Towards Keystroke Continuous Authentication Using Time Series Analytics

Abdullah Alshehri, Frans Coenen and Danushka Bollegala

Abstract An approach to Keystroke Continuous Authentication (KCA) is described founded on a time series analysis based approach that, unlike previous work on KCA (using feature vector representations), takes the sequencing of keystrokes into consideration. The significance of KCA is in the context of online assessments and examinations used in eLearning environments and MOOCs, which are becoming increasingly popular. The process is fully described and analysed, including comparison with established feature vector approaches. Our proposed method outperforms these other approaches to KCA (with a detection accuracy of 94 %, compared to 79.53 %), a clear indicator that the proposed time series analysis based KCA has significant potential.

Keywords Keystroke Pattern Recognition · Keystroke Time Series · Behavioural Biometrics · Continuous Authentication

1 Introduction

Keystroke patterns (typing patterns) are a recognised behavioural biometric for establishing the security credentials of users in the context of static user authentication. The fundamental idea is that the rhythm of typing a predefined text by a legitimate user can be learned, and consequently used for authentication purposes [1]. Keystroke Static Authentication (KSA) has been applied with respect to applications such as password, username and pin number authentication [2–5]. However, KSA is unsuited to applications that require continuous authentication such as in the context of the online assessments and examinations frequently used in eLearning environments

A. Alshehri (✉) · F. Coenen · D. Bollegala
Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK
e-mail: a.a.alshehri@liverpool.ac.uk

F. Coenen
e-mail: coenen@liverpool.ac.uk

D. Bollegala
e-mail: danushka.bollegala@liverpool.ac.uk

M. Bramer and M. Petridis (eds.), *Research and Development in Intelligent Systems XXXIII*, DOI 10.1007/978-3-319-47175-4_24

and MOOCs, where Keystroke Continuous Authentication (KCA) is required. KCA is significantly more challenging than KSA because the process relies on detecting patterns from free text (unlike in the case of KSA where we are looking for a single fixed pattern).

Work on KCA to date has been predominantly focussed on feature vector based binary classification where the features are statistics, such as the average hold time (duration of a key press) and digraph latency (duration between the start or end of pairs of common consecutive key presses) [6–11]. These systems operate by continuously measuring the similarity between a learnt user statistical profile and previously unseen profiles presented in the form of a data stream. However, there is a great deal of variability in the statistical features used to make up the feature vectors and consequently the reported results to date have tended to not be as good as anticipated. The overriding disadvantage of the feature vector based approaches is that the sequencing of key presses is largely lost. In addition, classifiers (predictors) need to be built for each user and this in turn adversely affects the efficiency of the application of KCA in real environments. The idea presented in this paper is to conceptualise the keystroke process in terms of a ongoing time series from which KCA can be realised through a time series analysis process rather than using a feature vector based classification approach. More specifically the idea is to view keystrokes in terms of press-and-release temporal events such that a series of successive events can be recorded. Each keystroke is defined in terms of a pair $P = (t, k)$, where t is a time stamp or temporal identifier of some form; and k is some keystroke attribute (such as flight time between keys or key-hold length). The intuition is that the time series paradigm can be more readily used to dynamically identify "suspect behaviour", in real time, because it serves to capture keystroke sequences (unlike in the case of statistical techniques).

The rest of this paper is organised as follows. In Sect. 2 a brief state-of-the-art review of KCA is presented. This is followed in Sect. 3 with a description of the proposed keystroke time series representation, while in Sect. 4 we introduce the proposed KCA approach. Keystroke time series similarity is then discussed in Sect. 5. The evaluation and comparison of the proposed approach is reported on in Sect. 6. Finally, the paper is concluded with a summary and some recommendations for future work in Sect. 7.

2 Previous Work

Dealing with keystroke patterns, especially KCA, in terms of time series has received little attention in the literature. Reference is made in [12] where a sequence pattern mining algorithm is presented for which a potential suggested application is KCA; an idea that has some similarity with the time series approach proposed in this paper. The majority of work on keyboard usage authentication has been directed at the idea of using statistical feature vectors to recognise keystroke patterns. As mentioned in the introduction to this paper, keystrokes have a range of timing features associated with them including key-down, key-up and hold time. Also, given sequences of

pairs of keystrokes, the flight time between n successive keystrokes can also be considered. These features have been represented in terms of feature vectors by computing statistical quantitative equivalents. Such feature vectors have then been used to recognise typing patterns with a view to keyboard usage authentication. The similarity, between (say) two typing profiles may be measured, for example, in terms of the extent of the distance between two vectors.

