Non-conventional keystroke dynamics for user authentication

Article

Accepted Version

Non-Conventional Keystroke Dynamics for User Authentication

Arwa Alsultan[a]*, Kevin Warwick[b] and Hong Wei[a]

[a]*School of Systems Engineering, University of Reading, Reading RG6 6AH, UK*
[b]*Vice Chancellors Office, Coventry University, Priory Street, Coventry CV1 5FB, UK*

Abstract

This paper introduces an approach for user authentication using free-text keystroke dynamics which incorporates the use of non-conventional keystroke features. Semi-timing features along with editing features are extracted from the users' typing stream. Decision trees were exploited to classify each of the users' data. In parallel for comparison, Support Vector Machines (SVMs) were also used for classification in association with an Ant Colony Optimization (ACO) feature selection technique. The results obtained from this study are encouraging as low False Accept Rates (FAR) and False Reject Rates (FRR) were achieved in the experimentation phase. This signifies that satisfactory overall system performance was achieved by using the typing attributes in the proposed approach. Thus, the use of non-conventional typing features improves the understanding of human typing behavior and therefore, provides significant contribution to the authentication system.

1. Introduction

The ongoing quest to find a technique to protect sensitive data and computer systems from harmful imposters, whilst also maintaining ease of use, is an important challenge in the field of computer and information security. Because the ID/password pair, the most common method for authentication, frequently fails to deliver an adequate balance between security and user-friendliness, more sophisticated methods have to be used. This is due to the ID/password pair being prone to social engineering, cracking and other forms of exploitation. Therefore, users are compelled to use extreme measures to safeguard their passwords, a procedure which includes remembering long and complex passwords in addition to the need for changing their passwords periodically [1] which causes them to endure great amounts of frustration and apprehension.

This research focuses on an alternative to the ID/password that verifies the identities of users based on their unique typing rhythms. This method provides a sufficient balance between practicality and safety, without requiring any additional hardware. Keystroke dynamics is considered to be an effortless behavior-based method for user authentication which employs the person's typing patterns for validating his/her identity. As was mentioned in [2], keystroke dynamics is "not what you type, but how you type." In this approach, the user types in text, as usual, without any extra work to be done for authentication. Moreover, it only involves the user's own keyboard and no other external hardware.

Keystroke dynamics is normally based on timing features that compute time lapses between two actions on the keyboard such as key press and key release. In this study, however, we investigate the use of non-conventional keystroke features in the authentication of users. Features such as typing speed, error rate, and shift key usage are utilized to find typing patterns that can be used to distinguish between individuals. Non-conventional features are considered during long free text input as they are extracted using calculations that spread along extended typing time.

These non-conventional features are important due to the lack of sufficient measurements that conventional keystroke dynamics present. Conventional keystroke data, in a very different way to other biometrics (e.g. image processing), captures very little information [3]. This information consists of the data that can be extracted from two consecutive keystrokes such as: the elapsed time between the release of the first key and the press of the second (digraph latency) and the amount of time each key is held down (keystroke duration) [2]. The majority of research, carried-out earlier in this area, focused only on these conventional features.

To enlarge the amount of information that can be extracted from a user input and therefore assemble better indications about his/her typing behavior, we focus our studies on non-conventional typing features that can be extracted collectively during long text input, in which more information is available. Long free text input is experienced daily in a manner that can be used to achieve continuous authentication [4].

Although there are many applications of keystroke biometrics used with fixed short text such as password hardening [5], there are scenarios where long free text input is more suited. For example: identification of one-of-many users who all have access to the resources in a work environment, the subject is identified when using any easily accessed desktop by his/her typing behavior of an e-mail or any other document. Another potential application for such long free text is verifying the identity of students taking online quizzes or tests.

Most of the work done in the field of keystroke dynamics authentication focuses primarily on timing features while ignoring other typing behavior such as editing patterns. Even previous studies that have included some non-timing features have not delivered the significance of these features in the way that they still focused on the importance of the conventional timing features, in the authentication process [6,7]. For that reason, we were motivated to explore the area of non-conventional typing features in order to concentrate on their distinctive ability to distinguish between individuals. A more in depth study on the effect of using various non-conventional feature subset sizes, which is to our knowledge not covered in the literature, has also been conducted.

In our work decision trees and Support Vector Machines (SVMs) are used to classify the typing samples collected from participants. Also Ant Colony Optimization (ACO) is utilized to select features that contribute more to the system in the case of SVMs, as decision trees are capable of performing feature selection in the tree building phase [8].

The rest of this paper proceeds as follows. Section 2 briefly introduces keystroke dynamics theory and discusses similar prior research in the area of keystroke dynamics user authentication. Section 3 describes the method developed in this study, in which we discuss the specific non-conventional features included in the study. In Section 4 we present our experimental results and consider the data space under investigation. Discussion about our results and some comparisons with previous studies are also included in this section. The final section concludes the topic and points out our research contributions and future work.

