*Asian Journal of Research in Computer Science*

***4(4): 1-26, 2019; Article no.AJRCOS.53624***
***ISSN: 2581-8260***

# Keystroke Dynamics Algorithm for Securing Web-based Password Driven Systems

**Michael Boakye Osei[1*], Enoch Opanin Gyamfi[2] and Mohammed Okoe Alhassan[3]**

*[1]School of Computer Science and Engineering, Nanjing University of Science and Technology, 210094, China.*
*[2]School of Software Engineering, University of Electronic Science and Technology of China, Chengdu, China.*
*[3]School of Mechanical Engineering, Nanjing University of Science and Technology, 210094, China.*

***Authors' contributions***

*This work was carried out in collaboration among all authors. Author MBO designed the study, system diagrams, experimental setups, data collection and analysis and wrote the abstract, introduction and conclusion, as well as the first draft of the manuscript and analyses of the study. Author EOG designed and implemented the proposed model. Author MOA managed the literature searches and did the proof reading. All authors read and approved the final manuscript*

***Article Information***

DOI: 10.9734/AJRCOS/2019/v4i430119
Editor(s):
(1) Dr. Sasikumar Gurumoorthy, Professor, Department of Computer Science and Systems Engineering, Sree Vidyanikethan Engineering College, Tirupati, Andhra Pradesh, India.
Reviewers:
(1) Jackson Akpojaro, University of Africa Toru-Orua, Nigeria.
(2) Pasupuleti Venkata Siva Kumar, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering &Technology, India.
(3) Samer I. Mohamed, October University of Modern Science and Arts MSA, Egypt.
Complete Peer review History: http://www.sdiarticle4.com/review-history/53624

***Original Research Article***

***Received 25 October 2019***
***Accepted 29 December 2019***
***Published 04 January 2020***

## ABSTRACT

Web security is a critical aspect for many web-based applications, along its research track, keystroke dynamics techniques have attracted broad interests due to their high efficiency in security. In this paper, the aim was to come out with a keystroke login system that overcomes the typical challenges associated with keystroke dynamics and improves on password security but with focus on irritability nature of keystroke dynamics based systems. Specifically, we proposed two stages user matching method, training/enrolment phase of users and authenticating registered users with previously stored data. Furthermore, the proposed algorithm added dwell, flight times and multiplied by the locate time to get the upper and lower bounds. Moreover, the uniform

**Corresponding author: E-mail: michael.ob@njust.edu.cn;*

differences between the bound timings were calculated to further enhance security. Experimental results show that the proposed keystroke dynamics approach used in augmenting password security emerged to be superior as compared to existing customary distance metrics.

*Keywords: Keystroke dynamics; web security; user credentials; flight time.*

## 1. INTRODUCTION

The universally accepted use of information system and computers has made humanity to carry out their daily activities with less effort. Advancement in information technologies for few decades now has made possible improvements in network performance, reliability and accessibility as well as reducing operating costs by adapting to these efficient technologies [1,2]. Efficient security measures are being sought now to secure computer resources against unauthorized access through fraudulent and masquerading in web applications. Online based applications are developed to allow individuals who have access to internet and web browser to access the content of a particular web application without any platform dependent problems. These systems are usually designed consisting of two main components thus, the front-tier application and a back-end database. The authentication credentials need to be secured properly to reduce system vulnerabilities which could be exploited by illegitimate users, as claimed by [3, 4].

Authentication is an essential element in networks and computer security and will continue be a vital part of a reliable system for granting authorization to corporate information. There are various methods currently used in authenticating system users. The most commonly used is the password. Passwords are appropriate as they are easily employed in software and require no dedicated hardware. Users are also at ease with their use. On the other hand, passwords also come with many flaws. Users frequently share passwords, fail to recall passwords, and select weak passwords characters that can be learned and mimic by an imposter [5,6].

Authenticating users to web application systems becomes a problem due to threats to computer and internet security [7]. The traditional measures used to safeguard credentials of legitimate users of a system such as passwords and pins are no longer much reliable as an efficient authentication approach to some extent [8]. Hence a new secured methodology that must be probed into is biometric techniques [9]. As compared to password and pin methods, biometrics technologies make use of physiological and behavioural traits to identify legitimate users to a system [10,11,12]. Biometric authentication system is appropriate and most secured authentication method, because it is practically impossible to be borrowed, stolen, or forged. Biometrics use unique personal traits to identify and authenticate users into a system whether being a physical or behavioural feature [13,14,15].

The biometric technique appropriate for the purpose of this study is keystroke dynamics. Biometrics based on typing of the user does not require any extra hardware other than keyboard. Keystroke biometric is less expensive to implement, more distributed and more unobtrusive than other biometrics [16]. Gathering of data needs keyboard and simple software utilizing JavaScript. It is relatively cheap investment than other biometrics like fingerprint and retinal scan. It's very easy to replicate the collected data if hardware is not available.

Keystroke dynamics doesn't depend upon the location of the client as we can gather the data from anywhere using Internet. Keystrokes collection software can be distributed via client-server methodology. No specialized preparation is required for keystroke as it's a daily activity. It is relatively unobtrusive technique as contrasted to retinal scan where we have to put some part of the body in front of a special retina scan device [17]. Keystroke dynamics can be considered as secure framework even if an illegitimate user knows the user login credentials [17,18].

However, there is no consistency in Keystroke analysis mechanism like other biometrics which last fairly long period of time. Keystroke analysis biometrics comes with irregularities in the typing style of users due to; utilizing single hand for entering the user credentials, and casual typing [6,19]. The design of computer keyboards also adds some difference in typing style. The posture can lead to change in the keystroke as it's easy to enter the password by sitting rather than standing. Keystroke biometrics can be an irritating technique if a user has to enter the same string repeatedly trying to login accurately into his/her account [3,20].

Gradually, attention is moving towards biometric methods as unique validation and verification measure and as a more secured approach to verify users to a system [21,22]. Biometrics, specifically keystroke dynamics is not in to eliminate the usefulness of passwords and pins but to augment their efficiency as security methods [23,24]. The aim of this study is to determine the effectiveness of keystroke dynamics that identifies unique patterns in the typing behaviour of users used together with passwords as authentication measure to solve the various authentication issues in computer web-based applications. To achieve this enhanced security using keystroke dynamics, this study proposes an algorithm that ensures less irritation of users (i.e. less False Rejection Rate and False Acceptance Rate) but still maintains maximum security.

## 2. RELATED WORKS

In a study Monrose and Rubin utilized Keystroke dynamic biometrics as an authentication technique [1]. They built up a toolkit utilizing C++ for examination of data based on user keystroke patterns which is a computer desktop application. The toolkit was composed of adjusting the x-view library routines, and serves as a front-end to their primary verification engine. The toolkit was useful in diagnosing framework conduct and can create graphical response for both MATLAB and Gnuplot frameworks. They addressed the practical relevance of utilizing keystroke analysis as a biometric feature for verifying access to workstations. They surveyed the present conditions of keystroke analysis and present classification methods on basis of template matching and Bayesian probability models. Their designed toolkit was used to simulate the efficiency of using keystroke analysis as an authentication measure.

Another study utilized Keystroke analysis as a biometric for authentication to predict secured password for users to be used as login credentials. Their methodology empowers the creation of long-term hardened password that can be tried for login purposes or utilized for encryption of documents, entry access to a virtual private system [4,25]. Furthermore, their approach naturally adjusts to progressive changes in a client's keystroke timings while keeping up the same enhanced password over numerous logins, for use in document encryption or different applications requiring for user password. Their approach was mainly to generate password for clients based on their typing behaviour to be used in other systems and the keystroke dynamic metric used was continuous keystroke technique.

Vinayak [26] on user authentication used advanced keystroke analysis. The study indicates that everybody needs to authenticate himself on his computer before using it, or even before using different applications like email. Most of the times, the adopted authentication procedure is the use of a classical couple of login and password. This method no longer provides consistent safety measures because passwords are prone to shoulder surfing and passwords can also be hacked. In this present study concept of keystroke dynamics is introduced to eradicate above said problems. This method is based on the assumption that every person types in a unique manner. He indicates that the advantage of introducing the concept of keystroke dynamics as compared to conventional system is that even the unauthenticated user cracks the password, he will be denied access as this method is based on the typing pattern of the user. In his research FAR error is 0% and FRR is also minimized and accuracy level has reached to a higher level but did not solve the irritability nature of keystroke driven authentication systems.

Seham et al. [27] proposed a method that was based on calculating the flight time and dwell time. They indicated an important point that the identification of users for authentication on computer systems is vital. They claim that keystroke dynamics is a biometric measurement in terms of keystroke press duration and keystroke latency. In their study, it was indicated that several problems are arisen like the similarity between users and identification accuracy. In their paper, they propose innovative model that can help to solve the problem of similar user by classifying user's data based on a membership function but did not consider much on the irritability nature of keystroke driven authentication systems. Also, they employ sequence alignment as a way of pattern discovery from the user's typing behaviour. Experiments were conducted to evaluate accuracy of the proposed model. Their results show high performance compared to standard classifiers in terms of accuracy and precision.

This paper elaborates on the effectiveness of augmenting passwords security by keystroke dynamics in web applications. This study specifically adapts static keystroke dynamic

techniques to support password security to verify users during login sessions in web applications. An algorithm is proposed to enhance the effectiveness of adopting keystroke dynamics as an additional security in web based system by combining the strengths of both Euclidean distance and Manhattan distance metrics. Biometric innovation can take care of issues associated with various login systems, yet numerous biometrics have failed to fulfill the criteria of low equipment prerequisites, minimal effort and non-obtrusiveness.

Keystroke dynamics, then again, is a procedure that overcomes large portions of the challenges different techniques, both customary and new, cannot address. Therefore, keystroke dynamics is a hopeful new answer for an already exiting question: “By what means would we verify clients on web applications?” [10,26].

## 3. KEYSTROKE DYNAMICS

The study purposed to ensure less irritability (i.e. decreasing the chances of where legitimate users of keystroke dynamic systems are denied access many times they try accessing their account) but still maintains needed security over the intended system. Experimental research design is adapted for this exploration to investigate its assertions. Experimental design is the process of planning a study to meet specified objectives. The prototype model was adapted in designing the proposed system because it permitted more flexibility for reconsideration of the system functionalities. Further, this model made it easy to discover mistakes associated with the system functionalities and features at the early stages of the system design. Therefore, the development of the proposed keystroke algorithm to supplement passwords authentication in web based system was done along the structures of prototyping techniques, hence, it followed an experimental research design.

### 3.1 Study Population, Sample and Sampling Techniques

The sample is given by:

$$n = \frac{N}{1 + Ne^2}$$

Where N = known population size, [28].

*e* = alpha level, that is, *e* = 0.2 if the confidence interval is 80%. So the population of 1283, using 0.80 confidence interval, the sample size is calculated as:

$$n = \frac{1283}{1 + 1283(0.2)^2} = 24.5315 \cong 25$$

Thus, in all, and through a convenient sampling technique, an approximate total of 25 volunteers were added to the sample size of this study and who participated in the study. The study agreed that this group of individuals would be able to reflect the best information needed for the overall performance of the proposed system.

### 3.2 Data Collection Procedures

Data for analysis were collected through the proposed algorithm as the users interacts with the system.

Due to the sensitive nature of keystroke logging, participants were first educated on the intended exercise for them to be informed with the purpose about the overview of the experiment and what exactly the data was being logged for. The initial stage of the data collection involved collecting user typing data samples using a structured text (i.e. username and password). The writing samples that were used in this part of the data collection were a collection of 5 instances of every participant user credentials. A large amount of typing data needed to be recorded in order to create good reference profiles. However, this required users to type their credentials for a significant amount of time. This paper hoped to make the experience more bearable by having interesting/humorous writing samples. The typing samples totaled at least 8 characters forming the password and at least 6 characters forming the username. The data logged were: key that was pressed, time the key was depressed, and time the key was released. The time recorded was the amount of time passed since the start of the application and the unit of measurement was in milliseconds. The data structures used to store this data was a multidimensional 2 dimensional array tables. The key-code of every character was determine by the embedded JavaScript as when the participants invokes the keys that form their user credentials by appending the keystroke timings of each character typed.

Once a user finished typing the 5 instances, all the data that was logged is processed and first inserted into a MySQL database and then an acknowledgement is made to notify that a successful writing sample had just been captured

but before the data is stored it goes through series of process. If an error occurs anywhere in the process, the error message will prompt the user to follow the expected rules set about the user credentials in order to proceed. Because of the limitation of time, this project only collected data from Latin characters such as A-Z and the “space”, and “@”. The difference between uppercase and lowercase is assumed to be handled by the multi mixtures in each key. All other unsupported characters (e.g. Backspace key and delete key) would be removed from the raw data to avoid recording keystroke timings of most outliers. The 25 volunteers submitted a complete 5 instances of their typing samples which were the data that was used to feed and evaluate the proposed system.