One of the earliest studies that have considered the idea of keyboard authentication is [13] where the authors utilized the concept of digraph latency for the feature vector representation from which a binary classifier was generated. The classifier operated using the mean and standard deviation of digraph occurrences in a training profile. Only digraphs that satisfied a predefined threshold of occurrences were used; not all digraphs were included. The principal disadvantages of the approach were that: (i) to achieve a reasonable classification performance a substantial amount of data was required with which to train the classifier, the classifier requires an average of 6,390 digraphs to recognise a useful pattern, and (ii) a dedicated classifier was required for each individual. The approach would thus be difficult to apply in the context of eLearning platforms and MOOCs. An alternative approach was presented in Gunetti and Picardi [9] where the average time for pressing frequent key sequences (n-graphs) was recorded and stored in arrays, one per n-graph. Common n-graphs were extracted for corresponding samples (reference and test). The elements of the arrays were then ordered and the distance between sample pairs computed by comparing the ordering in the reference array with the ordering in the test array. This measure was referred to as "the degree of disorder". However, learning a reference sample depends on all other samples in the reference profile. This can cause an efficiency issue when dealing with large numbers of samples. The training of legitimate users profiles is thus as time consuming as in [13]. Ahmed et al. [6] have used key-down time information and the average of digraph flight time and monograph to represent features. An Artificial Neural Network classifier was used. This mechanism worked reasonably well in a controlled experimental setting, typing of the same text using the same keyboard layout in an allocated environment. This is not the situation we would find in uncontrolled environments, such as those used for conducting eLearning assessments and examinations.

From the above, most KCA studies have been directed at the usage of quantitative statistical measures to represent keystroke features founded on a feature vector representation. However, it is argued here that the feature vector representation may not necessarily be representative of useful typing patterns. Therefore, it is conjectured that representing keystroke features as a time series can lead to a better interpretation of typing patterns with respect to KCA.

3 Keystroke Time Series Representation

Keyboard usage is typically undertaken in a sequential manner key-press by key-press, on occasion two keys may be pressed together (for example using the shift or control keys). Thus typing action is well suited to representation in terms of

time series where each key press describes a point (event) within the time series. More formally a Keystroke time series K_{ts} is a sequential ordering of a set of data points that occur within a specified interval of time, $K_{ts} = \{p_1, p_2, \ldots, p_n\}$ where each point p_i corresponds to a tuple of the form $\langle t_i, k_i \rangle$ where t_i is a temporal identifier of some description and k_i is some associated attribute value. Thus $K_{ts} = \{\langle t_1, k_1 \rangle, \langle t_2, k_2 \rangle, \ldots, \langle t_n, k_n \rangle\}$. As such, a time series can be viewed as a 2D plot with time along the x-axis and attribute value along the y-axis. The value for t_i can be either: (i) key-down time KD^t, (ii) key-up time KU^t or (iii) a sequential ID number KN per keystroke. However, when using KD^t or KU^t the "ticks" along the time series x-axis are typically irregularly spaced which in turn tends to hinder the time series analysis. Therefore, in the context of the work presented in this paper, KN was used for t. For k_i we have used flight time F^t. Flight time serves to capture the duration between keystrokes which is lost if we use KD^t or KU^t for t. Thus, F^t was adopted, $K_{ts} = \{\langle KN_1, F_1^t \rangle, \langle KN_2, F_2^t \rangle, \ldots\}$. With respect to the forgoing the following definitions should be noted:

Definition 1 A Keystroke Time Series K_{ts} is an ordered discrete sequence of points; $K_{ts} = [p_1, p_2, \ldots, p_i, \ldots, p_n]$ where $n \in \mathbb{N}$ is the length of the series, and p_i is a tuple corresponding to a feature pairing.

Definition 2 A keystroke time series sub-sequence s_k, of length l, is generated from K_{ts} and starts at the position i, $s_k = [p_{ki}, p_{k(i+1)}, \ldots, p_{k(i+j)}]$, where $k \in \mathbb{N}$ is the identifier of current sub-sequence, $j = l - 1$ and $1 < l < n - l$. Thus an ordered subset of K_{ts} can be indicated using the notation $s_k \preceq K_{ts}$ $(\forall p_{ki} \in s_k, \exists p_i \in K_{ts})$.