2. Keystroke Dynamics

Keystroke dynamics is categorized into two basic classes, namely: fixed-text and free-text [9]. The fixed-text keystroke dynamics method uses the typing pattern of the user when entering a predefined text. The same text has been previously used to train the system and is delivered by the user at log-in time. In contrast, the free-text keystroke method is considered easier for the user as it overcomes the problem of memorizing the text, something that the fixed-text method suffers from. As its name suggests, in free-text keystrokes, the text used for enrolment does not have to be the same as the text used for log-in. Moreover, free-text keystroke dynamics is used for enhancing security through continuous and nonintrusive authentication [10]. Thus, this research uses the typing behavior of free-text to resemble real-world situations, which allows users the freedom of not having to remember any text in order to go through the authentication process.

Keystroke dynamics is utilized in users' authentication by extracting typing features at the log-in session and comparing them with the typing features extracted at the enrolment session. These features include, among others: typing latency[11], keystroke duration [2], typing speed [11], shift key usage patterns [12] and typing pressure [13]. If the extracted features are adequately similar, the user is authenticated and if not the user is denied access.

Keystroke features extraction is usually performed after obtaining the users' raw data [14]. Among the data, timing features are popularly used and they are computed using two main values, specifically: the press time and the release time of each key, in milliseconds. These features are: Hold time, Down- Down, UP-UP and Up-Down time. Most previous studies have typically employed more than one of these features [15].

Other non-conventional features, which are mainly used in free-text keystroke dynamics, were also considered in few studies. These features make use of extra information that can be obtained collectively during the training process. Unique patterns were produced after observing users for a longer period of time. Attributes such as the error rate and editing patterns have been found to give a fair idea about a user's typing behavior [9].

A large amount of research has been carried-out for quite some time to investigate how keystroke dynamics can aid user authentication in general. Specifically, we look here at some of the research that focuses on the extraction of non-conventional keystroke features, utilizing them in different ways.

The research conducted by Hempstalk et al. [16] included, in total, eight features in the typist dataset. Most of these features were based around the typing speed or error rate. The typing speed features included: average words-per-minute (WPM) rate, peak WPM and trough WPM, whilst error rate features included: backspaces, paired backspaces and average backspace block length.

In the research conducted by Villani et al. [3] long-text-input features were extracted. The feature set mainly consisted of percentages of key presses of many of (what were referred to as) special keys. Some of these percentage features were intended to capture the users' preferences for using certain keys or key groups. For instance some users do not capitalize or use much punctuation, which is a distinctive trait of their typing behavior.

Other percentage features were planned to acquire the user's text editing patterns. As an example, there are many ways to locate a specific key, such as using other keys, i.e. Home, End and Arrow keys, or using mouse clicks. There is also a large number of ways to delete, such as Backspace, Delete keys and Edit-Delete. Inserting and moving of words and characters can be done in different ways too, such as: Insert, shortcut keys, or Edit-Paste.

Shift-key patterns were incorporated in Bartlow and Cukic's research [17]. A password designed to enforce shift-key behavior consisting of 12 randomly generated characters was employed. The feature vector collected for each input sequence included many shift-related features. Examples of such features are: the average, standard deviation, maximum, minimum and total of the hold time for right shifts and left shifts. It also included the average, standard deviation, maximum, minimum and total of the delay time for right shifts and left shifts.

Based on the literature, only a few studies have taken into consideration non-conventional typing features such as features associated with editing patterns. Therefore, we are focusing, in this study, on these features to try and find consistent typing patterns that can be utilized for recognizing the particular typist.

3. Methodology

3.1. Feature Definition

A great deal of the research done in the keystroke dynamics field has been focused mainly on the timing features extracted from the user's typing stream. These features compute the time lapses between performing two actions on the keyboard such as calculating the time it takes a person to press a certain key, i.e. the Hold time, which can be done by subtracting the release time from the press time of that key. Latency time is computed in a similar way but the two actions are performed on two different keys pressed successively rather than both actions being performed on one key in the case of the Hold time. It is calculated by finding the time difference between the press time of the first key and the press time of the second key, in the case of Down-Down time. The Up-Up time and Up-Down time are also computed in similar manner [9].

