input and has lower error rates. Overall a positive, well functioning system was created.

[9] This paper provides a sharp contrast between different keystroke dynamic methods that are plausible and the advantages of individual features. The classification technique used is of the highest order and results obtained are ideal.

[10] This approach uses Convoluted Neural Networks to create a secure keystroke authentication process for the user. This method is a good safeguard to brute force cracking and other forms of hacking and attacks. A certain threshold is generated for the user and if the adulterated user tries to replicate the password the access will be denied. The accuracy of this model is 97%. The project also adds parallel computing into the system which enhances speed by a multiplier of four.

3 Proposed Methodology

Keystroke Dynamics is the study of the typing patterns of people to distinguish them from one another, based on these patterns. Every user has a unique typing pattern that is very tough to replicate. There will always be minor variations characteristic of a user that cannot be replicated. This forms the basis of the study of Keystroke Dynamics. In this project, we aim to implement User Authentication Based on Keystroke Dynamics and provide a dashboard to allow testing.

Three features are widely used for keystroke dynamics:

- **Hold time** – the time between press and release of a key.
- **Keydown-Keydown time** – the time between the pressing of consecutive keys.
- **Keyup-Keydown time** – the time between the release of one key and the press of the next key.

Our system aims to be an extensible one, allowing the users to customise and try out the various detectors available in order to figure the best one for their use-case. The system also has algorithms designed with the end-user in mind, and is thus modular and easily customisable. The keylogger and the JavaScript files required for the login and signup pages can be added to any existing page with minimal changes to the logic. This makes integration very simple with changes only required in the HTML.

Our system will provide a dashboard which gives the user the freedom to experiment and understand the product in detail with graphs and detailed tables in order to determine the best detectors and their appropriate parameters for any particular use case.

The production system can then use pre-determined values for these parameters to obtain the best results.

4 Architecture/Design

The project is based on a client-server architecture, using Node.JS for the backend, interfacing with a MongoDB database. The client side is built using HTML, CSS and JavaScript.

The backend of the system is written in **Node.JS** to allow for high concurrency and scalability, even when used in systems with high throughput. The database being used is **MongoDB**, which further improves the performance of the system, allowing for easy extensibility and providing high availability.

The backend is written completely in Node.JS with no external dependencies on other environments, allowing for a simple and streamlined setup.

The backend also performs the majority of the calculations, so the client side never has access to sensitive data and the computing cost is shifted to the server, instead of the clients' systems.

It also has a lot of minor optimizations like using the database to cache frequently computed values in order to decrease the Round Trip Time on each login request, significantly improving the performance of each request. The system averages ~100-120ms for each authentication request, which is a very good duration for any secure authentication system.

We use a NoSQL database to allow for flexibility in adding and updating more detectors. It also allows us to cache pre-computed values without requiring other solutions and saves on unnecessary setup and additional memory.

5 Description of Modules

5.1 Signup

The signup module is a simple webpage that interfaces with our backend API. The signup process is very easy and only slightly more tedious than normal signup processes, requiring three inputs of the users' passwords instead of the usual two. This makes the process more secure, while not inconveniencing the user.

5.2 Login

The login module is where the user enters his/her login id and password and they can view if their attempt was either successful or unsuccessful using our 2-factor authentication metrics and the parameters defined by the user. This is part of the Dashboard of our project, and interfaces with our backend API.

5.3 Data Visualisation and Parameter Modification

The HTML and JS are interlinked with an algorithm that lets the user change only one object in the JS file and update the HTMl accordingly to add more elements. Updating values and handling interactions are all done by the script with no user interaction required. Charts and Tables are given to visualise the data and understand the impacts of the parameters chosen.

A section of the dashboard is dedicated to the detectors, wherein the user is allowed to change and experiment with the various detectors by enabling and disabling them as well as modifying their parameter values. This allows the system admin to determine the strictness of the system before deploying it to make sure it works for their intended use-cases without causing disruption.

5.4 Feature Extraction

This is the module that deals with the processing of input data. This is handled on the backend to prevent malicious actors from tampering with the user data and to move all computation away from the client side, to minimise the code required to integrate our system to existing products.

We extract the times for the three main features (hold, flight, and down-down). These can then be used by the detectors to give a result.