Collection of data through the proposed algorithm can be also categorized into two main procedures the Data Preprocessing Stage and the Data Feature Extraction Stage. In the Data Preprocessing Stage, unwanted keystroke timings were removed from the main list keystroke timings that were used in the next stage Data Feature Extraction Stage. With the Data Preprocessing Stage, the proposed algorithm is designed to remove outliers in a form of unwanted keystrokes timings recorded during data acquisition from users. It further describes the actual parameter used to extract the final keystrokes timings that is used as the main benchmark for user authentication. These outliers removed at the Data Preprocessing Stage, are keystroke timings of special keys which are the backspace, delete, enter, shift and tab keys. First, in excluding some unnecessary key characters typed, the individual key codes of these three key characters were included in the code for exclusion of their keystroke timings.

## 4. SYSTEM DESIGN

The Authentication module consists of a sign up and login pages where the sign up page is linked with the login page (Fig. 1). The sign up allows users of the system to create their accounts where their keystrokes latencies are determined and stored for the enhancement of the password security.

### 4.1 System Architecture

The proposed system deployed for the purposes of this study, consist of two main parts: the training and the testing phases. The training part collects training keystrokes from users and updates them to the server. It also accesses the database in the server to populate the keyed in username and password credential for particular users. Data collection in both training (sign-up) and testing (Login) phase is performed in client's browsers by embedded JavaScript while PHP is used to handle database queries in the server side. To perform these tasks, a JavaScript application was developed to stimulate the data collection. It also formats the raw data of typing behavior in browsers and then processes it further before storing it in the database. This JavaScript can be embedded to any web applications authentication system, in order to apply the result of this project in reality.

On another hand, in order to simulate the real collection data phase, the application is able to help users to append data in separate times rather than one session. In addition, the advantage of handling raw keystroke data in the client is the avoidance of performance problems such as delaying time when transferring data to the server or saving more resources in the server. Moreover, the data of training keystrokes and user's models are stored in the server side by MySQL database.

Hence, data processing is also performed in the server while the recognition is done through the communication between clients and server to transfer the testing data and the result of recognition tests back to clients from the server. Also, the proposed system learns during the training section of the application by allowing the users to enter their credentials five separate instances where the needed extracted values from the keystroke events are compared to each other in order to determine accurate range keystroke timings for each user, within which each user has to maintain his/her typing behaviour to be successfully authenticated together with other parameters. There are five main components that aided the design of the proposed system which are the database (MySQL), server-side scripting language (CSS, JavaScript and AJAX), server-side scripting language (PHP), server simulator (XAMPP) and the web browser.

This paper proposes a new distance metric combining both Euclidean distance and Manhattan distance such that one complements the other. As a result, the proposed distance metric combines the benefits of both Euclidean and Manhattan distance metrics while overcoming their limitations when used individually. As it turns out, this new distance metric also has a nice statistical interpretation

and also the new technique suppress the influence of outliers for improved performance.

## 4.2 Flow Chart of the Proposed System

In the authentication (template matching) process we take the Euclidean distance and Manhattan distance between the template and the query time series. During template matching a threshold value is set upon which to accept or reject a user pattern. If the query is within the threshold the system accepts the pattern otherwise it rejects the data as fraud and signals as imposter. It is a natural requirement to have a user adaptive threshold so we set the initial threshold to be the largest distance (Upper-Bound) and the minimum distance (Lower-Bound) of a user's record from the user's uniform distance as computed in data processing discussed in subsequent sections of this study, this ensures less irritability of users but still maintains maximum security.

The flow chart that resulted when this system was implemented is as follows and indicated in the Fig. 2.

## 5. SYSTEM ANALYSIS AND EVALUATION

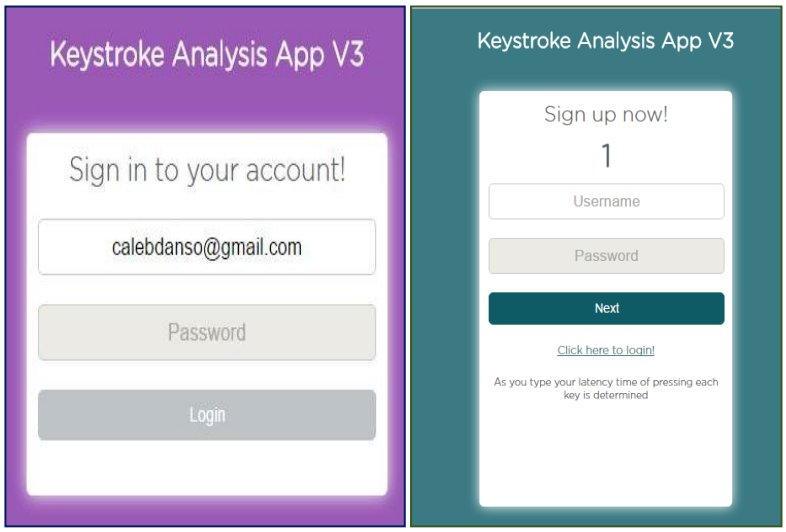
This section illustrates results of the study by evaluating the capabilities of the proposed authentication system. This paper conceptualized the idea of upper bounds and lower bounds in keystroke authentication system. The proposed system tends to narrow the frames between the upper bound and lower bound keystroke latencies with the sole purpose to very much increase the security. In addition to the introduction of the upper and lower bounds in Key-Press events of the proposed procedure, an anomaly detection distance metrics, The Euclidean and the Manhattan was applied together, not separately, to calculate and record the average distance between the stored bounds for all the Key-Press events. The features of the proposed algorithm are divided into two major segments, the training phase and the testing phase.

## 5.1 Training Phase of the Proposed Algorithm

The training phase serves as the enrolment phase, which instructs and guides the users of the system to go through a five-series signup process. The training phase involves gathering the information of users (usernames and passwords) before login process which is the next stage, testing phase. The phase also generates a unique signature reference, called bound, which acts as a template to authenticate user. The bound templates are kept in record of a database to be used later in the testing stage, in authenticating users. The purpose of this training phase to the system is that, it enables the system to learn and store the individual bounds (in milliseconds) of any Key-Press events initiated by the users, except for the initial key character for the username or password, which will always be recorded as a zero bound. For example, a username '*ernestina*' will always have an initiated bound for the key character '***e****'* to be zero, which according to the algorithm, is used for special purposes described in subsequent paragraphs. Again, the algorithm uses the stored bounds timings to calculate a constant distance value which is also an additional security feature used in identifying a legitimate users and allowing their accesses into the system.

### 5.1.1 Calculating of bound timings

A bound in the context of the proposed algorithm, is therefore the time, in milliseconds, taken by a user to locate, initiate Key-Down and Key-Release/Key-Up events for a particular key on the keyboard. According to the Fig. 3, which describes how the algorithm calculates the bound timing values, there are three timings recorded by the algorithm for the calculation of its bound timing values. These three timings are the Locate time, keystroke duration/dwell time and the keystroke latency/flight time. The Locate time has a sole purpose of one linking to its consecutive key so that key characters can form a string of meaningful or human understandable words, sentences and paragraphs. In simple explanation, typical real word scenarios were made to clarify the algorithm. A male, user 'A' also referred with the name 'Michael', and with the username '*michael*' and password '*osei@22*', is used in subsequent scenarios. In addition, a female, user 'B' referred with the name 'Ernestina', and with the username '*ernestina*' and password '*tinao@22*', will be used in some other instances. Therefore, if the locate times of each user are taken into consideration, it will definitely differentiate user 'B' from user 'A' in typing a word like '*michael'.* In this, user 'A' might be slower or faster than, but never equal to a user 'B', who is also typing the same word using the same keyboard, User 'A' might be faster or slower in locating the keys on the keyboard when compared to user 'B'.




**(a)** **(b)**

**Fig. 1. module of the proposed system (a) Login page module, (b) Signup page module**

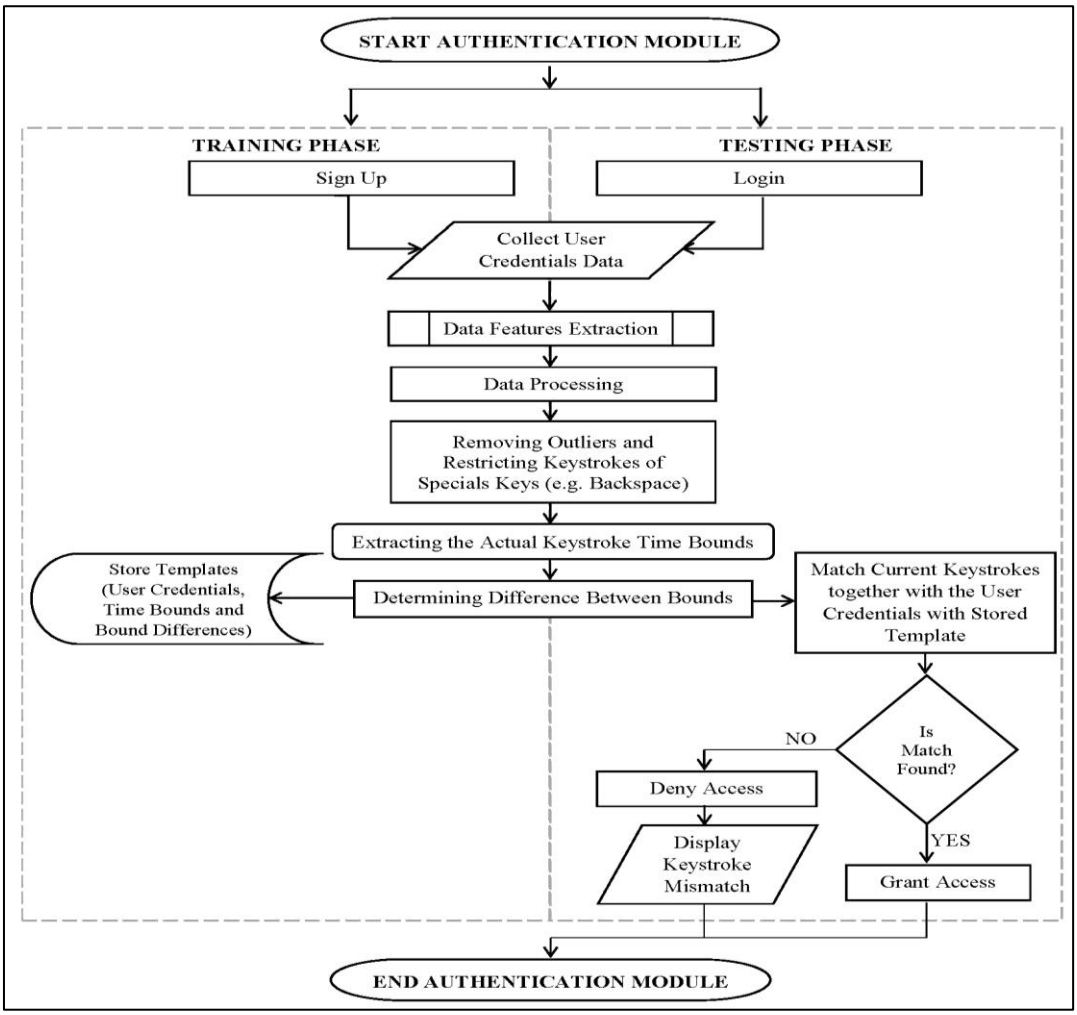



**Fig. 2. Dataflow diagram of the proposed system**

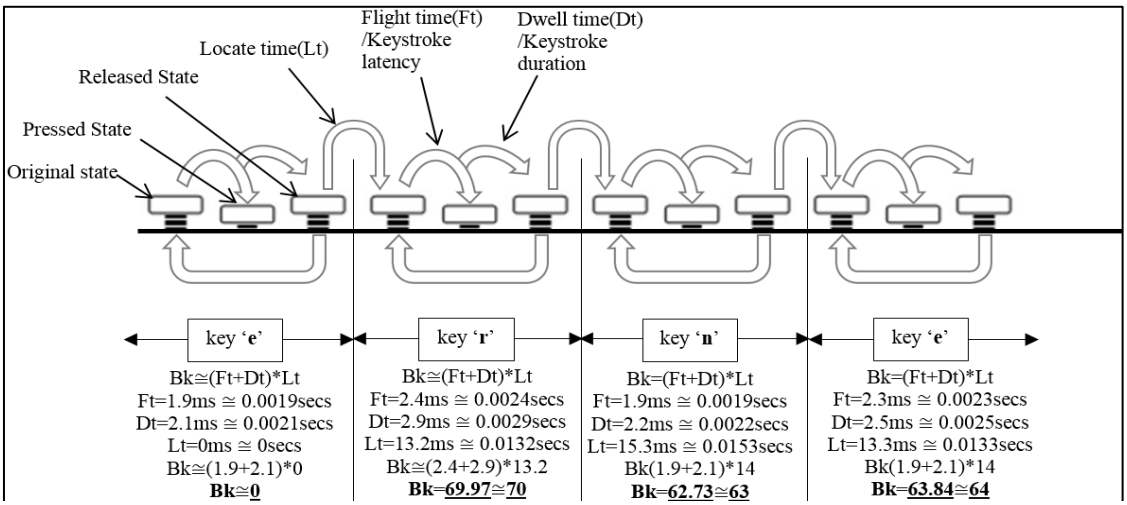



**Fig. 3. Bounds calculation using the three key states**

In simple terms, the bound is a recorded measure of time, whenever the key is pressed and ends when the pressure is released from the key. The bound of a key (Bk), according to proposed algorithm is calculated by adding the Flight time (Ft) to the Dwell time (Dt) and then multiplying it with Locate time (Lt).