In this research, we are striving to explore new features. Non-conventional features step away from the conventional methods which rely on computing the time lapses between performing two actions on the keyboard. Instead, non-conventional features focus on the overall typing patterns that a user follows during input that extends over a relatively long period of time. It considers the percentage of performing certain actions (in relation to the total number of actions), i.e. general typing actions or editing actions, which leads to understanding the user's typing behavior. Better perception of human typing patterns is particularly easier to capture while typing long free text in which more information can be extracted. We consider two types of typing features, namely: semi-timing features and editing features. We will briefly describe each category in this section as follows:

3.1.1. Semi-Timing Features

Different from the standard timing features used in most of the literature, we incorporate features that have been extracted using some form of time calculation. The time calculation followed in this category however, is slightly different from that of the regular timing features. These features have a collective property to them, as most of them are calculated during longer periods of time.

The first feature is the Word-per-Minute (WPM) feature which, as the name suggests, measures the user's average typing speed [16]. The total typing time is calculated from the very first key press until the very last key release and this is used in the final calculation of the WPM. The number of words are totaled and then divided by the total typing time in minutes; this is shown in Equation (1). Of course, this feature will easily distinguish between slow and fast typists. Nonetheless, it is not enough to find the difference between individuals who are close in typing speed.

$$\text{WPM} = \frac{\text{Number of words}}{\text{Total typing time in minutes}} \qquad (1)$$

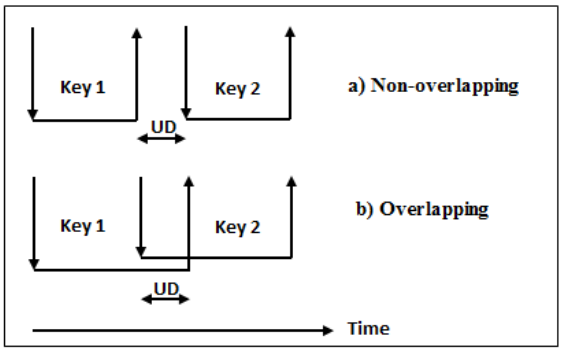

Fig. 1. Negative UD caused by overlapping keystroke events.

An interesting characteristic that can be found in some user's typing behavior is the number of negative Up-Down (negUD) actions detected in their typing stream. The negative Up-Down is due to an overlap happening between two successive keys being typed. This particular typing behavior is found in the typing stream of users who have the tendency to press the second key before releasing the first one. While most timing features are always positive because they represent the sequence determining the keyboard output, the Up-Down feature, can be negative in some cases that might involve fast typists [3].

Figure 1 illustrates two different two-key sequences showing the Up-Down time in a non-overlapping situation and in an overlapping one. A keystroke is represented as a horizontal line with the down arrow marking the press and the up arrow indicating the release time. In part (a), a positive Up-Down time was produced from non-overlapping keystroke events and in part (b), a negative Up-Down time was produced from overlapping keystroke events where the first key was released after the second was pressed.

Some studies found it challenging to deal with negative UD time [18]. Here we are using it to our advantage by finding the percentage of negative Up-Down instances for each user. As mentioned in [19], a negative value of UD implies time reduction or faster pressing while positive values imply time addition or slower pressing between two sequences of keystrokes. We found that some users have absolutely no negative UDs whilst others have a fair amount, which was consistent in all the typing tasks they produced. This gives a good indication that comparing the percentage of negative UDs can be a good method to assist in user recognition. NegUD is computed as the percentage of the number of negative UD appearances and the total number of key-pairs, i.e. two keys typed consecutively, this is shown in the following equation:

$$\text{negUD} = \frac{\text{Number of negative UDs}}{\text{Total number of keypairs}} \qquad (2)$$

A very similar typing behavior that has been, to our knowledge, hardly ever referred to in the literature is the negative Up-Up (negUU) time, which occurs when the typist tends to release the second key before releasing the first key. This characteristic happened with a few of our volunteers who participated in the data collection. Moreover, a negative UU only happens when there is a negative UD between the two successive keys. However if there happens to be a negative UD this does not mean that there is definitely a negative UU as shown in Figure 2.

Having said that, negative UU has the property of occurring less frequently, but if it does, there is a high possibility that it is a particular characteristic that an individual possesses intuitively. Thus there is a very good chance that it can be a good measure to employ in order to recognize that particular typist.

Similar to the previous feature, negUU is calculated as:

$$\text{negUU} = \frac{\text{Number of negative UUs}}{\text{Total number of keypairs}} \qquad (3)$$

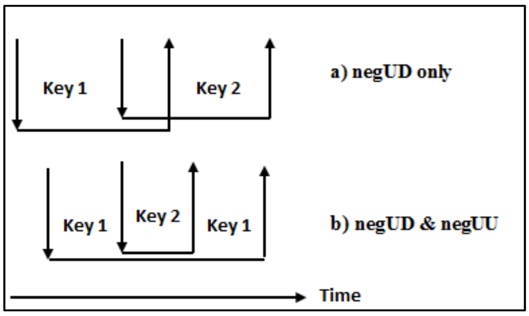

Fig. 2. Cases of negative UD only and negative UD/negative UU.