5.5 Database

For the database we are using MongoDB which is a NoSQL database. In the database we are storing data about the user's id, password and keystroke dynamics like hold time, flight time, down time, etc. For each of these properties we also store pre-calculated aggregates like means, standard deviations, filtered values and covariances. This allows us to reduce our response time and save on computation cost for failed attempts.

5.6 Authentication (Detectors)

5.6.1 Manhattan

Manhattan Distance is the distance between two points measured along their axes at right angles. It is also known as the city-block distance due to its resemblance to walking along a city block.

Manhattan distance is used if you want to place less emphasis on outliers, as it will try to reduce all errors equally since the gradient has constant magnitude. It is also more efficient than Euclidean distance when dealing with high dimensional data, which is the case in this project. This distance D is calculated between the inputs and the mean vectors generated from the user's dataset.

$$D = \sum_{0}^{n} \left\| \mu - x_i \right\|$$

5.6.1.1 Manhattan (Individual)

For the Individual version of this detector, we take two parameters, namely the SD Multiplier (M) and a threshold value (T). The distance of each keystroke from the mean of that keystroke in the database is calculated. This distance should lie in the range $[-M \cdot SD,\ M \cdot SD]$. If atleast $T\%$ of the inputs lie in the accepted range, the attempt is considered accepted.

This detector performs best with a low amount of data (<10 attempts), after which the accuracy deteriorates.

5.6.1.2 Manhattan (Population)

This detector uses the same concept of Manhattan distance, but instead of checking each keystroke, it calculates the distance between the input vector and population mean vector. If the distance falls within a set threshold, the attempt is accepted.

This detector performs much better than the Individual detector when being used on datasets with more than 5 elements and is extremely sensitive.

5.6.2 Manhattan Filtered

Manhattan Filtering works on the principle of filtering out of outliers from the user dataset. This yields a clean dataset that is much more accurate and useful to determine valid attempts. This helps get rid of erroneous inputs that can occur naturally. This detector is thus more sensitive than the Manhattan Detector.

5.6.2.1 Manhattan Filtered (Individual)

This works on the same principle as the Manhattan (Individual) detector, with a cleaner dataset. This improves its accuracy significantly, especially in smaller datasets.

This detector performs best with a low amount of data (<10 attempts), after which the accuracy deteriorates.

5.6.2.2 Manhattan Filtered (Population)

This works on the same principle as the Manhattan (Population) detector, with a cleaner dataset. This improves its accuracy significantly, especially in smaller datasets.

This detector performs much better than the Individual detector when being used on datasets with more than 5 elements and is extremely sensitive.

5.6.3 Mahalanobis

Mahalanobis Distance is another method which can be used for outlier detection and group classification. It is chiefly used with the Chi-Squared and the PCA techniques. It is calculated by taking the sum of squares of all the non zero standardized principal components.

Mahalanobis Distance is one of the best methods to detect outliers and gives results with high accuracy. We use it as one of the filters to detect any pattern that might be an outlier.

The detector calculates distance from the input vector to the population mean vector using the given formula

$D = \sqrt{\left((x - \mu) \cdot S^{-1} \cdot (x - \mu)^T\right)}$, where S is the covariance matrix of the features.

If the value falls within a defined threshold, the attempt is accepted.

6 Implementation

6.1 Feature Extraction

```
const processKeystrokeData = ({ keydown, keyup }) => {
      const data = {
            hold: {
                  keys: [],
                  times: [],
            },
            flight: {
                  keys: [],
                  times: [],
            },
            dd: {
                  keys: [],
                  times: [],
            },
            full: {
                  keys: [],
                  times: [],
            },
      };

      for (let i = 0; i < keydown.length; i += 1) {
            const { code: downCode, key: downKey, time: downTime } =
            keydown[i];
            const { code: upCode, key: upKey, time: upTime } = keyup[i];
            const holdTime = upTime - downTime;

            if (downKey !== upKey || downCode !== upCode) {
                  logger.error(`Found a mismatch ${downKey} & ${upKey}`);
                  logger.error(`Keydown: ${keydown}\nKeyup: ${keyup}`);
```

```
        }

        data.full.keys.push(`H:${downKey}`);
        data.full.times.push(holdTime);
        data.hold.keys.push(downKey);
        data.hold.times.push(holdTime);

        if (i < keydown.length - 1) {
            const { key: nextDownKey, time: nextDownTime } = keydown[i +
            1];
            const keyString = `${downKey}:${nextDownKey}`;
            const flightTime = nextDownTime - upTime;
            const ddTime = nextDownTime - downTime;

            data.full.keys.push(`F:${keyString}`);
            data.full.times.push(flightTime);

            data.full.keys.push(`DD:${keyString}`);
            data.full.times.push(ddTime);

            data.flight.keys.push(keyString);
            data.flight.times.push(flightTime);

            data.dd.keys.push(keyString);
            data.dd.times.push(ddTime);
        }
    }

    return data;
};
```