Bound of a key ≅ (Flight Time + Dwell time) x Locate time

According to the algorithm, the relationship between all the three states of a key is that the pressed state serves as the intermediary between the original state and the release state of the key. This simply means that, a key in an original state enters the pressed state on any Key-Down event for that particular key, if and only if the key immediately enters its released state. Also, the next relationship between the three states of a key is that, the states are cyclic in nature.

The Fig. 3 describes how the locate times for each key pressed by a user 'B', the female user named Ernestina, subjected the calculation of bound times for these keys in the algorithm.

After the bound for each key character is calculated, they are stored in an array. The multidimensional array of characters and their respective bounds were recorded for each key. The username of the user 'B', which is '*ernestina',* consisted of 9 characters and therefore the algorithm will calculate a bound time for all the 9 characters. The algorithm therefore generates 9 bound times. The 9 characters are stored in a 1-Dimensional (1-D) array (userN[]) while the 9 bound times are stored in another 1-Dimensional (1-D) array (bound[]). Then, a new 2-Dimensional (2-D) array (userNBound[][]) was created by combining the two 1-Dimensional arrays userN[] and bound[] as represented in the Fig. 4.

### 5.1.2 Determining the lower bounds and upper bounds of the proposed algorithm

At this point, it is finally derived from the training phase that, the system stores the bounds (in milliseconds) for all the Key-Press events in a 2D-Dimensional table array, assigning independent indexes to these bound timings. Now, it has to be pointed out that, the algorithm, determines the lower and upper bound timings that are stored into a database, based on all the five signup processes. These lower and upper bounds are used by the proposed system to accept legitimate users and reject imposters. Therefore, the primary aim for calculating the bound timings for all the key characters stroked on the keyboard is to find these highest and lowest bounds of all the Key-Press events.

As shown in Fig. 5, the bound timings for key characters are stored as elements within a 1-D array. These bound timings were recorded for the user 'B', who went through all the five series in her signup process.

The algorithm makes these sorting arrangements to make it easier to extract the final upper bond and lower bond of the keystroke timings that is stored as thresholds for testing the system. From Fig. 6 these timings are then arranged in order of

magnitude from lowest to highest to determine the lower and upper bounds for all the stored bound timings.

The 2-Dimensional table array are arranged in ascending order of magnitude (using the bound timings), from the lowest to the highest values and stored in a new 1-Dimensional array (see Fig. 7).

However, the highest value is realized by the system as the highest bound for all the Key-Pressed Events. The highest bound forms the upper bound (in milliseconds). This upper and lower bounds limits the user. In that sense, the upper bound is a limit for the next phase (testing phase) to which the user, when finished with the training phase, cannot instantiate a bound above it (upper bound). This is also referred to as stored upper bound. Similarly, the lowest bound recorded at the training phase for a particular user, serves as the lower bound limit (in milliseconds), in the next phase (testing phase), below which the user cannot initiate a bound afar. This is also referred to as stored lower bound (see Fig. 8).

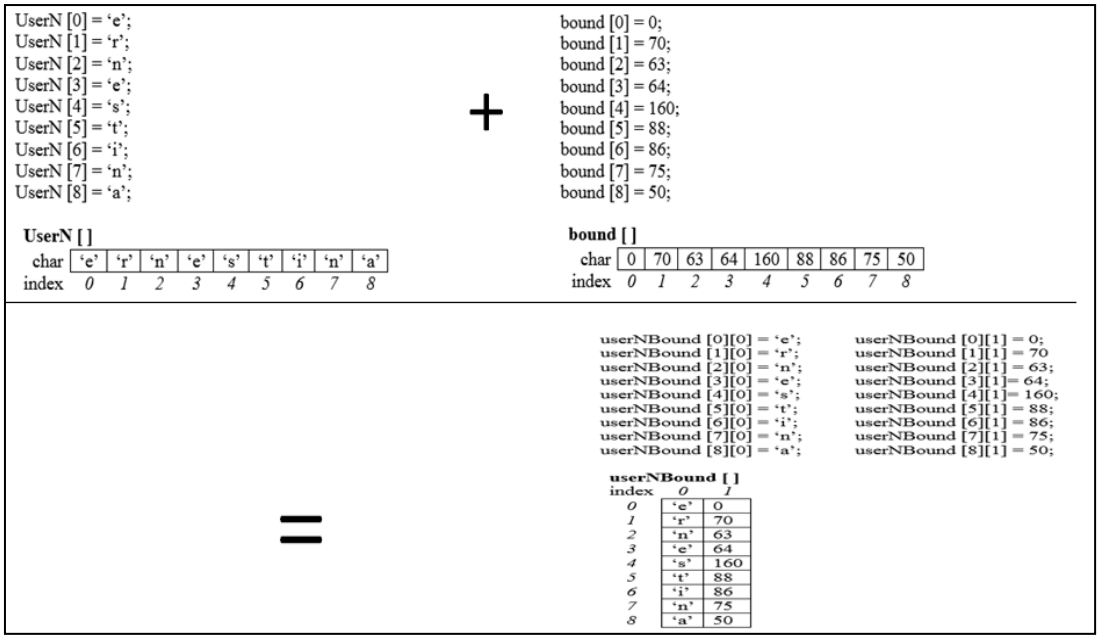



**Fig. 4. Code snippet representation of 2-D array, userNBound[][] for storing bound timings**

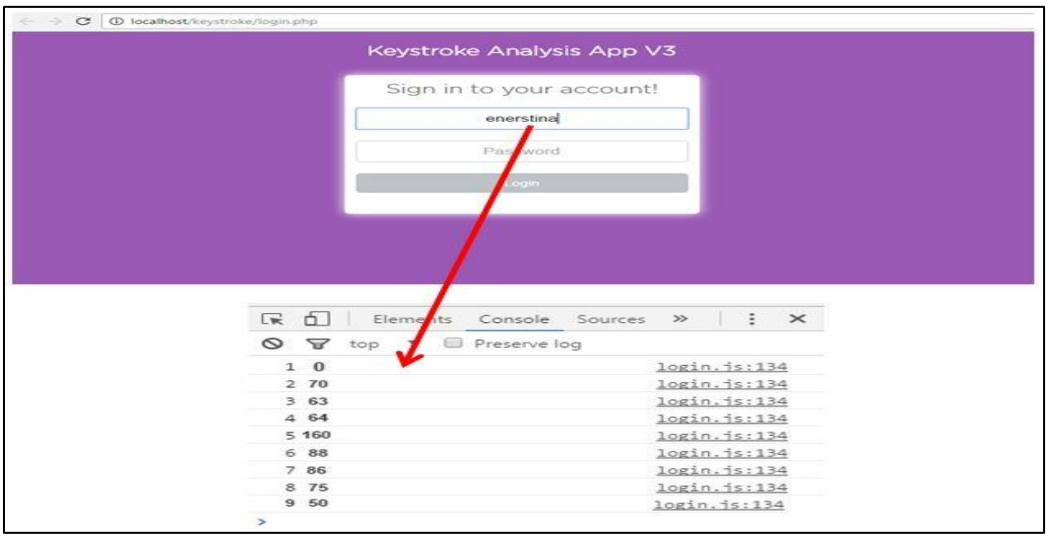



**Fig. 5. Snipet of the first instance of Keystroke Timings for Username at Five-Series SignUp Process (username is 'ernestina')**

| Char | 1st Process | 2nd Process | 3rd Process | 4th Process | 5th Process |
|---|---|---|---|---|---|
| e | 0 | 0 | 0 | 0 | 0 |
| n | 70 | 74 | 72 | 73 | 71 |
| e | 63 | 54 | 60 | 61 | 63 |
| r | 64 | 62 | 61 | 63 | 62 |
| s | 160 | 149 | 160 | 154 | 158 |
| t | 88 | 80 | 82 | 86 | 88 |
| i | 86 | 73 | 82 | 83 | 85 |
| n | 75 | 62 | 71 | 72 | 73 |
| a | 50 | 74 | 60 | 58 | 54 |

**Fig. 6. Keystroke timings for username at five-series sign up processes (Username is 'ernestina')**

| | 1st Process | 2nd Process | 3rd Process | 4th Process | 5th Process |
|---|---|---|---|---|---|
| | e 0 | e 0 | e 0 | e 0 | e 0 |
| | a 50 | e 54 | e 60 | a 58 | a 54 |
| | e 63 | r 62 | a 60 | e 61 | r 62 |
| | r 64 | n 62 | r 61 | r 63 | e 63 |
| | n 70 | i 73 | n 71 | n 72 | n 71 |
| | n 75 | n 74 | n 72 | n 73 | n 73 |
| | i 86 | a 74 | t 82 | i 83 | i 85 |
| | t 88 | t 80 | i 82 | t 86 | t 88 |
| | s 160 | s 149 | s 160 | s 154 | s 158 |
| LowerBounds[] | 50 | 54 | 60 | 58 | 54 |
| UpperBounds[] | 160 | 149 | 149 | 154 | 158 |

**Fig. 7. Keystroke Timings for username at five-series sign-up processes arranged in ascending order of magnitude (Username is 'ernestina')**

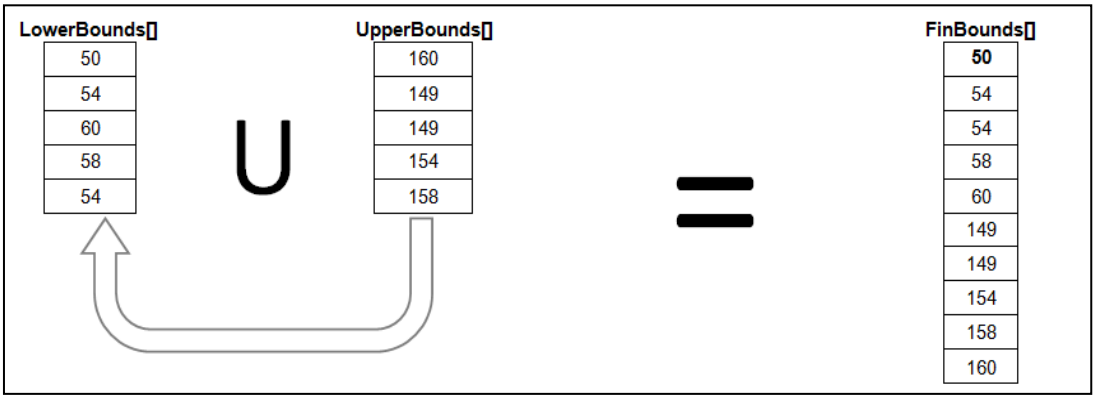



**Fig. 8. Lower and upper bounds of user 'B', stored and merged into a 1-D array, to find the final lower and upper bounds**

### 5.1.3 Determining the uniform difference between bounds

The differences in the bounds were determined using the Euclidean and Manhattan (Taxicab/city-block) distance metric approaches. The algorithm choses to calculate the distance between the bounds either with Euclidean distance metric or Manhattan distance measure, based on the locate time of the user typing pattern. If the algorithm records in the FinBounds[] array that any locate time is greater than or equal to one seconds which is approximately 1000 milliseconds and also any

locate time is lesser than or equal to 0.5 seconds, which is approximately 500 milliseconds, then the Manhattan distance measure is used.

This is because, from literature [22], it was revealed that the Euclidean distance metric is good for measuring real distances on a plane within relatively small areas with distinctively single intervals. But the Euclidean distance metric cannot be used to determine how shorter or longer a distance is, one point to another while moving at a given speed. Manhattan distance measure is therefore a proposed distance measure that was used by the algorithm to determine the shorter distance value to be stored for a slower typing pattern of a user.