3.1.2. Editing Features

The second category of features does not give any attention to the time a user spends typing, rather it considers the way a user goes about the process of typing. Characteristics such as how frequently a user commits typing errors and how he/she edits text are studied here.

The error rate is the first feature in this category and it captures the percentage of times a user performs a typing error and corrects it [16]. This is simply calculated by dividing the number of times that a user commits an error, i.e. presses the backspace button, by the total number of letters typed, as follows:

$$\text{Error rate} = \frac{\text{Number of errors}}{\text{Total number of letters}} \qquad (4)$$

The next five features are closely related as they all associate with the way a user incorporates capital letters in typing. Including a capital letter is done either by using the CapsLock key on the keyboard or by using a shift key together with the letter intended to be capitalized. We noted that if a user has the habit of using the CapsLock key, then he will hardly ever use the shift key for capitalizing letters, and vice versa. Therefore, using these two attributes simultaneously might be a good clue to understand the user's editing habits.

The first measure is CapsLock key usage which calculates the percentage of the CapsLock keys being used to produce capital letters in a given typing task. This is simply computed using the following equation:

$$\text{CapsLock usage} = \frac{\text{Number of CapsLocks}}{\text{Total number of keys}} \qquad (5)$$

Shift key usage is a bit more complicated than it might appear to be as there are two different aspects in which users differ when it comes to shift key usage. The first shift key usage attribute is the right/left shift key choice. Some users use strictly the right shift or strictly the left shift whilst others alternate between the two [7]. The second attribute is the order of which the shift/letter keys are released. The shift key is always pressed before the letter key if the user is intending to produce a capital version of that letter. However, there are two orders that users go about when releasing those keys, they either release the letter key before releasing the shift key or they release the letter key after releasing the shift key. This behavior proved to be quite consistence throughout the different typing tasks for most users.

Based on the previous observations we suggest four different features that combine the two aspects of shift key usage. The percentage of each of the following was utilized; for the right shift key: Right Shift released After letter (RSA), Right Shift released Before letter (RSB); and for the left shift key: Left Shift released After letter (LSA), Left Shift released Before letter (LSB). They are calculated using Equation (6).

$$\text{S} = \frac{\text{Number of x}}{\text{Total number of shifts}} \qquad (6)$$

Where: x= right shifts released after letter, incase S=RSA;
x= right shifts released before letter, incase S= RSB;
x = left shifts released after letter, incase S= LSA;
x = left shifts released before letter, incase S= LSB.

4. Experiment, Results and Discussion

4.1. Data Space

A total of thirty users participated in this study for data collection. Participants had different levels of typing skills that varied between moderate and very good.

During data collection, the participants were asked to perform eight typing tasks. The tasks involved copying text that consisted of around 170 characters. The text included was an excerpt from the Guardian newspaper. The text included both upper and lower case letters in addition to numbers and punctuation marks. Although the tasks included text that was chosen for the users to type, it is still considered free-text as the text used for training is not related at all to that used for testing [20].

Users were directed to enter the samples in the most natural way possible, i.e. the same way they usually follow when typing. Users were allowed to enter carriage returns and backspaces if needed. The data collection was performed by a GUI program implemented using the C++ language. The application was downloaded on the users' personal machines to maximize their comfort as they are more familiar with their own machine and its surroundings. Therefore, they were able to feel more at ease, and thus, to perform the typing tasks in a manner closer to that of their real typing behavior.

A feature vector, containing the nine features used in this study, was created and was stored in the database as the user's profile. This process was carried out by considering each one of the eight typing tasks as a single typing sample, the features from which were extracted separately. Therefore, eight samples per subject were included in the analysis phase for classifier training and testing.

4.2. Experiment and Results

Decision trees have been chosen as a classifier in this research as they are strictly nonparametric and do not require assumptions regarding the distributions of the input data [21]. Furthermore, decision trees handle nonlinear relations between features and classes [22].

Classification was carried-out through cross-validation as the number of samples was not sufficient enough to perform a regular training/testing process. Cross-validation is a statistical sampling technique that aims to ensure that every example from the original dataset has the same chance of appearing in the training and testing set. We followed the leave-one-out cross-validation protocol which is a special case of the well-known n-fold cross-validation [23].

N-fold cross-validation divides the data up into n chunks and trains n times, treating a different chunk as the test sample each time; such that for each of n experiments, it uses n-1 folds for training and the remaining one for testing. Leave-one-out cross-validation is exactly the same except that all chunks contain only a single sample.

In our experiment, eight samples were used to perform eight cross-validation experiments. Seven of the samples were treated as the training sample set and the remaining sample was regarded as the testing sample. In each experiment, a different sample was selected to act as the test data.