6.2 Covariance

```
const covariance = (x, y, xMean, yMean, n) => (1 / (n - 1)) * ss.sum(
Array(n).fill(0).map((v, i) => (x[i] - xMean) * (y[i] - yMean)),
);
```

6.3 User Model

```
const userSchema = new mongoose.Schema({
	username: { type: String, unique: true },
	password: String,

	keystrokeDataTimestamps: [Date],
	keystrokeData: {
		hold: {
			keys: [String],
			times: [[Number]],
			sums: [Number],
			means: [Number],
			sd: [Number],
			filteredMeans: [Number],
			filteredSd: [Number],
			covMatrix: [[Number]],
		},
		flight: {
			keys: [String],
			times: [[Number]],
			sums: [Number],
			means: [Number],
			sd: [Number],
                filteredMeans: [Number],
                filteredSd: [Number],
                covMatrix: [[Number]],
            },
            dd: {
                keys: [String],
                times: [[Number]],
                sums: [Number],
                means: [Number],
                sd: [Number],
                filteredMeans: [Number],
                filteredSd: [Number],
                covMatrix: [[Number]],
            },
            full: {
                keys: [String],
                times: [[Number]],
                sums: [Number],
                means: [Number],
                sd: [Number],
                filteredMeans: [Number],
                filteredSd: [Number],
                covMatrix: [[Number]],
            },
        },
});
```

7 Testing

7.1 Data Description

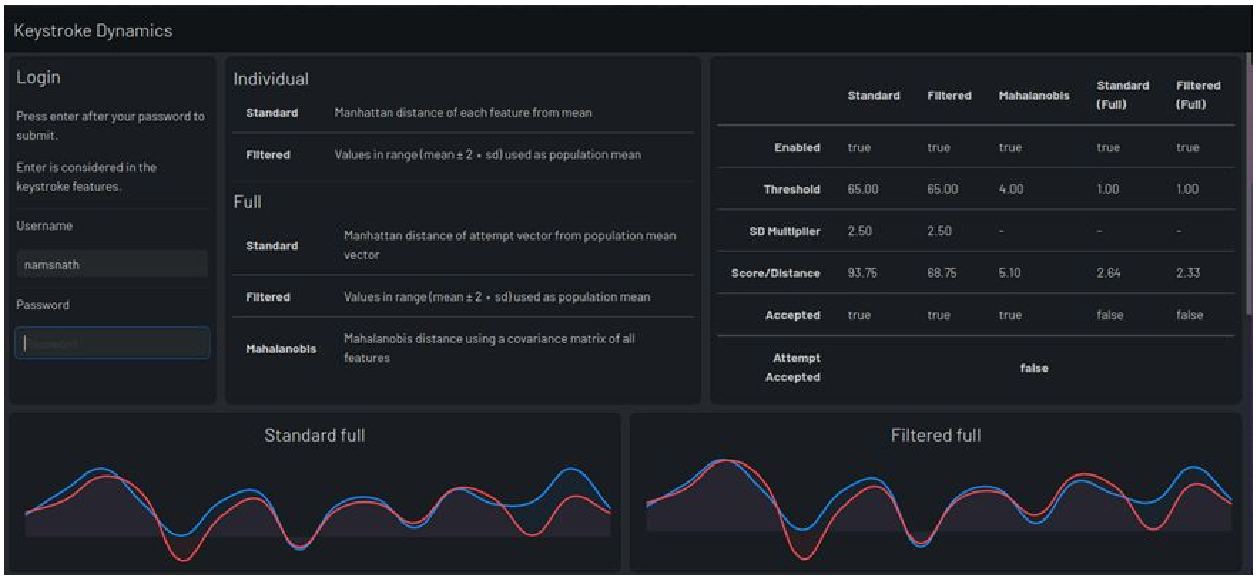
We will be tracking the user's keystrokes and providing detailed feedback to the administrator of the site. This data will include the user's keystrokes and the time taken for each keystroke. The keystrokes can be tracked to allow the system to get more refined over time. These data will be stored in the system and used to generate analytics for the administrator to configure the system for better performance.