In determining the differences between bounds using the Euclidean distance measure, the bounds are treated as points on a Cartesian coordinate. If for each distance between two consecutive bounds is to be measured on the Cartesian coordinate, the bounds are perceived as points, for instance, P and Q, where P is the beginning bound and the Q is the end bound, under concentration. Therefore, the Euclidean distance measures the length of the line segment connecting points P and Q on the plane of the Cartesian coordinate. If P = ($P_1$, $P_2$, $P_3$, ..., $P_n$) and Q = ($Q_1$, $Q_2$, $Q_3$ ..., $Q_n$) are two points in Euclidean *n*-space, then the distance ($E_d$) from P to Q, or from Q to P is given by the Pythagorean formula:

$$\mathrm{E}_d(\mathrm{P},\mathrm{Q}) = \mathrm{E}_d\,(\mathrm{Q},\mathrm{P}) = \sqrt{(P_1 - Q_1)^2 + (P_2 - Q_2)^2 + \ldots\ldots\ldots + (P_n - Q_n)^2}$$

$$\mathrm{E}_d(\mathrm{P},\mathrm{Q}) = \mathrm{E}_d\,(\mathrm{Q},\mathrm{P}) = \sum_{i=1}^{n}(P_i - Q_i)^2.$$

where n=FinBounds[].length=10

The position of points (bounds) in a Euclidean *n*-space is a Euclidean vector. So, P and Q are Euclidean vectors, and having two vectors, each, represented as *X* and *Y*. So for point P on the Euclidean *n*-space, its *x* and *y* values are derived from the Cartesian coordinate as well as the point Q. In this sense, point P is solely denoted as *P (X, Y)*. Similarly, point Q can be represented as *Q (X, Y).* Taking point P to be the source or starting point of the distance to be covered, it is further represented as *P* ($X_0$, $Y_0$), where $X_0$ and $Y_0$ are initial vector coordinates in the Cartesian plane. Likewise, taking point Q to be the destination or end point of the distance to covered, it can be further denoted as *Q* ($X_1$, $Y_1$), where $X_1$ and $Y_1$ are finishing vector coordinates in the Cartesian plane. In an n-Euclidean dimension, the distance between two points on the real plane (Cartesian coordinates) is the absolute value of their numerical difference. Thus if *P* ($X_0$, $Y_0$) and *Q* ($X_1$, $Y_1$) are two (source and destination) points on the real line, then the distance between them is given by:

$$\mathrm{E}_d(\mathrm{P},\mathrm{Q}) = \mathrm{E}_d\,(\mathrm{Q},\mathrm{P}) = \sqrt{(X_1 - X_0)^2 + (Y_1 - Y_0)^2}$$

$$\mathrm{E}_d(\mathrm{P},\mathrm{Q}) = \mathrm{E}_d\,(\mathrm{Q},\mathrm{P}) = \sum_{i=0}^{n-1}(X_{(i+1)} - X_i)^2 + (Y_{(i+1)} - Y_i)^2$$

where n=FinBounds[].length=10

The formula can be simply explained that, the first (source) vectors on the Cartesian coordinates $X_0$ and $Y_0$, are subtracted from the second (destination) vectors on the Cartesian coordinates, $X_0$ and $Y_0$, to be able to get the distance covered from $X_0$ to $X_1$ and $Y_0$ and $Y_1$, respectively. In explaining these, a real world experiment was conducted whereby the user 'B' typed her username as '*ernestina*'. As seen from previous paragraphs and subsections, after going through series of processing, the algorithm finally recorded the 10 bound timings as 50 ms, 54 ms, 54 ms, 58 ms, 60ms, 149 ms, 149 ms, 154 ms, 158 ms, 160 ms and stored them in the array FinBounds[]. The algorithm will therefore use the Euclidian distance metric to measure the uniform differences between the bounds, because it was obvious that during its (the algorithm) processing, no locate time recorded was above 1000 millisecond which is approximately 1 second.

The Fig. 9, further denotes how the algorithm arranges the element within the 1-D array, FinBounds[] on a Cartesian plane. After the algorithm places the bounds as points on a Cartesian plane, it uses each point's coordinates derived from its corresponding x-axis and y-axis to calculate for the distance between the element in the 1-D array FinBounds[]. The diagonal distance between points is calculated using the Euclidean distance metric which is denoted by $E_d$ on the figure. In calculating the distance, the coordinates are noticed from the horizontal (y-axis) and vertical (x-axis) perspective. On the y-axis, the coordinates represent the bound elements stored in the array while on the x-axis, the coordinates represent the 10 indexes (from 0 to 9) of the array at which bound elements are stored.

From the figure, distances that need are calculated by the proposed algorithm are, from ponts $P_3$ (4, 58) to $P_4$ = (5, 60), then to $P_5$ = (6, 149). From $P_6$ (7, 149) through $P_7$ (8, 154) to $P_8$ (9, 158) also needs to be calculated. Lastly, distances between the points $P_8$ (9, 158) to $P_9$ (10, 160) also needs to be calculated. When all these distances are calculated for all the 10 points, they are added together to result in the uniform difference between bounds of a user typing pattern. Table 1 further illustrates this concept of the algorithm.

In the second case, when the algorithm decides to use the Manhattan distance measurement metric to determine the differences between bounds, each distance between two consecutive bounds are calculated along the right/left and top/down directions. A comparison is then made on all the possible distances between the two (source and destination) points to choose the shortest to be stored. The Manhattan Distance is the distance between two points measured on the grid-like Cartesian plane layout, which can determine the shortest time a user can use in striking the first key and also striking of the last key on the keyboard. Typically, the Manhattan distance measures the differences between the elements within the sorted array FinBounds[], which provides 1 value, each, for *x* and *y* at each of the 10 alternative points on the plane. Therefore, just like with the Euclidian distance, for the 10-point distance measure using the Manhattan metric, if $X_i(s)$ and $Y_i(s)$ are the *X* and *Y* coordinates of the points P and Q, and if the starting point $X_i$ and the ending point $Y_i$ are the X and Y coordinates of points P and Q, in the Cartesian state, the formular is given by:

$$\mathrm{M}_d(\mathrm{P},\mathrm{Q}) = \mathrm{M}_d(\mathrm{Q},\mathrm{P}) = |P_1 - Q_1| + |P_2 - Q_2| + |P_3 - Q_3| + \ldots\ldots\ldots + |(P_n - Q_n)|$$

$$\mathrm{M}_d(\mathrm{P},\mathrm{Q}) = \mathrm{M}_d(\mathrm{Q},\mathrm{P}) = \sum_{i=1}^{n} |(P_i - Q_i)|$$

where n=FinBounds[].length=10

Manhattan metric takes the sum of the absolute values of the differences in the coordinates of the points. For example, if point P with coordinates $X_0$ and $Y_0$, which serves as the starting coordinates and point Q, with ending coordinate $X_1$ and $Y_1$, then the Manhattan distance ($M_d$) between coordinates $X_i$ and $Y_i$ is primarily given as

$$\mathrm{M}_d(\mathrm{P},\mathrm{Q}) = \mathrm{M}_d(\mathrm{Q},\mathrm{P}) = |X_1 - X_0| + |Y_1 - Y_0|$$

$$\mathrm{E}_d(\mathrm{P},\mathrm{Q}) = \mathrm{E}_d(\mathrm{Q},\mathrm{P}) = \sum_{i=0}^{n-1} |X_{(i+1)} - X_i| + |Y_{(i+1)} - Y_i|$$

where n=FinBounds[].length=10

In explaining these, a real world experiment was conducted whereby the user 'A' typed his username as '*michael*'. The algorithm classified this user as a slow typist as it (the algorithm) recorded some locate times of the user to be more than 1 second which is approximately 1000 milliseconds. In other words, the difference between the upper bound and the lower bound was greater than 0.5 seconds or 500 milliseconds. After going through series of processing, the algorithm finally recorded the 10 bound timings as 251ms, 354ms, 458ms, 464ms, 658ms, 668ms, 668ms, 670ms, 690ms and 1150ms which were stored in the array FinBounds[] as follows in Fig. 10.

In calculating the Manhattan distance, a standard heuristic for a square grid is taken into consideration. In other words, the algorithm looks at all the difference in bounds in relation to the Cartesian plane and finds the minimum difference for moving from one point to another adjacent point. In simple case, on a square grid, the algorithm can move in a minimum of four directions in each given distance between points. In picking the distance, the best path chosen to be the difference between the bounds is the one with the lowest value between the two adjacent points.

In the Fig. 11 and as shown on the Cartesian plane, there are 10 points (from $P_0$ to $P_1$). Each point on the Cartesian plane is represented by an *X* and *Y* coordinate values. For example, point $P_0$ is represented on the Cartesian plane as $P_0$ (1, 251) and $P_1$ represented as $P_1$ (2, 354), $P_2$ as $P_2$ (3, 458) and so on till last point $P_9$, also represented as $P_9$ (10, 1150). However, because the algorithm decided to use Manhattan metric of distance to measure the differences in the bounds on the Cartesian plane, several minor points are also considered. Take for instance, if the algorithm wanted to measure the difference in milliseconds, between the first bound point $P_0$ and the second bound point $P_1$, it (the algorithm) will consider all the two paths ($M_{d1\,x\,1}$ and $M_{d1\,x\,2}$) from $P_0$ to $P_1$, in which one ($M_{d1\,x\,1}$), path which is represented as $P_0 \rightarrow P_{0\text{-}up} \rightarrow P_{0\text{-}upRight} \rightarrow P_1$ and another path ($M_{d1\,x\,2}$) which is also represented as $P_0 \rightarrow P_{0\text{-}down} \rightarrow P_{0\text{-}downRight} \rightarrow P_{0\text{-}downRightUp} \rightarrow P_1$.

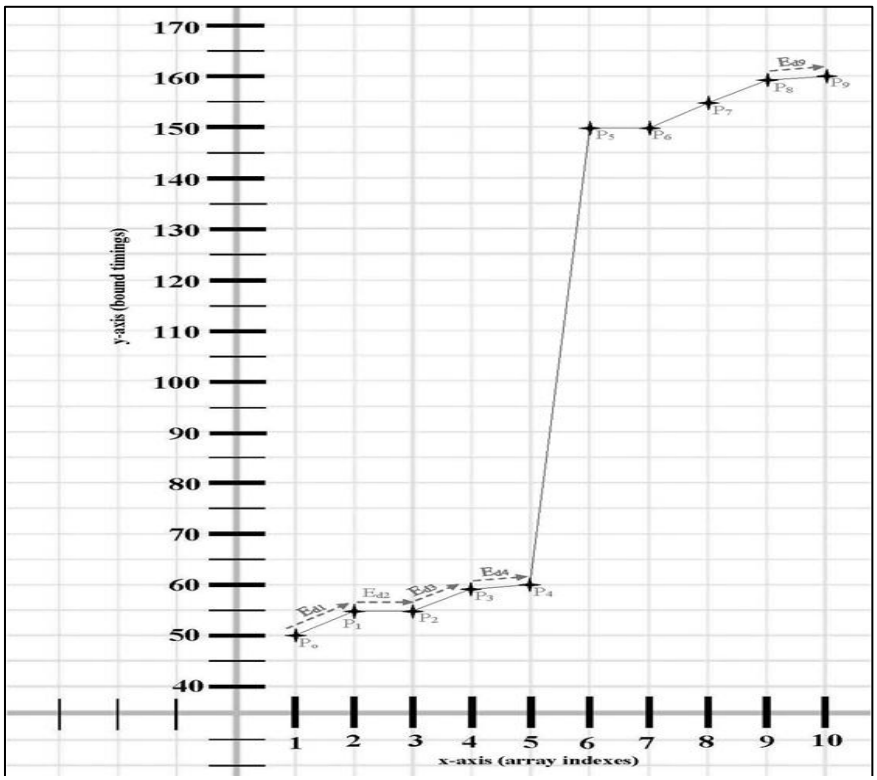



**Fig. 9. Array representation in determining the uniform difference between bounds on a Cartesian plane (Euclidean Distance)**

**Table 1. Uniform difference between bounds by euclidean metric**

| Cartesian plane points | Array representation of points | Distance by euclidean metric |
|---|---|---|
| $P_0$ to $P_1$ | $P_0$ = (1, 50) to $P_1$ = (2, 54) | 4.123 |
| $P_1$ to $P_2$ | $P_1$ = (2, 54) to $P_2$ = (3, 54) | 1 |
| $P_2$ to $P_3$ | $P_2$ = (3, 54) to $P_3$ = (4, 58) | 4.123 |
| $P_3$ to $P_4$ | $P_3$ = (4, 58) to $P_4$ = (5, 60) | 2.236 |
| $P_4$ to $P_5$ | $P_4$ = (5, 60) to $P_5$ = (6, 149) | 89.001 |
| $P_5$ to $P_6$ | $P_5$ = (6, 149) to $P_6$ = (7, 149) | 1 |
| $P_6$ to $P_7$ | $P_6$ = (7, 149) to $P_7$ = (8, 154) | 5.011 |
| $P_7$ to $P_8$ | $P_7$ = (8, 154) to $P_8$ = (9, 158) | 4.123 |
| $P_8$ to $P_9$ | $P_8$ = (9, 158) to $P_9$ = (10, 160) | 2.236 |
| **Uniform difference between bounds (Summation) =** | | **112.853** |