The Statistics toolbox in Matlab was used to fit the tree and predict the class of each of the test data. Moreover, the tree structure, i.e. the order in which attributes were chosen to be tested at each node, differs each time when a different training set was selected.

Furthermore, two error rates were used to infer the performance, namely: False Accept Rate (FAR) and False Reject Rate (FRR). FAR is the percentage of impostors who have successfully gained access to the system whereas FRR indicates the percentage of legitimate users who were denied access to the system [24]. Low error rates were produced by this study. The FAR and FRR derived from the decision tree classification process are listed in Table 1. Both error rates are presented utilizing datasets created by different number of participants. Results produced by 15, 25 and 30 users showed an increase in the error rates between 15 and 25 users. Yet, when increasing the number of users from 25 to 30 the error rates were very similar which proves the scalability of this method and its ability to cope with larger sample size.

Using the nine features simultaneously had a good impact on the overall classification performance as the decision tree performs a form of feature selection in which only features that contribute to the overall-system decision are used in building the tree [8]. This is not the case when using only one or two features separately. This is due to the individual characteristics that each feature holds and that contribute collectively to the system's performance.

Table 1: System performance using multiclass classification.

	FAR			*FRR*		
Participants no.	15	25	30	15	25	30
Decision Tree	0.007	0.0104	0.0109	0.1	0.25	0.28
SVMs	0.0125	0.0181	0.0183	0.175	0.435	0.444

For comparison purposes, Support Vector Machines (SVMs) were also used in this experiment as it is one of the most successful classification techniques [25]. SVMs were chosen as a rival classifier because it follows a completely different mechanism to that of decision trees [26].

When using SVMs in classification, feature subset selection is in place. This is because a number of the non-conventional features are correlated with each other. Therefore, it is necessary to incorporate a feature subset selection mechanism when utilizing these features in order to reduce the dependency levels between the features [27]. Feature subset selection is also included in the building process of the decision tree where all redundant features are removed [8].

Feature subset selection is considered as an optimization problem, in which the space of all possible features is scrutinized to recognize the feature or set of features that produce optimal or near-optimal performance, i.e. those that minimize classification error [28]. Ant Colony Optimization (ACO) proved to be a good candidate for achieving that goal [29].

The selected features were passed to the multiclass SVMs machine learning mechanism in order to be used as the basic data for differentiating between classes. Leave-one-out cross-validation was also used to treat seven of the samples as the training sample set and the remaining one as the testing sample, in each cross-validation experiment. The classification process was implemented on MATLAB with the aid of the LIBSVM library [30].

This was done gradually by selecting one feature, using ACO, and then increasing the size of the feature set. Using only one or a small number of features yielded higher error rates. Similarly, using all or most of the nine features caused performance deterioration. The ideal size of feature set was 5 features which produced good FAR and FRR rates. A 0.0183 FAR and a 0.444 FRR were delivered using 5 features. Table 2 illustrates the influence of increasing the feature set size on the overall system's performance in a database containing 30 users.

Having the best features subset size to be only 5 features refers directly to the Curse of Dimensionality which corresponds to the problem that the amount of training needed grows exponentially with the number of features [31]. Since there were only 8 samples per person in this experiment, there has to be a reduction in the number of features used for classification to the least amount possible while conserving the maximum benefit provided to the classification process.

Using ACO, the features that contribute the most to the system performance in our experimentation were: negUD, Error Rate, RSB, LSA and LSB. Using these features solely in the classification process eliminated the redundancy caused by using all 9 features. That clearly contributes to improving the overall system performance. Furthermore, using only one or two of these features is not enough to find the fine differences between the typing behaviour of individuals in free-text keystroke dynamics.

Table 2: Error rates using different feature subset sizes.

No.	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
FAR	0.0315	0.0251	0.0248	0.0226	0.0183	0.0187	0.0191	0.0194	0.0203_
FRR	0.8194	0.6528	0.6435	0.5879	0.444	0.4861	0.4954	0.5046	0.52788

We understand that using a larger dataset and incorporating data from a greater number of participants will likely produce more reliable results. Therefore, similar to DTs, we incorporated data from datasets with different numbers of participants in the SVMs tests to understand how increasing the sample size will affect the system performance. In all these tests we decided to perform a subset selection of 5 features which proved to yield the best performance (as shown in Table 2).

Using datasets of samples size varying between 15, 25 and 30 users delivered a noticeable reduction in the system performance when increasing the number of participants from 15 to 25 (as shown in Table 1). Nonetheless, the increase from 25 users to 30 have produced very similar FAR and FRR. This illustrates the system reaching a stable level when enlarging the number of participants which proves its ability to work with datasets with large number of participants.