The system will also show graphs for each individual user so that the time variations and different patterns can be used by the system to refine itself for the particular user without affecting the rest of the population data.

7.2 Performance Metrics

Our demo has been proven to be almost 50% faster than current industry standards. In terms of security, it is up to the administrator to tweak the settings to obtain optimum accuracy for their use-case. The average response time of an authentication request on our system is ~100-120ms which is a very good time for any Two Factor Authentication system.

7.3 Screenshots



Keystroke Dynamics

Login

Press enter after your password to submit.

Enter is considered in the keystroke features.

Username: namsnath

Password: Password

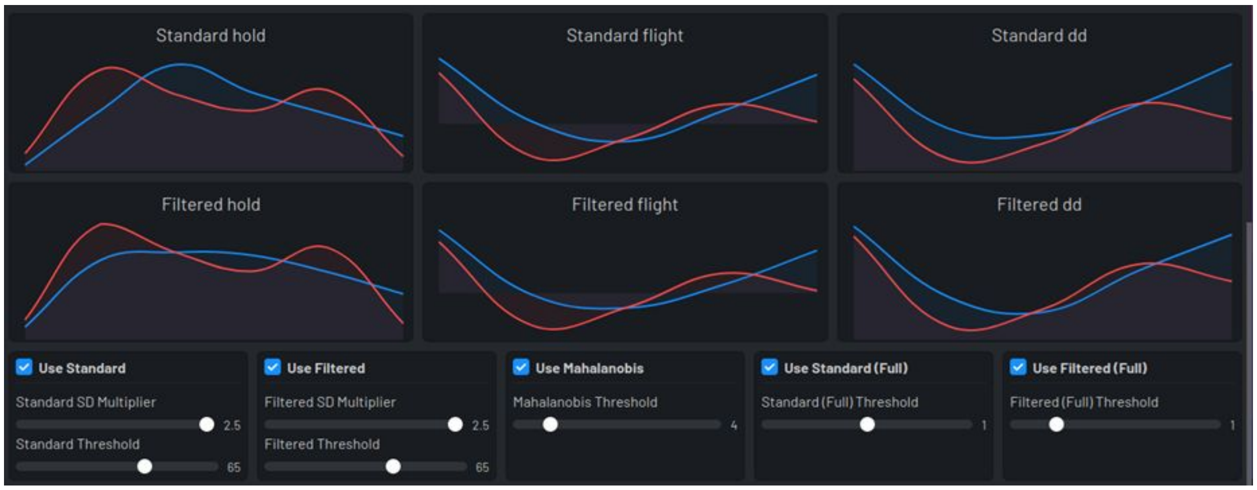
Individual

Standard	Manhattan distance of each feature from mean
Filtered	Values in range (mean ± 2 • sd) used as population mean

Full

Standard	Manhattan distance of attempt vector from population mean vector
Filtered	Values in range (mean ± 2 • sd) used as population mean
Mahalanobis	Mahalanobis distance using a covariance matrix of all features

	Standard	Filtered	Mahalanobis	Standard (Full)	Filtered (Full)
Enabled	true	true	true	true	true
Threshold	65.00	65.00	4.00	1.00	1.00
SD Multiplier	2.50	2.50	-	-	-
Score/Distance	93.75	68.75	5.10	2.64	2.33
Accepted	true	true	true	false	false
Attempt Accepted			false		