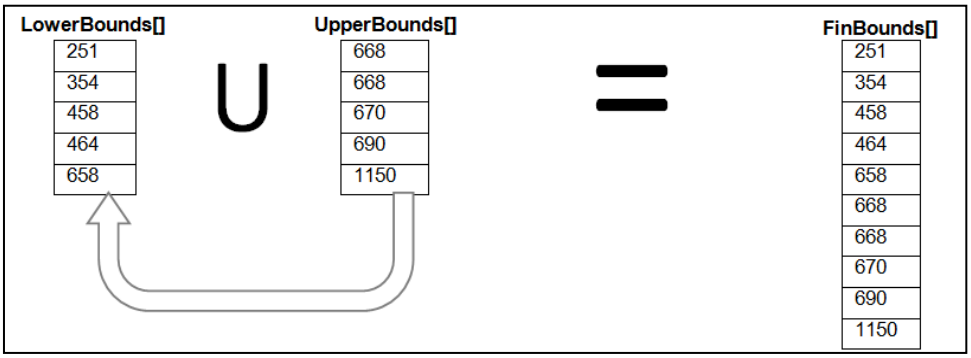



**Fig. 10. Lower and Upper Bounds of user 'A', stored and joined into a 1-D array, to find the final Lower and Upper Bounds**

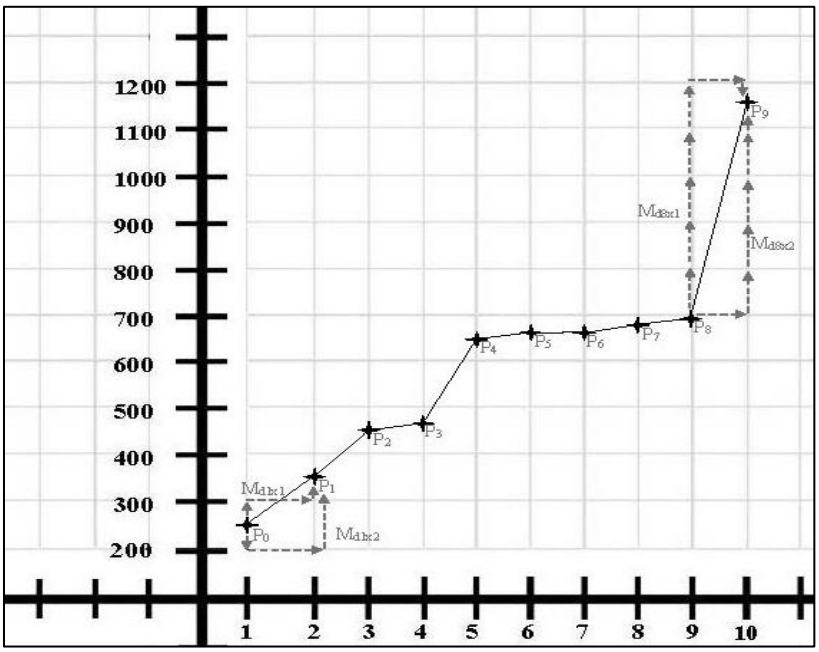



**Fig. 11. Array representation in determining the uniform difference between bounds on a Cartesian plane (Manhattan Distance)**

| Points | ($M_{dn\ x\ 1}$) | ($M_{dn\ x\ 2}$) |
|---|---|---|
| $P_0$ to $P_1$ | $P_0 \rightarrow P_{0\text{-up}} \rightarrow P_{0\text{-upRight}} \rightarrow P_1$ | $P_0 \rightarrow P_{0\text{-down}} \rightarrow P_{0\text{-downRight}} \rightarrow P_{0\text{-downRightUp}} \rightarrow P_1$ |
| $P_1$ to $P_2$ | $P_1 \rightarrow P_{1\text{-up}} \rightarrow P_{1\text{-upRight}} \rightarrow P_2$ | $P_1 \rightarrow P_{1\text{-down}} \rightarrow P_{1\text{-downRight}} \rightarrow P_{1\text{-downRightUp}} \rightarrow P_2$ |
| $P_2$ to $P_3$ | $P_2 \rightarrow P_{2\text{-up}} \rightarrow P_{2\text{-upRight}} \rightarrow P_3$ | $P_2 \rightarrow P_{1\text{-down}} \rightarrow P_{1\text{-downRight}} \rightarrow P_3$ |
| $P_3$ to $P_4$ | $P_3 \rightarrow P_{3\text{-up}} \rightarrow P_{3\text{-upUp}} \rightarrow P_{2\text{-upUpRight}} \rightarrow P_4$ | $P_3 \rightarrow P_{1\text{-down}} \rightarrow P_{1\text{-downRight}} \rightarrow P_{1\text{-downRightUp}} \rightarrow P_{1\text{-downRightUpUp}} \rightarrow P_4$ |
| $P_4$ to $P_5$ | $P_4 \rightarrow P_{4\text{-up}} \rightarrow P_{4\text{-upRight}} \rightarrow P_5$ | $P_4 \rightarrow P_{4\text{-down}} \rightarrow P_{4\text{-downRight}} \rightarrow P_5$ |
| $P_5$ to $P_6$ | $P_5 \rightarrow P_{5\text{-up}} \rightarrow P_{5\text{-upRight}} \rightarrow P_6$ | $P_5 \rightarrow P_{5\text{-down}} \rightarrow P_{5\text{-downRight}} \rightarrow P_6$ |
| $P_6$ to $P_7$ | $P_6 \rightarrow P_{6\text{-up}} \rightarrow P_{6\text{-upRight}} \rightarrow P_7$ | $P_6 \rightarrow P_{6\text{-down}} \rightarrow P_{6\text{-downRight}} \rightarrow P_7$ |
| $P_7$ to $P_8$ | $P_7 \rightarrow P_{7\text{-up}} \rightarrow P_{7\text{-upRight}} \rightarrow P_8$ | $P_7 \rightarrow P_{7\text{-down}} \rightarrow P_{7\text{-downRight}} \rightarrow P_8$ |
| $P_8$ to $P_9$ | $P_8 \rightarrow P_{8\text{-up}} \rightarrow P_{8\text{-upUp}} \rightarrow P_{8\text{-upUpUp}} \rightarrow P_{8\text{-upUpUpUp}} \rightarrow P_{8\text{-upUpUpUpUp}} \rightarrow P_{8\text{-upUpUpUpUpUp}} \rightarrow P_{8\text{-upUpUpUpUpUpRight}} \rightarrow P_{8\text{-upUpUpUpUpUpRightDown}} \rightarrow P_9$. | $P_8 \rightarrow P_{8\text{-up}} \rightarrow P_{8\text{-upRight}} \rightarrow P_{8\text{-upRightUp}} \rightarrow P_{8\text{-upRightUpUp}} \rightarrow P_{8\text{-upRightUpUpUp}} \rightarrow P_{8\text{-upRightUpUpUpUp}} \rightarrow P_9$. |

**Fig. 12. Determining the shortest path by manhattan distance**

**Table 2. Uniform difference between bounds by manhattan metric**

| Points | ($M_{dn\ x\ 1}$) | ($M_{dn\ x\ 2}$) | Relationship | Chosen |
|---|---|---|---|---|
| $P_0$ to $P_1$ | 104 | 206 | $(M_{d1\ x\ 1}) < (M_{d1\ x\ 2})$ | $(M_{d1\ x\ 1}) = 104$ |
| $P_1$ to $P_2$ | 105 | 213 | $(M_{d2\ x\ 1}) < (M_{d2\ x\ 2})$ | $(M_{d2\ x\ 1}) = 105$ |
| $P_2$ to $P_3$ | 79 | 123 | $(M_{d3\ x\ 1}) < (M_{d3\ x\ 2})$ | $(M_{d3\ x\ 1}) = 79$ |
| $P_3$ to $P_4$ | 195 | 259 | $(M_{d4\ x\ 1}) < (M_{d4\ x\ 2})$ | $(M_{d4\ x\ 1}) = 195$ |
| $P_4$ to $P_5$ | 75 | 127 | $(M_{d5\ x\ 1}) < (M_{d5\ x\ 2})$ | $(M_{d5\ x\ 1}) = 75$ |
| $P_5$ to $P_6$ | 65 | 137 | $(M_{d6\ x\ 1}) < (M_{d6\ x\ 2})$ | $(M_{d6\ x\ 1}) = 65$ |
| $P_6$ to $P_7$ | 63 | 139 | $(M_{d7\ x\ 1}) < (M_{d7\ x\ 2})$ | $(M_{d7\ x\ 1}) = 63$ |
| $P_7$ to $P_8$ | 41 | 161 | $(M_{d8\ x\ 1}) < (M_{d8\ x\ 2})$ | $(M_{d8\ x\ 1}) = 41$ |
| $P_8$ to $P_9$ | 561 | 461 | $(M_{d9\ x\ 1}) > (M_{d9\ x\ 2})$ | $(M_{d9\ x\ 2}) = 461$ |
| **Uniform difference between bounds (Summation) =** | | | | **1188** |

After the Manhattan distance have been computed using the coordinate of the identified points the in the Fig. 12, Table 2 also gives the distance values for the paths and shows the relationship between the two alternate paths so as to make a decision as to which path should be chosen.

#### 5.1.4 The proposed algorithm for the training phase of keystroke dynamic

Summarizing the above processes involved training phase of the keystroke dynamics, it was revealed that the algorithm generates the bound timings from three main recorded values which are the dwell time, flight time and the locate time of a key pressed. After generating the bound timings of key characters pressed, they are then arranged in ascending order of magnitude in an array to find the lower and upper bounds of the timings.

The lowest value, inserted into the first index of the sorted array is deemed the lower bound whereas the highest value inserted into the last index area of the sorted array is stored as the upper bound. The differences in all the stored bounds is calculated and added together by using either Euclidian distance metric approach or the Manhattan distance measure approach. A decision is taken by the algorithm if it learns the user is slow/inexpert typist it adopts to use Manhattan distance metric else if it learns the user is fast typist it uses Euclidian distance metric in calculating the uniform distance.

### 5.2 Testing Phase of the Proposed Algorithm

The procedural features of the proposed algorithm at the testing phase has significant similarities and minor differences with the training phase, hence, in this section, although much has been said about the similarities, substantial portion of the section will greatly concentrate of the differences too. In the testing phase of the proposed system, a user is presented with the login interface. A user, after successfully going through the training phase, can successfully login into the testing phase. In the testing phase, the user has to meet certain credentials before he/she is allowed into the system. First, the user has to correctly supply into the login interface, his/her username and password that was used to sign-up into the system. Second, the user typing pattern must record a lower bound (hitherto referred to as lower bound threshold) equal to or greater than the stored lower bound for that particular user. Similarly, the user must record an upper bound (hitherto referred to as upper bound threshold), equal to or less than the stored upper bound for that particular user. Back to the scenario for instance, the male user ‘A’, who has successfully sign-up into the system with the username ‘michael’, has a stored lower bound of 251 milliseconds and a stored upper bound of 1150 milliseconds. The user ‘A’, can therefore be allowed to login into the system if and only if he records a lower boundary latency, called lower bound threshold, greater than or equal to stored lower bound and also, records an upper bound, upper bound threshold, that is less than or equal to the stored upper bound for that particular user (i.e. upper bound ≥ threshold ≥ lower bound). When this user ‘A’ was permitted to partake in the testing phase, so as to determine if he could be verified into the system, he recorded a lower bound of 359 milliseconds and an upper bound of 1057 milliseconds. This is illustrated in Fig. 13.

In the testing phase, what is important is the threshold. The algorithm treats the recorded thresholds in two descriptions. The first threshold is the bound thresholds while the second threshold is the distance threshold. The algorithm associates the bound thresholds with the stores lower bound and the stored upper bound, whereas the distance threshold is associated with the uniform difference between the bounds calculated using either the Euclidean or the Manhattan distance metric. The threshold is the recorded bound timing at the testing phase at which it becomes reasonable certain that a keystroke sample matches a particular reference template. Typically, the value of a threshold is never exact and hence the algorithm chooses a measure of similarity at which a keystroke sample may be declared identical to the reference template.

It is therefore important to note here that, in the testing phase, the users are grouped into two main classes. These two main classes of users also have to do with either the recorded thresholds are met by the user or not. Users of the system are classified as an individual who is legitimate or a user who is illegal. A legitimate user, as the name implies, is a user who has successfully gone through and completed all the five-series sign-up process at the training phase of the proposed system. This is also referred to as legal or valid user. A user is believed to be legitimate if he/she provides an acceptable value for his/her thresholds. An illegal user, also as the

name infers, is a user who has not successfully gone through the training phase of the proposed system. A user is alleged to be illegal if he/she does not provide an acceptable value for his/her thresholds. To the system, a user's first time visit to the system is supposed to be at the training phase; else that particular user is deemed as illegitimate user. If a user's first time visit to the system is at the testing phase, then that user is classified as an illegal user. This is hereafter referred to as illegitimate user or imposter.