Moreover, decision trees operate by automatically performing feature subset selection in which the non-important or redundant features are not involved in the tree building process [8]. Features: LSB, negUD, negUU and CapsLock usage contributed most in building the decision tree as they formed the first levels of the tree structure. Thus, they collectively have a high ability to split the targets [32], which allows for better differentiation between individuals. Therefore, these features correspond to the features with higher impact on the performance of the recognition system. This partly matches the features extracted using ACO; as both LSB and negUD were found to have a considerable effect on system performance in both decision tree and SVMs/ACO classification cases.

Conclusively, Decision trees have a slight performance advantage over SVMs. They produced a higher accuracy system as the ROC is plotted closer to the upper left corner of the diagram in Fig.3.

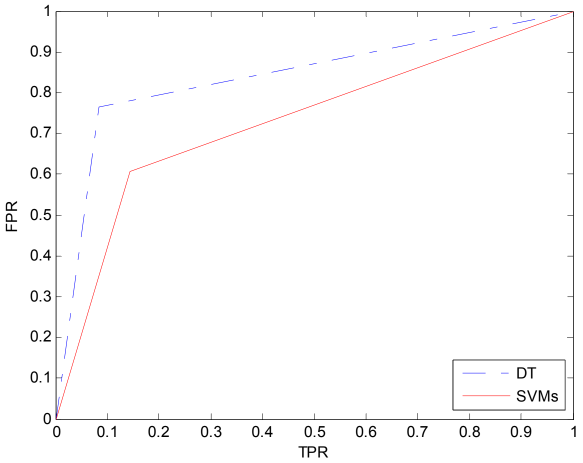

Fig. 3. Comparison between DT and SVMs by ROC curves.

The authentication process used until this point was done by training the system using data produced by the system's users to test if the system is able to recognize which of the system's users produced the test samples. Multiclass classification was utilized to achieve this aim. Multiclass classification works by deciding the test sample belongs to which of the available classes using the training data produced by all of the available classes [33].

In the second part of this study, we will focus on true intruder's recognition. In this section, typing samples from users who are completely un-known to the system are used to test the system's ability to recognize them as intruders and reject them. To achieve true intruder recognition, binary classification is used. For every test sample, binary classification, i.e. one-to-one classification, is performed against all available class to check if the system recognizes the intruder as any of the legitimate users.

Binary classification was performed using the training data of 25 genuine users and testing data from five intruders producing three typing samples each for testing the system. Table 3 shows the error rates produced by the 25 legitimate users without any intruders. The binary classification was performed for each user by representing the sample produced by that user as the positive class and all other samples are represented as the negative sample [34]. This was carried-out using cross-validation similar to the multiclass classification experiment.

Lastly, the data from the five intruders was tested against each of the legitimate users' training data using binary classification. This produced similar FAR to that produced by the 25 legitimate users especially in case of SVMs which indicates the power the system holds against un-known intruders. Moreover, in true intruder recognition SVMs performed better than DTs. This is due to the nature of SVMs which leans towards the class with heavy samples [35] i.e. the class with negative samples in this study. The FRR in the intruders test was not computed due to not testing any legitimate users in this experiment.

Table 3: System performance using binary classification.

	Legitimate users		***Intruders***	
	FAR	**FRR**	**FAR**	**FRR**
Decision Tree	0.011	0.375	0.051	n/a
SVMs	0.0112	0.49	0.014	n/a

4.3. Discussion

This study was performed using the data collected in the research conducted by Alsultan et al. [29] in which the researchers considered user classification based on timing features only. These features included the hold time, Up-Up, Down-Down and Up-down of specific key-pairs. Although the performance of the system described in [29] was acceptable, there was a larger than desired FRR.

By using non-conventional features the FRR has been dramatically improved with a value of 0.28 in this study. While this figure is still not ultimate, it is quite good when considering the small amount of text used to recognize individuals. Nonetheless, a satisfactory FAR was also produced. The FAR, being as small as 0.011, is very comparable that produced by conventional features which leads to high expectations of further research in this area. This proves the superiority of such non-conventional typing features over conventional timing ones, in user authentication.

The use of non-conventional features proposed in this paper have succeeded in providing a reliable medium for user authentication because employing these features enlarges the amount of information that can be extracted from a user's input. This is due to the fact that non-conventional typing features are extracted collectively during the whole time a text is being input by the user, in which more information is available, such as: words-per-minute, error rate, percentage of negative UDs … etc. Therefore, using this wide range of information available about the user's typing patterns, the system is able to assemble better indications about the user's typing behaviour, thus distinctively distinguish between individuals. Moreover, as the none-conventional features are collected during the whole time of text typing i.e. relatively long period, any random incidence that might occur will be averaged. As appose to the conventional timing features where few noisy appearances can affect the overall understanding of the use's typing pattern significantly.