Definition 3 A profile P is a set of k keystroke time series sub-sequences $P = \{s_1, s_2, \ldots, s_k\}$.

In practice each K_{ts} describes a task dependent keyboard session. For example, in the context of an *online* assessment where a student is answering an assessment question, the generated K_{ts} should represent that keyboard session associated with the student conducting this task. For KCA to operate each time series K_{ts} needs to be evaluated at the start (by comparison of an initial sub-sequence s_1 of the time series with P, a previously stored "bank" of sub-sequences for the subject). As the session proceeds continuous comparison needs to be undertaken by comparing the most recent sub-sequence s_i with earlier collected sub-sequences $S = \{s_1, s_2, \ldots, s_{i-1}\}$ (of the time series K_{ts}).

The fundamental idea proposed in this paper is illustrated in Fig. 1 generated using several randomly selected time series from the sample data used for evaluation purposes as presented later in this paper (see Sect. 6). The figure shows four keystroke time series sub-sequences representing two subjects, two sub-sequences from each subject (Subjects 2 and 9). From the figure it can be seen that there are clear similarities in the keystroke sub-sequences associated with the same subjects (despite the sub-sequences being related to different texts), and clear dissimilarities in the keystroke sub-sequences associated with different subjects. It is thus argued here that such time series can be fruitfully used for KCA.

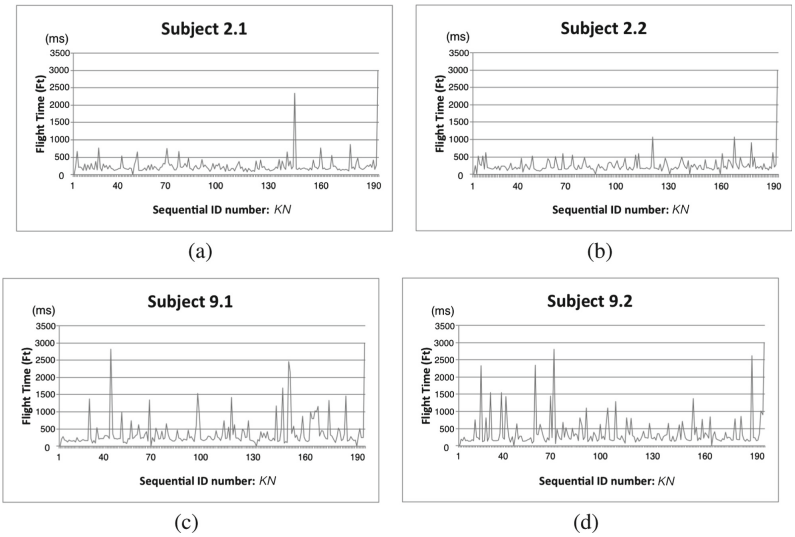

Fig. 1 Examples of keystroke time series: **a** and **b**, time series for Subject 2 writing two different texts; **c** and **d**, time series for Subject 9 writing two different texts

4 Proposed KCA Approach

The proposed KCA process is presented in Algorithm 1. The inputs are: (i) a desired sampling frequency f, (ii) a desired keystroke time series sub-sequence length m and (iii) a minimum size l for S (a set in which to hold collected sub-sequences) before similarity measurements can be made and (iv) a similarity threshold σ value. The process operates on a continuous loop; after every f ticks (line 4) a keyboard time series sub-sequence s of length m is constructed (line 5). Not every time series is usable, for example, there may be sizeable durations between key presses indicating "away from keyboard" events, thus the generated subsequence s needs to be verified for its usability (line 6). Recall that for the time stamps keypress indexes, not the actual time of the key press (for reasons presented earlier) were used. The usability of a time series can be simply identified from the presence of an excessive flight time value. If s is usable and we have collected a sufficient number of sub-sequences ($|S| > l$) authentication can be undertaken (line 7). This is done by calculating a similarity index (simIndex); the simplest way of doing this is to obtain an average of the similarity values between s and the sub-sequences in S. If $simIndex \geq \sigma$, then an authentication error has occurred (lines 8 to 10). Note that in a similar manner to plagiarism checkers (such as Turnitin[1]) the proposed KCA is essentially a similarity checker, thus when dissimilarity is found this is an indicator of further investigation being required. The most recent time series s is then added to S (line 11) and the process continuous until a data stream end signal is received (line 19).

[1] http://www.turnitinuk.com/.