Also, in the testing phase, the calculated bound timings and its correspondent uniform difference were set as the reference threshold at which a user's typing sample must match. The algorithmic procedures in this phase is similar to the training phase because, at this phase, the propose system likewise calculates the bound timings of the sample typing pattern provided, arranged these bound timings in order of magnitude, from lowest to highest using selection sort in an array, and sets the lower bound timing of the typing sample to be the lowest bound in the sorted array as well as the upper bound timing to be the highest bound also in the same sorted array. The only difference here is that; the array was 1-D instead of 2-D, as with the case of the training phase. 1-D array was very significant in decreasing the time and cost complexity used in processing the algorithm.

In simple terms, the algorithm for testing phase ran faster than that of the training phase because of the 1-D array used. The algorithm then goes further by calculating the uniform differences between the bounds also using either Euclidian of Manhattan distance measure. Likewise, the training phase, in choosing the distance metric to be used for a particular user, the locate time (typing speed was considered). The uniform differences between the bound timings of a slow typist with highest locate times above 500 milliseconds were calculated using the Euclidian Distance metric while that of a fast typist with highest locate times below 500 milliseconds was calculated using the Manhattan Distance metric.

These values for the lower bound, upper bound and the uniform difference calculated at the testing phase is then compared with the lower bound, upper bound and uniform difference thresholds calculated from the training phase. There were three major if-condition set that controls the state of the tested sample, whether as a legitimate user or an imposter. If the lower bound at the testing phase is greater than the lower bound of the training phase, and if the upper bound at the testing phase is lesser than the upper bound of the training phase and if the uniform difference at the testing phase is lesser than the uniform difference of the training phase, then the tested sample is nearer the referenced template recorded in the database and hence the user is deemed legitimate and can therefore log into the system.

#### 5.2.1 Calculating of bound timings in the proposed algorithm

Still using in subsequent scenarios, the processing of the algorithm on the typing pattern of the male, user 'A' who supplied into the system, the username '*michael*' and the password '*osei@22*', and also, the typing template of the female, user 'B' with the username '*ernestina*' and password '*tinao@22*', this paragraph explains how the testing phase was designed. Just with the training phase, the algorithm multiplied the dwell time (keystroke duration) and the flight time (keystroke latency) with the locate times when calculating the bounds of a user's typing pattern. Again, the locate time is initially set to zero to denote a start of a new process within the system. This simply means that, the locate time of a user's typing pattern is reset to zero when he/he moves the computer's focus to the password textarea, which is an entirely a new process, to type his/her password. As indicated earlier, the bound of a key (Bk), according to proposed algorithm is calculated by adding the Flight time (Ft) to the Dwell time (Dt) and then multiplying it with Locate time (Lt). Furthermore, for simplicity sake, the tables below summarize the bound times calculated from the typing template of the user 'A' and user 'B'.

#### 5.2.2 Determining the lower bounds and upper bounds of the proposed algorithm

After the bound for each key character is calculated, the next step for the algorithm at this testing phase is to store these bounds in a 1-Dimendioanl array. Hence, the array of characters is not stored by the algorithm for further processing. The username of the user 'B', which is 'ernestina', consisted of 9 characters and therefore the algorithm will calculate a bound time for all the 9 characters. The algorithm therefore generates 9 bound times. The 9 bound times are stored in a 1-Dimensional (1-D) array

(userBound[]). The code snippet for storing bound timings in the 1-D array userBound[], is represented.

Fig. 14 shows how the proposed algorithm stores the bounds (in milliseconds) for the user 'B' who initiated key-press events. These bound timings are stored in a 1D-Dimensional array, assigning independent indexes to these bound timings. Just like in the training phase, the primary purpose for calculating the bound timings of all key-press events of a user is to determine the lower and upper bounds for the user's typing sample. These lower and upper bounds are used by the proposed system to accept legitimate users and reject imposters. To determine the lower and upper bounds for all the stored bound timings, the recorded bound timings for the user's typing pattern are arranged in order of magnitude from the lowest to the highest bound values. The lowest bound value, which will always be '0' is stored for the first character, and in the first index position of the array. As clarified in previous paragraphs, this first bound for the first key character is always zero as, a locate time of zero is always multiplied by the sum value of flight and dwell times. Likewise, the training phase, the algorithm of the testing phase does not consider the bound timing of the first character keyed in the user, when determining the lower and upper bounds. As a result, the actual bound timings for determining the lower and upper bounds of the user is calculated from the bound timing of the second key character, stored in the second index position of the sorted array. Thus, the algorithm always considers the bound timing element stored in the second index of the array. The array will by this time be sorted from lowest to highest, with the lowest value placed at the second index of the array (userBound[1]) and the highest value placed at the last index of array ((userBound[].length)-1). In this testing phase, the lowest bound value is stored by the system as the lower bound for all the Key-Pressed Events. Again, the highest value is stored by the system as the upper bound for all the Key-Pressed Events as illustrated in the Fig. 15.

### 5.2.3 Determining the uniform difference between bounds

Also just like in the training phase, after the bound limits (lower and upper bound) have been set, the differences in the bounds will be determined using the Euclidean and Manhattan (Taxicab/city-block) distance metric approaches. The differences between all recorded bounds was also calculated by the algorithm and stored in a well-structured database, as the Euclidean or Manhattan distance between the bounds. With the distance measurement, the difference between the recorded final bounds, from the lowest final bound to the highest final bound, was calculated and added up to get a rounded single value. It has been established that algorithm was much interested in using the upper and lower bounds in deciding on which user is given access into the system. It also uses the uniform distance as a distance threshold in deciding which user is legitimate or imposter.

The algorithm choses to calculate the distance between the bounds wither with Euclidean distance metric or Manhattan distance measure, based on the locate time of the user typing pattern. Below are the calculated uniform distances for the two users, user 'A' using Manhattan metric and user 'B' using Euclidean metric.

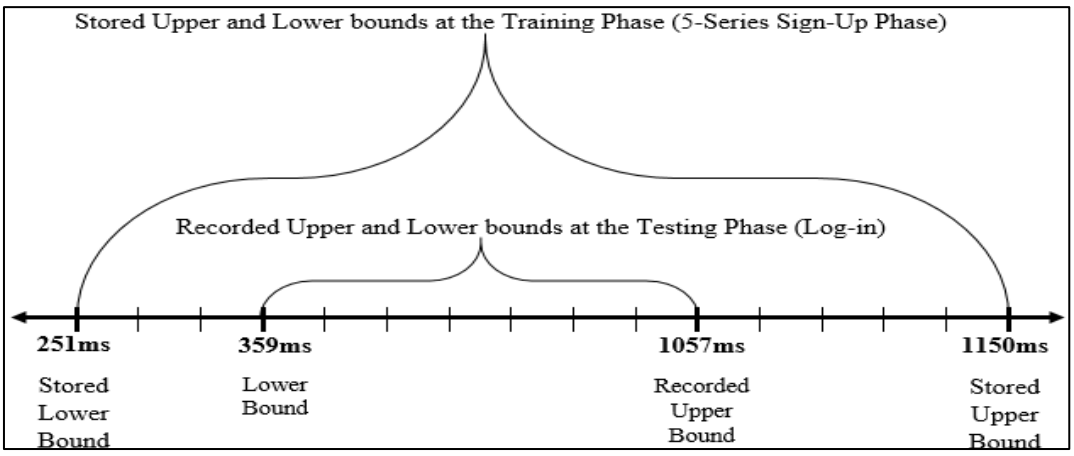



**Fig. 13. Lower and upper bound thresholds of user 'A' in the testing phase**

**Table 3. Recorded bound timings for user 'A'**

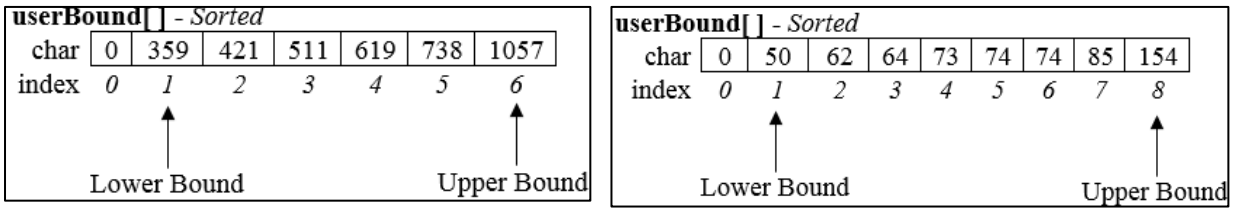
| Key character | Flight time (Ft) | Dwell time (Dt) | Locate time (Lt) | Bk=(Ft+Dt)xLt |
|---|---|---|---|---|
| 'm' | 0 | 0 | 0 | 0 |
| 'i' | 4.2 | 3.8 | 52.6 | 421 |
| 'c' | 4.9 | 5.4 | 60.1 | 619 |
| 'h' | 4.1 | 4.6 | 58.7 | 511 |
| 'a' | 5.9 | 6 | 88.8 | 1057 |
| 'e' | 5.1 | 5.1 | 72.4 | 738 |
| 'l' | 4.4 | 3.9 | 43.2 | 359 |

**Table 4. Recorded bound timings for user 'B'**

| Key Character | Flight time (Ft) | Dwell time (Dt) | Locate time (Lt) | Bk=(Ft+Dt)xLt |
|---|---|---|---|---|
| 'e' | 0 | 0 | 0 | 0 |
| 'n' | 2.2 | 2.9 | 14.3 | 73 |
| 'e' | 2.5 | 2.8 | 11.7 | 62 |
| 'r' | 2.1 | 2.4 | 14.2 | 64 |
| 's' | 4 | 4.9 | 17.3 | 154 |
| 't' | 3.2 | 3.3 | 13.1 | 85 |
| 'i' | 2.7 | 2.5 | 14.3 | 74 |
| 'n' | 2.6 | 2.8 | 13.7 | 74 |
| 'a' | 1.8 | 1.9 | 13.6 | 50 |

| char | 0 | 73 | 62 | 64 | 154 | 85 | 74 | 74 | 50 |
|---|---|---|---|---|---|---|---|---|---|
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

**Fig. 14. Code snippet representation of 1-D array, userBound[], for storing bound timings**




**(a)** **(b)**

**Fig. 15. Sorted array to determine the lower and upper bounds of the typing pattern of (a) user 'A'; (b) user 'B'**

**Table 5. Uniform differences between bound of user 'A' using manhattan distance**

| Points | ($M_{dn \times 1}$) | ($M_{dn \times 2}$) | Relationship | Chosen |
|---|---|---|---|---|
| $P_0$ to $P_1$ | 93 | 113 | ($M_{d1 \times 1}$) < ($M_{d1 \times 2}$) | ($M_{d1 \times 1}$) = 93 |
| $P_1$ to $P_2$ | 132 | 153 | ($M_{d2 \times 1}$) < ($M_{d2 \times 2}$) | ($M_{d2 \times 1}$) = 132 |
| $P_2$ to $P_3$ | 186 | 212 | ($M_{d3 \times 1}$) < ($M_{d3 \times 2}$) | ($M_{d3 \times 1}$) = 186 |
| $P_3$ to $P_4$ | 243 | 275 | ($M_{d4 \times 1}$) < ($M_{d4 \times 2}$) | ($M_{d4 \times 1}$) = 243 |
| $P_4$ to $P_5$ | 404 | 445 | ($M_{d5 \times 1}$) < ($M_{d5 \times 2}$) | ($M_{d5 \times 1}$) = 404 |
| **Uniform difference between bounds (Summation) =** | | | | **1058** |

#### 5.2.4 Determining the legitimacy of the user using logical thresholds

Now, the concept of thresholds in Keystroke dynamics comes into play. These thresholds, as indicated earlier, limit the user and help secure the system by distinguishing between a legitimate user and an imposter. In that sense, the upper bound calculated from the training phase is a limit or threshold for the upper bound

calculated in the testing phase where a user at the testing phase cannot instantiate an upper bound above the threshold.

Similarly, the lower bound calculated from the training phase serves as the lower bound limit or threshold for the lower bound calculated in the testing phase where a user at the testing phase cannot instantiate a lower bound below the threshold.

Finally, the uniform distance determined in the training phase is a limit or threshold for the uniform distance calculated in the testing phase, where a user at the testing phase cannot instantiate a uniform distance far away from the threshold, either greater or lesser than.

The scenario for the two users, with username '*ernestina*' and '*michael*' yielded the following figures. With the user 'A', who is Michael, at the training phase, in registering with the user name '*michael*', he recorded a stored upper bound of 1150ms, a stored lower bound of 251 ms and a uniform difference of 1058 ms between the recorded bounds. He does this at the testing phase of the proposed system.