☑ **Use Standard**

Standard SD Multiplier: 2.5

Standard Threshold: 65

☑ **Use Filtered**

Filtered SD Multiplier: 2.5

Filtered Threshold: 65

☑ **Use Mahalanobis**

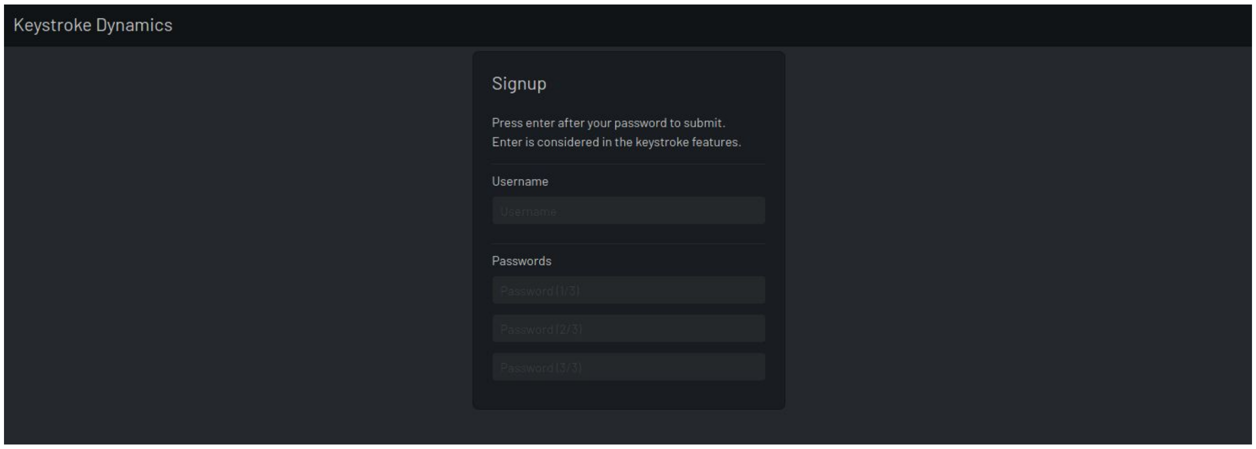
Mahalanobis Threshold: 4

☑ **Use Standard (Full)**

Standard (Full) Threshold: 1

☑ **Use Filtered (Full)**

Filtered (Full) Threshold: 1



Keystroke Dynamics

Signup

Press enter after your password to submit.
Enter is considered in the keystroke features.

Username: Username

Passwords: Password (1/3), Password (2/3), Password (3/3)

8 Contributions

Register Number	**Name**	**Contribution**
17BCI0026	Satyaki Suman	Database, backend development
17BCI0084	Ishita Gupta	Frontend design and scripting
17BCI0113	Namit Nathwani	Detectors, backend development
17BCI0186	Njokosi J Kawunju	Frontend scripting, backend routing

9 Conclusion

In this project, we address the practical importance of using keystroke dynamics as a biometric for authenticating. Keystroke dynamics is the process of analysing the way users type by monitoring keyboard inputs and authenticating them based on habitual patterns in their typing rhythm. We review the current state of keystroke dynamics and present some techniques to validate the user depending on his typing pattern.

We argue that although the use of a behavioural trait (rather than a physiological characteristic) as a sign of identity has inherent limitations when implemented in conjunction with traditional schemes, keystroke dynamics allows for the design of more robust authentication systems than traditional password-based alternatives alone

The future of biometric technologies is promising. Biometric devices and applications continue to grow worldwide. Several factors will push the growth of biometric technologies. A major inhibitor of the growth of biometrics has been the cost to implement them. Moreover, increased accuracy rates will play a big part in the acceptance of biometric technologies. Keyboard Dynamics, being one of the cheapest forms of biometric, has great scope. In this paper, an effort has been taken to give the existing approaches, security and challenges in keystroke dynamics to motivate the research to further come with more novel ideas.

9.1 Future Work

Our tool is the most complete keystroke dynamics based authenticator but there are certain components we need to improve upon. These improvements will require time to implement as each of them require specialized care and cannot be generalized so that "one size fits all".

1. **Integration with password managers:** Since users are moving towards using password managers we need to develop tighter integration with the most popular password managers so that users do not face login issues when using password managers. Currently, our system will not allow users to login using a password manager.
2. **Anonymization of the data:** We can add auto-anonymization of the data so that user's data cannot be exploited by any third party. It will enable users to use the site without worrying about their data being linked to them.
3. **Adding hardware security key-based login:** We can add a third factor for authentication without hindering the login process. Hardware security keys have been proven to be robust and unbeatable until now. We can add it to further enhance the security of our authenticator.
4. **Adding Windows Hello Integration:** Most of the end-users use a Windows-powered machine. Windows Hello is one of Windows' latest login features which has been built according to FIDO rules. Windows Hello has been proven to be unbeaten till now and runs as a native app on Windows. It will reduce the resources used for running our authenticator and also increase security at the same time.
5. **Add more detectors:** We can add more keystroke detectors to increase options for customizability and allow administrators to tweak them according to their liking.