Moreover, non-conventional features were utilized in the research conducted by Hempstalk et al. [16]. In their experiment, 8 features were extracted, some of which were based around the typist's speed: average words-per-minute (WPM) rate, peak WPM, trough WPM, error rate: backspaces, paired backspaces, average backspace block length or slurring of key press and release events: press/release ordering, press/release rate. A dataset consisting of 15 emails for each of 10 participants was created. Using one-class SVMs an FAR of 0.113 and an FRR of 0.331 were achieved. These results show that our research proved to realize a better FAR/FRR.

Similar research was conducted by Curtin et al. [36] in which 58 features were extracted. The features varied between conventional timing ones and non-conventional ones such as total time to enter the text, total number of key presses for Space, Backspace, Delete, Insert, Home, End, Enter, Ctrl, all four arrow keys, left and right shift keys and the number of left, right and double mouse clicks. Recognition accuracy of 98.5% resulted from data collected from 8 subjects typing ten 600-characters long training samples and ten 300-characters long testing samples. This would have been a very encouraging result if the number of subjects was larger and/or the length of text was much shorter. A comparison between the method proposed here and some of the state of the art studies is presented in Table 3.

Table 3: Comparison with state of the art studies.

Study	Participant no.	*Features*		*System performance*		
		Convent.	*Non-convent.*	*Accuracy*	*FAR*	*FRR*
Alsultan et al. [29]	25	√			0.001	0.504
Hempstalk et al. [16]	10		√		0.113	0.331
Curtin et al. [36]	8	√	√	0.985		
Proposed method	30		√	0.76	0.011	0.28

5. Conclusion

In this paper we examined the usefulness of incorporating non-conventional keystroke features in the user authentication process. Unlike conventional timing features, non-conventional features benefit from the extra information that can be extracted from long free-text input. Features that have semi-timing properties such as words-per-minute, percentage of negative Up-Down time and percentage of negative Up-Up time were used. Moreover, features that explain the user's editing behavior were also used. These included the error rate, percentage of CapsLock usage, and percentage of both right and left shift keys usage.

The experiment produced good results considering the fact that it used free-text for user authentication which gave a good balance between the system's security and the user's comfort. The FAR and FRR rates were both satisfactory with the FAR being the slightly better of the two.

Therefore, non-conventional features such as those used in this study appear to be highly significant in keystroke dynamics applications such as user authentication. Moreover, decision tree classifiers also demonstrated a high level of success in such cases.

There is much more that can be done to improve this approach. One example of which is to expand on the typing features to include other non-conventional features such as the users' inserting and moving habits. Experimenting with different classification methods might also contribute positively to the overall system performance.

The fusion of conventional timing features and the non-conventional features presented here might work in favor of a better understanding the user's typing patterns which can be utilized to improve the error rates produced by merely non-conventional features. This is clearly ongoing research in which results thus far are extremely encouraging.

Acknowledgements

The authors wish to extend their gratitude to the participants who were involved in this experiment for the time they took out of their busy schedules to contribute in this study.

References

[1] R. Biddle, M. Mannan, P.C.V. Oorschot, T. Whalen, User study, analysis, and usable security of passwords based on digital objects, IEEE Transactions on Information Forensics and Security. 6 (2011) 970–979.

[2] F. Monrose, A. Rubin, Authentication via keystroke dynamics, in: 4th ACM Conference on Computer and Communications Security, New York, 1997: pp. 48–56.

[3] M. Villani, C. Tappert, G. Ngo, J. Simone, H. St. Fort, S. Cha, Keystroke Biometric Recognition Studies on Long-Text Input under Ideal and Application-Oriented Conditions, in: The IEEE Computer Society Workshop on Biometrics, 2006.

[4] P. Bours, H. Barghouthi, Continuous authentication using biometric keystroke dynamics, in: The Norwegian Information Security Conference, 2009.

[5] F. Monrose, M.K. Reiter, S. Wetzel, Password Hardening Based on Keystroke Dynamics, in: The 6th ACM Conference on Computer and Communications Security, New York, 1999: pp. 73 – 82.

[6] D. Stefan, D. Yao, Keystroke-dynamics authentication against synthetic forgeries, in: The 6th International Conference on Collaborative Computing (CollaborateCom), Chicago, 2010: pp. 1–8.

[7] E. Lau, X. Liu, C. Xiao, X. Yu, Enhanced User Authentication Through Keystroke Biometrics, in: Computer and Network Security, Massachusetts Institute of Technology, 2004.

[8] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed., Prentice Hall, Englewood Cliffs, 2010.