Therefore, at the testing phase, the threshold expected to be set by the algorithm for this particular users are the aforementioned values, 1188 ms as an upper bound threshold, 251 ms as a lower bound threshold and 1188 ms as a uniform difference threshold in the testing phase. After this user has successfully registered into the system, he can therefore go on to log into his account using his credentials. Also at this phase, it has been recorded from previous paragraphs and from the system that, the user 'A' had a recorded upper and lower bounds of 1022 ms and 289 ms respectively. Also, he recorded a uniform difference of 1150 ms between the recorded bounds in the testing phase. In doing this, the algorithm employed the unification techniques in array manipulations. In other words, the algorithm merged two arrays' numeric indexes, appended two 1-D arrays and sorted the array in order of magnitude.

**Table 6. Uniform differences between bound of user 'b' using euclidian distance**

| Cartesian plane points | Array representation of points | Distance by euclidian metric |
|---|---|---|
| $P_0$ to $P_1$ | $P_0$ = (1, 50) to $P_1$ = (2, 62) | 12.042 |
| $P_1$ to $P_2$ | $P_1$ = (2, 62) to $P_2$ = (3, 64) | 2.236 |
| $P_2$ to $P_3$ | $P_2$ = (3, 64) to $P_3$ = (4, 73) | 9.055 |
| $P_3$ to $P_4$ | $P_3$ = (4, 73) to $P_4$ = (5, 74) | 1.414 |
| $P_4$ to $P_5$ | $P_4$ = (5, 74) to $P_5$ = (6, 74) | 1.000 |
| $P_5$ to $P_6$ | $P_5$ = (6, 74) to $P_6$ = (7, 85) | 11.045 |
| $P_6$ to $P_7$ | $P_6$ = (7, 85) to $P_7$ = (8, 154) | 69.007 |
| $P_7$ | $P_7$ = (8, 154) | |
| **Uniform difference between bounds (Summation) =** | | **105.800** |

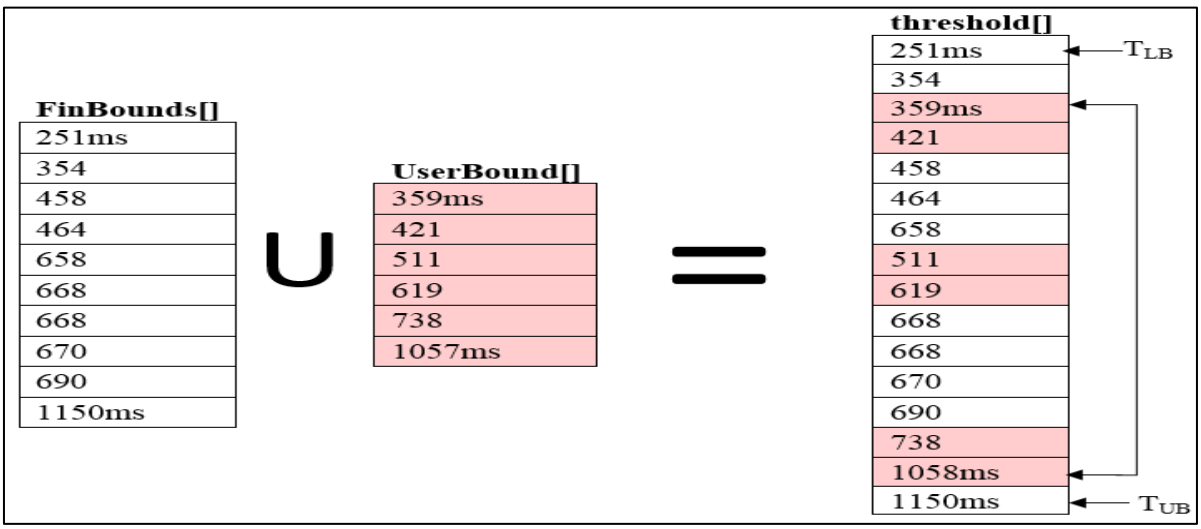



**Fig. 16. Determining thresholds using array for User 'A'**

In this example, given two sorted arrays FinBounds[] and UserBound[] as the input arrays for the user 'A' which can be represented as FinBounds[] = {251, 354, 458, 464, 658, 668, 668, 670, 690, 1150} from the training phase, and UserBound[] = {359, 421, 511, 619, 738, 1057}, from the testing phase, in finding their union set array, threshold[], the program will limelight out the union array as and sort their indexed values as threshold[] = {251, 354, 359, 421, 458, 464, 658, 511, 619, 668, 668, 670, 690, 738, 1058, 1150}. In this union array, the first value considered for making decision is the value at a specific indexed position of the threshold[] that took the value of the first indexed position of the UserBound[], which should not be less than the threshold for lower bound ($T_{LB}$) as indicated in the Fig. 16. The second value considered for making decision is the value at last indexed position of the threshold[] that took the value of the last indexed position of the UserBound[], which should not be greater than the threshold for upper bound ($T_{UB}$) as indicated in the Fig. 16. With this instance, user 'A' was allowed into the system successfully.

Similarly, by means of the user 'B', who is Ernestina, the algorithm at the training phase recorded a stored upper bound of 158 ms, a stored lower bound of 50 ms and a uniform difference of approximately 113 ms between the recorded bounds. Therefore, at the testing phase, the thresholds expected to be set by the algorithm for this particular user are the aforementioned values, 158 ms as an upper bound threshold, 50 ms as a lower bound threshold and 113 ms as a uniform difference threshold in the testing phase.

After this user has successfully registered into the system, he can therefore go on to log into his account using his credentials. At the testing phase, user 'B' recorded 50 ms as a lower bound, 154 ms as an upper bound and approximately 106 ms as a uniform difference of between the recorded bounds. In doing so, the algorithm can also therefore allow user 'B' into the system because, she recorded a lower bound at the testing phase, greater that of the training phase. She also recorded an upper bound at the testing phase, lesser than that of the training phase. Finally, the uniform difference between the bounds, calculated at the testing phase was lesser but closer to the uniform difference between the bounds, calculated and stored at the training phase.

#### 5.2.5 The proposed algorithm for the testing phase of the Keystroke Dynamic

In summary, the proposed algorithm at the testing phase is similar to that of the training phase; however, the differences had to do with the introduction of the concept of thresholds in the keystroke dynamic algorithm. It was discussed that the processes in the algorithm at the testing phase are similar with the algorithm for the training phase in the sense that, it (algorithm in the testing phase) also computes the bound timings from the three main biometric keystroke dynamic features, the dwell time, flight time and the locate time of a key pressed.

The bound timings calculated for each key are then arranged in ascending order of magnitude in an array to find the lower and upper bounds of the timings. The lowest value, inserted into the first index of the sorted array becomes the lower bound whereas the highest value inserted into the last index area of the sorted array becomes as the upper bound. After calculating the bound timings, the proposed further processed the uniform difference between bounds. The algorithm chooses to either use the Euclidian distance metric approach or the Manhattan distance measure approach in computing the differences in all the stored bounds based on a conditions set around the locate time of the user typing pattern.

If the algorithm records any locate time greater than or equal to 0.5 seconds, which is approximately 500 milliseconds, then the Manhattan distance measure is used, however, if the algorithm calculates majority of the locate time of the user to be less than 500 milliseconds which, then it uses the Euclidean distance metric.

The only additional feature of the algorithm at the testing phase was the segment to determining the legitimacy status of the user using thresholds. These thresholds, restricts the user and help secure the system by distinguishing between a legitimate user and an imposter. In that sense, the upper bound calculated from the training phase is a limit or threshold for the upper bound calculated in the testing phase where a user at the testing phase cannot instantiate an upper bound above the threshold.

Similarly, the lower bound calculated from the training phase serves as the lower bound limit or threshold for the lower bound calculated in the testing phase where a user at the testing phase

cannot instantiate a lower bound below the threshold. Finally, the uniform distance determined in the training phase is a limit or threshold for the uniform distance calculated in the testing phase, where a user at the testing phase cannot instantiate a uniform distance far away from the threshold, either greater or lesser than.

### 5.3 Ease of Use from Technical Perspective

This section gathers information on performance indicators that can depict and measure how easy and simple the proposed system was to users. In par with this description, Failure to Enroll Rate (FER) and False Rejection Rate (FRR) were the performance indicators used to quantify how easy the proposed system could be. In must be noted that these indicators are tested in worst case scenarios, meaning more attempts are made to put the system at a weak spot at failing miserably. The system will therefore be classified as annoying or not annoying, easy or not easy to use, if it is able to give a reasonably standard values for its' FERs and FRRs.

Failure to Enroll Rate (FER) is a situation where users will genuinely fail to enroll into the system, a couple of times before they actually successfully enroll into the system. This is used to measure how easy users can get into the keystroke system on the first place. It can be calculated as follows;

Failure to Enroll Rate (FER) = (Number of User having Failed Attempts)/(All Genuine Enrollment attempts) x 100.

For the 25 people initially enrolled in the system, who were numbered from 1st to 25th, most of them successfully login to the system but few had issues with their login attempts. In this case, user 3rd, 9th and 17th failed at their first instance to enroll into the system and hence, 3 users had failed attempts. Consequently, Failure to Enroll Rate (FER) is:

FER = 3/68 x 100 = 4.412% ≅ 4%

This can be interpreted as, for every 100 people enrolled into the system, approximately 4 to 5 users might fail for their first time or not get enrolled at all. This can also imply that, for the 25 people who were enrolled into the system, approximately 1 or 2 users might fail for their first time or not get enrolled at all. Failure to Enroll Rate is therefore obviously low for the proposed system.

False Rejection Rate (FRR) refers to the percentage ratio between falsely denied genuine users against the total number of genuine users accessing the system. Occasionally known as False Nonmatching Rate (FNMR), this is a type I error. In keystroke dynamics, an acquisition problem is a typing mistake which forces the individual to type again the text from scratch. This metric is important for this biometric modality as it measures how annoying the system is to, for a lot of users in keystroke dynamics. Mathematically, FRR is calculated as:

False Rejection Rate (FRR) = (Genuine Logins failing thresholds)/(Total number of Genuine Logins) x 100

In this case, when all the 25 respondents were successfully registered into the system at the training phase, when testing the FRR, each user was friendly instructed to login in five times. Basically, the respondents were numbered from the 1st to the 25th. Out of the 25 respondents, the 2nd, 8th, 14th and 21st users had challenges login in at all five instances.

In this formula, the numerator which is the genuine logins failing thresholds were 8 whilst the denominator which is the total number of genuine logins were 125. In calculating the FRR therefore is;

FRR = 8/125 x 100 = 6.4% ≅ 6%

This can be interpreted as, for every 100 login attempts into the system; approximately 6.4% representing 6 to 7 of these genuine logins might fail to get into the system.

### 5.4 Security from Technical Perspective

This section gathers information on performance indicators that can depict and measure how secured the proposed system was to users. In line with this description, False Acceptance Rate (FAR) was the only biometric performance indicator used to quantify how secured and authentic the proposed system could be.

False Acceptance Rate (FAR) is defined as the percentage ratio between falsely accepted unauthorized users against the total number of imposters accessing the system. Terms such as, False Match Rate (FMR) or type II error refers to

the same meaning. Mathematically, FAR is calculated as:

False Acceptance Rate (FAR) = (Imposter Logins passing thresholds)/(Total number of Imposter Logins) x 100

By means of the FAR test, the system was proved on false acceptance of fraud login attempts. In order to achieve this, all login user credential details were printed out. The users were pre-informed that, their usernames and passwords will be interchanged among themselves to how secured the system was. Basically, the imposters were numbered from the 1st to the 25th so is the numbering of login credentials. In this logic, all usernames and passwords registered into the system was printed out and shared among them.

Out of the 25 imposters, the 24th user managed to enter into the system once. Overall all, out of the 125 imposter attempts, of all the 25 imposters, only 1 succeeded, at the worst case. In this formula, the numerator which is the number of imposter logins passing thresholds was 1 whilst the denominator which is the total number of imposter logins was 125. Therefore, FAR calculated as:

$$FAR = 1/125 \times 100 = 0.8\% \cong 1\%$$

This can be interpreted as, for every 100 imposter attempts into the system, approximately 0.8% representing 0 to 1 of these imposters might succeed to get into the system. This supports the study conducted by [6,11,29] that any biometric enhanced authentication system having low FAR than FRR is of high performance rate.

## 6. RESULTS AND DISCUSSION

This paper was limited to implementing its system in a web-based format, using only QWERTY form of keyboards in input acquisitions, ignoring other keyboard structures like the Dvorak Keyboard. It is therefore recommended that, for a keystroke system to be more robust, algorithm developers should incorporate all the most commonly used keyboard structures in a single system so that the limitation of a keystroke biometric system by keyboard features will be minimized for once.