We believe that adding the above features in the future will allow our product to be utilized in the industry very easily. It will also strengthen its position as an alternative to 2FA.

10 References

[1] Bergadano, Francesco, et al. “User Authentication through Keystroke Dynamics.” ACM Transactions on Information and System Security, vol. 5, no. 4, 2002, pp. 367–397., DOI:10.1145/581271.581272.

[2] Giot, Romain, et al. “GREYC Keystroke: A Benchmark for Keystroke Dynamics Biometric Systems.” 2009 IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems, 2009, DOI:10.1109/btas.2009.5339051.

[3] Killourhy, Kevin S., and Roy A. Maxion. “Comparing Anomaly-Detection Algorithms for Keystroke Dynamics.” 2009 IEEE/IFIP International Conference on Dependable Systems &amp; Networks, 2009, DOI:10.1109/dsn.2009.5270346.

[4] Monrose, Fabian, and Aviel D. Rubin. “Keystroke Dynamics as a Biometric for Authentication.” Future Generation Computer Systems, vol. 16, no. 4, 2000, pp. 351–359., DOI:10.1016/s0167-739x(99)00059-x.

[5] Karnan, M., et al., “Biometric Personal Authentication Using Keystroke Dynamics: A Review.” Applied Soft Computing, vol. 11, no. 2, 2011, pp. 1565–1573., DOI:10.1016/j.asoc.2010.08.003.

[6] Bours, Patrick. “Continuous Keystroke Dynamics: A Different Perspective towards Biometric Evaluation.” Information Security Technical Report, vol. 17, no. 1-2, 2012, pp. 36–43., DOI:10.1016/j.istr.2012.02.001.

[7] Mhenni, Abir, et al. “Double Serial Adaptation Mechanism for Keystroke Dynamics Authentication Based on a Single Password.” Computers &amp; Security, vol. 83, 2019, pp. 151–166., DOI:10.1016/j.cose.2019.02.002.

[8] Alsultan, A., Warwick, K., & Wei, H. (2017). Non-conventional keystroke dynamics for user authentication. Pattern Recognition Letters, 89, 53-59.

[9] Krishnamoorthy, S., Rueda, L., Saad, S., & Elmiligi, H. (2018, May). Identification of user behavioural biometrics for authentication using keystroke dynamics and machine learning. In Proceedings of the 2018 2nd International Conference on Biometric Engineering and Applications (pp. 50-57).

[10] Lin, C. H., Liu, J. C., & Lee, K. Y. (2018). On neural networks for biometric authentication based on keystroke dynamics. Sensors and Materials, 30(3), 385-396.

[11] Çeker, H., & Upadhyaya, S. (2016, September). User authentication with keystroke dynamics in long-text data. In 2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS) (pp. 1-6). IEEE.

[12] Kochegurova, E. A., Gorokhova, E. S., & Mozgaleva, A. I. (2017, January). Development of the keystroke dynamics recognition system. In Journal of Physics: Conference Series (Vol. 803, No. 1, p. 012073).

[13] Raul, N., Shankarmani, R., & Joshi, P. (2020). A Comprehensive Review of Keystroke Dynamics-Based Authentication Mechanism. In International Conference on Innovative Computing and Communications (pp. 149-162). Springer, Singapore.

[14] Huang, J., Hou, D., & Schuckers, S. (2017, February). A practical evaluation of free-text keystroke dynamics. In 2017 IEEE International Conference on Identity, Security and Behavior Analysis (ISBA) (pp. 1-8). IEEE.

[15] Zhang, J., Tan, X., Wang, X., Yan, A., & Qin, Z. (2018). T2FA: Transparent Two-Factor Authentication. IEEE Access, 6, 32677–32686

[16] Švogor and T. Kišasondi, "Two-factor authentication using EEG augmented passwords," Proceedings of the ITI 2012 34th International Conference on Information Technology Interfaces, Cavtat, 2012, pp. 373-378, DOI: 10.2498/iti.2012.0441.

[17] Nath, Asoke & Mondal, Tanushree. (2016). Issues and Challenges in Two Factor Authentication Algorithms. International Journal of Latest Trends in Engineering and Technology(IJLTET). 6. 318-327.