[9] A. Alsultan, K. Warwick, Keystroke dynamics authentication: a survey of free-text methods, International Journal of Computer Science Issues. 10 (2013) 1–10.

[10] S.P. Banerjee, D.L. Woodard, Biometric Authentication and Identification using Keystroke Dynamics : A Survey, Journal of Pattern Recognition Research. 7 (2012) 116–139.

[11] J. V. Monaco, J.C. Stewart, S. Cha, C.C. Tappert, Behavioral biometric verification of student identity in online course assessment and authentication of authors in literary works, in: IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS), Arlington, 2013: pp. 1–8.

[12] E. Lau, X. Liu, C. Xiao, X. Yu, Enhanced user authentication through keystroke biometrics, in: Computer and Network Security, 2004.

[13] H.R. Lv, Z.L. Lin, W.J. Yin, J. Dong, Emotion recognition based on pressure sensor keyboards, in: IEEE International Conference on Multimedia and Expo, Hannover, 2008: pp. 1089 –1092.

[14] H. Saevanee, P. Bhattarakosol, Authenticating user using keystroke dynamics and finger pressure, in: The 6th IEEE Consumer Communications and Networking Conference, 2009: pp. 1–2.

[15] L.C.F. Araujo, L.H.R. Sucupira, M.G. Lizarraga, User authentication through typing biometrics features, IEEE Transactions on Signal Processing. 53 (2005) 851 – 855.

[16] K. Hempstalk, E. Frank, I.H. Witten, One-class classification by combining density and class probability estimation, in: The European Conference on Machine and Learning and Principles and Practice of Knowledge Discovery in Database, 2005: pp. 505–519.

[17] N. Bartlow, B. Cukic, Evaluating the reliability of credential hardening through keystroke dynamics, in: The 17th International Symposium on Software Reliability Engineering (ISSRE'06), 2006.

[18] J. Ilonen, Keystroke dynamics, Lappeenranta University of Technology. (2006).

[19] D.Y. Liliana, D. Satrinia, Adaptive behaviometrics using dynamic keystroke for authentication system, in: International Conference on Future Information Technology, Singapore, 2011.

[20] D. Gunetti, C. Picardi, Keystroke analysis of free text, ACM Transactions on Information and System Security. 8 (2005) 312–347.

[21] M. Friedl, C. Brodley, Decision tree classification of land cover from remotely sensed data, Remote Sens. Environ. 61 (1997) 399–409.

[22] B. Deshpande, Decision tree digest - understand, build and use decision trees for common business problems with RapidMiner, SimaFore. (2014).

[23] P. Refaeilzadeh, L. Tang, H. Liu, Cross-Validation, Encyclopedia of Database Systems. (2009) 532–538.

[24] M. Karnan, M. Akila, N. Krishnaraj, Biometric personal authentication using keystroke dynamics: a review, Applied Soft Computing. 11 (2011) 1565–1573.

[25] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery. 2 (1998) 121–167.

[26] K. Warwick, R. Craddock, An introduction to radial basis functions for system identification. A comparison with other neural network methods, in: 35th IEEE International Conference on Decision and Control, Kobe, 1996: pp. 464–469.

[27] D. Shanmugapriya, G. Padmavathi, An Efficient Feature Selection Technique for User Authentication using Keystroke Dynamics, International Journal of Computer Science and Network Security (IJCSNS). 11 (2011) 191–195.

[28] M. Dorigo, A. Colorni, V. Maniezzo, The ant system: an autocatalytic optimizing process, Tech. Rep. 91-016, Université Libre de Bruxelles, Milano. (1991).

[29] A. Alsultan, K. Warwick, H. Wei, Feature Subset Selection for Free-text Keystroke Dynamics, 2016.

[30] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, (2001). http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[31] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Transactions on Pattern Analysis And Machine Intelligence. 22 (2000) 4 – 37.

[32] L. Rokach, O. Maimon, Data mining and knowledge discovery handbook, 2010.

[33] G. Saggio, G. Costantini, M. Todisco, Cumulative and ratio time evaluations in keystroke dynamics to improve the password security mechanism, Journal of Computer and Information Technology. 1 (2011) 2–11.

[34] R. Giot, M. El-Abed, B. Hemery, C. Rosenberger, Unconstrained keystroke dynamics authentication with shared secret, Computers & Security. 30 (2011) 427–445. doi:10.1016/j.cose.2011.03.004.

[35] H. He, A. Ghodsi, Rare class classification by support vector machine, in: Proceedings - International Conference on Pattern Recognition, 2010: pp. 548–551. doi:10.1109/ICPR.2010.139.

[36] M. Curtin, C. Tappert, M. Villani, G. Ngo, J. Simone, H.S. Fort, et al., Keystroke biometric recognition on long-text input: a feasibility study, in: IMECS, 2006.