Furthermore, it must be emphasized that, the training phase is a five-series sign-up process and it is therefore recommended to other algorithm developers who wants to delve into the proposed receptacle, to make the number of series in the training phase, more dynamic so that the administrators of the proposed system can increase or decrease the number of sign-up instances either above or below 5 as with the proposed system. Time restricted the effort by the researcher to incorporate such a dynamic signup feature in the system. If the system's signup process is dynamic enough, the scope of the bounds and the distance values stored to control the access into the system will be widen to further decrease the FER, FRR and FAR toward an absolute zero percentage.

Also, as in the algorithm, after the bound for each key character is calculated, only the bounds are then stored in a 1-Dimendioanl array. Hence, the keyed-in characters are not stored by the algorithm for further processing. It can be recommended that algorithm developers can store the character sets into any data structure in addition to the calculated bound timings of the user's typing sample at both the testing and training phases, so that the algorithm can know which characters at the training and testing phase had similar timings to allow the user, an access into the system.

The keystrokes analysis design was based on bound timing values which helped in determining the uniform difference between them, and which serves as the main criteria together with the user credentials upon which authentication is performed. In light of the sorted concepts, various features were utilized in capturing user profiles, processing these features to obtain the values for the upper-bound, lower-bound timings and uniform distances of each template stored and then implementing these obtained values as the classifier for verification. In doing so, efforts were made to know if a satisfactory level of FER, FRR and FAR of the proposed system will or will not be accomplished. It was therefore concluded that, the proposed system, with an FER of about 4%, an FRR of around 6% and a minimized FAR of approximately 1%, is easy to use, accurate, and secured.

As opposed to other works which concluded that, in order to attain a low FER, FRR and FAR, the three major keystroke features, which are the dwell time (keystroke source duration), flight time (keystroke latency) and the locate time (terminus), should be captured individually from the user's typing pattern, this study concludes

**Table 7. Comparative studies of the proposed and existing methods**

| Study | Data size | Latency | Input repetition | Input freedom | Method | FAR (%) | FRR (%) | FER (%) | User irritability solved |
|---|---|---|---|---|---|---|---|---|---|
| [25] | - | DT, FT, | 1 | No | Euclidean Distance | 0 | 6 | - | No |
| [26] | 50 | DT,FT | 8 | Yes | Enhanced NM&W algorithm | Precision = 90.3 Accuracy = 80 | | | No |
| [30] | 31 | FT | 2 | Yes | Degree of Disorder | 1.99 | 2.42 | - | No |
| [4] | - | DT,FT | 10 | Yes | Euclidean Distance OR Manhattan Distance | 2.46 | 6.62 | - | No |
| [22] | 20 | DT,FT | 20 | Yes | Euclidean Distance | 0 | 35 | - | No |
| | | | 20 | | Mahalanobis distance | 20 | 2.5 | - | No |
| [17] | 30 | DT,FT | Minimum of 2 | Yes | Threshold minimum and maximum limits | 0 | 1 | - | No |
| Proposed Method | 25 | DT,FT,LT | 5 | Yes | Euclidean(Eu) + Manhattan Distance(Mh): (Eu+Mh) | 1 | 6 | 4 | Yes |

that these keystroke features should form a calculable equation whereby the flight time is always added to the dwell time and the summed value multiplied by the locate time of the user's inputting pattern. The output value form this equation is termed the bound timing.

The study also concluded that the calculated bound timings should be stored in a 1-D array to make the algorithm faster in arranging these bound timings from lowest to highest, in terms of value weights. The study also introduced the concept of uniform difference whereby the differences between the lower bound through to the upper bound timings are calculated using the Euclidean or Manhattan distance metric formula. The study also concludes that, if the values for lower, the upper bound timings and the uniform difference are used as a threshold at which a user must pass, before he/she is accepted into the biometric system, FER, FRR and FAR will be very low. The proposed biometric system, as compared with other biometric system (such as voice, fingerprint, eye recognition, etc.), although they are equally secured and require less effort in using them, keystroke dynamic approaches are very less costly biometric authentication system.

It was also therefore concluded that, in light of the fact that there is no requirement for any extra equipment other than an ordinary computer keyboard, and in light of the fact that the proposed system was viewed as easy to use and secured; the proposed system is comparatively secured and less costly. In general, it is concluded that utilizing keystroke analysis enhances the security features and efficiency of conventional username/password based verification methods, at a very low cost. It can also be generally concluded that, keystroke dynamics can be used as add-up security measure for securing web applications.

### 6.1 Comparative Analysis

A comparative analysis has been illustrated for the proposed and existing methods to compare the FRR, FER and FAR as well as irritability issues of the proposed Keystroke Dynamics Algorithm (KDA) method. This study achieved a minimum FER, FAR and FRR of 6%, 1% an 4% respectively. Also the user irritability issues is minimized by applying the techniques of Euclidean and Manhattan distance seamless together which provides flexibility for both slow and fast typists which intend limits false rejection of legitimate users.

Both dwell time (DT) and flight time (FT) are often extracted as feature vector for static authentication but one key feature which is very important to keystroke dynamics which is often not extracted is locate time (LT). The locate time is explained as, how long the user suspends his/her hand in the air, looking for the next key to press and allows the algorithm to record dynamic bound timings for different users and even for the same user at certain period. The inclusion of LT feature in the algorithm minimizes the issues of user irritability which bores a lot of keystroke dynamic driven systems users as detailed in section 5.1.1 of this paper.

Table 7 illustrates the comparative studies of the proposed system and the existing methods. The FAR of the proposed KDA is 1% percent, similar to the method presented in ([17], [26]) but better than methods used in ([4], [30] and [22] – using Mahalanobis). The proposed KDA method achieved a much similar FRR of 6% as presented in ([4], [26] and better than [22] – using Euclidean distance).

Since the intended system is able extract LT together with FT and DT and uses both Euclidean and Manhattan distance seamlessly to train user keystrokes samples by adapting Euclidean distance for fast typists and Manhattan distance for slow typist, the system leans and adapt to the user typing behavior to creates the thresholds that avoids user irritation during login sessions of most keystroke dynamic driven systems, therefore, the proposed KDA method ensured smooth enrollment process and allow legitimate users to access their accounts with minimized irritation than the other methods, thus FER is 4% which is not considered in other studies in literature.

## 7. CONCLUSION

It was therefore concluded that, the proposed system, with an FER of about 4%, FRR of around 6% and a minimized FAR of approximately 1%, is easy to use, accurate, and secured.

Literature proposes that, in order to attain a low FER, FRR and FAR, the three major keystroke features, which are the dwell time, flight time and the locate time should be captured individually from the user's typing pattern, of which this paper offers. This study adds to literature by introducing the bound timings concept whereby flight time is always added to the dwell time and the summed

value is multiplied by the locate time of the user's inputting pattern. In this case, a very low FER, FRR and FAR is far achieved.

## CONSENT

Informed consent was obtained from all individual participants included in the study and the confidentiality of the participants was assured.

## ACKNOWLEDGEMENT

We acknowledge the support of Nanjing Municipal Government-Nanjing University of Science and Technology (NMG-NJUST) Joint Scholarship for International Students.

## COMPETING INTERESTS

Authors have declared that no competing interests exist.

## REFERENCES

1. Monrose R, Rubin A. Keystroke dynamics as a biometric for authentication. Future Generation Computer Systems. 1999; 16(4):351-359.
2. Krause M, Tipton HF. Handbook of information security management. CRC Press LLC, Boca Raton. 2011;5.
3. Osei BM, Missah YM. Utilizing Keystroke Dynamics as an additional security measure to password security in computer web-based applications - A Case Study of UEW. International Journal of Computer Applications. 2016;149(5):35-44.
4. Chourasia N. Authentication of the user by keystroke dynamics for banking transaction systemǁ, Proceedings of International Conference on Advances in Engineering & Technology. 2014;41–45.
5. Nisha JR, Kumar Anto RP. User authentication based on Keystroke Dynamics. Journal of Engineering Research and Applications. 2014;4(3): (Version 1):345-349.
[ISSN: 2248-9622]
Available: http://www.ijera.com
6. Ahmed A, Traore I. Biometric recognition based on free-text Keystroke Dynamics, IEEE Transactions on Cybernetics. 2014; 44(4):458–472.
7. Senk C, Dotzler F. Biometric authentication as a servicefor enterprise identity management deployment: A data protection perspective, in Proceedings of the 6th International Conference on Availability, Reliability and Security (ARES'11). 2011;43–50.
8. Leedy PD, Ormrod JE. Practical research planning and design. 10th ed. Boston: Pearson; 2013.
9. Kaewwit C, Lursinsap C, Sophatsathit P. High accuracy EEG biometrics identification using ICA and AR model. Journal of Information and Communication Technology. 2017;16(2):354–373.
10. Bajaj S, Kaur S. Typing speed analysis of human for password protection (Based on Keystrokes Dynamics). International Journal of Innovative Technology and Exploring Engineering. 2013;3(2).
11. Rudrapal SD, Debbarma S. Improvisation of biometrics authentication, 10th International Conference, ICDCIT 2014, Bhubaneswar, India. 2014;287–292.
12. Wilson AAT, Nandita B, Bala S. Privacy-preserving biometrics authentication systems using fully homomorphic encryption. International Journal of Pervasive Computing and Communications. 2015;11(2):151- 168.
13. Krishnamoorthy S, Rueda L, Saad S, Elmiligi H. Identification of user behavioral biometrics for authentication using Keystroke Dynamics and machine learning. Proceedings of the 2018 2nd International Conference on Biometric Engineering and Applications - ICBEA 18; 2018.
DOI:10.1145/3230820.3230829
14. Odei-Lartey EO, Boateng D, Danso S, Kwarteng A, Abokyi L, Amenga-Etego S, Owusu-Agyei S. The application of a biometric identification technique for linking community and hospital data in rural Ghana. Global Health Action. 2016;9: 29854.
15. Boopathi M, Aramudhan M. Dual-stage biometrics-based password authentication scheme using smart cards. Cybernetics and Systems. 2017;48(5):415–435.
16. Attila M, Zoltán B, László C. Strengthening passwords by Keystroke Dynamics IEEE international workshop on intelligent data acquisition and advanced computing systems: Technology and Applications. Dortmund, Germany; 2007.
17. Venugopal PC, Viji KSA. Applying empirical thresholding algorithm for a keystroke dynamics based authentication system. Journal of Information and

Communication Technology. 2019;18(4): 383-413.
18. Mondal S, Bours P. Person identification by keystroke dynamics using pairwise user coupling. IEEE Transactions on Information Forensics and Security. 2017; 12(6):1319–1329.
19. Maheswari T, Anitha S. User authentication based on biometrics for convincing user in keystroke dynamics‖, ISR National Journal of Advanced Research in Computer Science Engineering and Information Technology. 2014;1(1): 63–70.
20. Kim J, Kim H, Kang P. Keystroke dynamics-based user authentication using freely typed text based on user-adaptive feature extraction and novelty detection. Applied Soft Computing. 2018;62:1077 1087.
21. Wangsuk K, Anusas-amornkul T. Trajectory mining for keystroke dynamics authentication. Procedia Computer Science. 2013;24:175. Available:http://dx.doi.org/10.1016/j.procs. 2013.10.041
22. Brochoux A, Clarke NL. Deployment of keystroke analysis on a smartphone. Australian Information Security Management Conference, Edith Cowan University: Perth, Western Australia; 2008. Available:http://ro.ecu.edu.au/ism/48
23. The PS, Teoh ABJ, Shigang Y. A survey of keystroke dynamics biometrics. The Scientific World Journal. Hindawi Publishing Corporation. 2013;24. [Article ID 408280] Available:http://dx.doi.org/10.1155/2013/40 8280
24. Krishna DR. Keystroke Dynamics - Dangling issues of providing authentication by Recognising User Input. Control Theory and Informatics. 2014;4(1). Available:http://www.iiste.org/Journals/inde x.php/CTI/article/view/9403 [ISSN 2224-5774 (Paper) ISSN 2225-0492 (Online)]
25. Ngugi B, Kahn BK, Tremaine M. Typing biometrics: Impact of human learning on performance quality. Journal of Data and Information Quality. 2011;2(2) article 11.
26. Vinayak R. User authentication using advanced keystroke analysis. International Journal of Advance Technology in Engineering and Science. 2015;3(1).
27. Seham B, Mohamed B, Osman H. Keystroke authentication on enhanced needleman alignment algorithm. Intelligent Information Management. 2014; 6:211-221.
28. Yamane T. Statistics, An Introductory Analysis, 2nd Ed., New York: Harper and Row; 1967.
29. Gagbla KG. Securing E-Business Applications. Using Keystroke Dynamics as a Biometric Authentication Technique; 2005. Available:http://citeseerx.ist.psu.edu/viewd oc/download?doi=10.1.1.101.9517
30. Gunetti D, Picardi C. Keystroke Analysis as a Tool for Intrusion Detection. Continuous authentication using biometrics: data, model, and metrics. Issa Traore: IGI Global; 2012.



*Peer-review history:*
*The peer review history for this paper can be accessed here:*
*http://www.sdiarticle4.com/review-history/53624*