Note also that Algorithm 1 does not include any "start up comparison" as described in the foregoing section. However, the similarity checking process is more-or-less the same; a similarity index can be generated and compared to a value σ. The most important part of Algorithm 1 is the similarity checking process, the mechanism for comparing two time series. This mechanism is the central focus, and contribution, of this paper; and is therefore considered in further detail in Sect. 5.

Algorithm 1 Dynamic KCA process

Input: f = sampling freq., m = time series subsequence length, l = min. size of S for sim. calc., σ = similarity threshold.
Output: Similarity "highlights".

```
1: S = Set of time series sub-sequences sofar (empty at start)
2: t_i = 1
3: loop
4:     if remainder t_i/f ≡ 0 then
5:         s = time series {p_i, ..., p_m}
6:         if usable(s) and |S| > l then
7:             simIndex = similarityIndex(s, S)
8:             if simIndex >= σ then
9:                 highlight
10:            end if
11:            S = S ∪ s
12:        end if
13:    end if
14:    if end data stream then
15:        Exit loop
16:    else
17:        t_i = t_i + 1
18:    end if
19: end loop
```

5 Measuring the Similarity of Keystroke Time Series

The most significant part of the KCA process described above is the time series analysis element where a current keystroke time series sub-sequences s is compared with one or more previous series. Given two keystrokes time series sub-sequence s_1 and s_2, the simplest way to define their similarity *sim* is by measuring the corresponding distances between each point in s_1 and each point in s_2. In other words the Euclidean distance is measured between the points in the two series, summed and divided by the sub-sequence length to give an average distance. However, this approach requires both sub-sequences to be of the same length and, more significantly, tends to be over simplistic as it assumes a one-to-one correspondence. By returning to the time series sub-sequences given in Fig. 1a, b we can notice that the shape of the two series is similar but the "peaks" and "troughs" are offset to one another. Euclidean distance measurement will not capture this noticeable similarity. To overcome this limitation, Dynamic Time Warping (DTW) mechanism was adopted. The idea here being to

measure the distance between every point in s_1 with every point in s_2, and recording these distances in a $|s_1| \times |s_2|$ matrix. This matrix can then be used to find the "best" path from the origin along to the opposite "corner". This path is referred to as the *Warping Path*, it length in turn can be used as a similarity measure for two time series.

DTW was first used as speech recognition technique to compare acoustic signals [14]. It has subsequently been adopted in the fields of data mining and machine learning [15]. Using DTW, the linearity of time series of different length is "warped" so that the sequences are aligned. Given two keystroke time series sub-sequences $s_1 = \{p_1, p_2, \ldots, p_i, \ldots, p_x\}$ and $s_2 = \{q_1, q_2, \ldots, q_j, \ldots, q_y\}$, where x and y are the lengths of the two series respectively, the two corresponding time series are used to constructed a matrix M of size $x \times y$. The value for each element $m_{ij} \in M$ is then computed by calculating the distance from each point $p_i \in s_1$ to each point $q_j \in s_2$:

$$m_{ij} = \sqrt{(p_i - q_j)^2} \tag{1}$$

A Warping Path ($WP = \{w_1, w_2, \ldots\}$) is then a sequence of matrix elements (locations), m_{ij}, such that each location is immediately above, to the right of, or above and to the right of, the previous location (except at the location opposite to the origin, which is the warping path end point). For each location the following location is chosen so as to minimise the accumulated warping path length. The "best" warping path is the one that serves to minimise the distance from $m_{1,1}$ to $m_{|s_1|,|s_2|}$. The idea is thus to find the path with the shortest *Warping Distance WD* between the two-time series

$$WD = \sum_{i=1}^{i=|WP|} if \quad m_i \in WP \tag{2}$$

WD is then an indicator of the similarity between the two keystroke time series under consideration. If $WD = 0$, the two time series in question are identical.

Figure 2 shows some results of the DTW process when applied to four of the time series sub-sequences used with respect to the evaluation of the proposed approach and reported on later in this paper. Figure 2(a) shows the *WP* for two keyboard time series sub-sequences from the same subject (user), while Fig. 2(b) shows the *WP* for two keyboard time series sub-sequences from two different subjects (users). The line included around the diagonal line in both figures indicates the WP that would have been obtained given two identical sub-sequences. The distinction between the generated WP in each can be observed from inspection of the figures.

6 Evaluation

In the foregoing a proposed KCA process was presented (Algorithm 1). Central to this KCA process was the ability to compare keyboard time series sub-sequences. The proposed mechanism for doing this was the DTW mechanism. To illustrate the

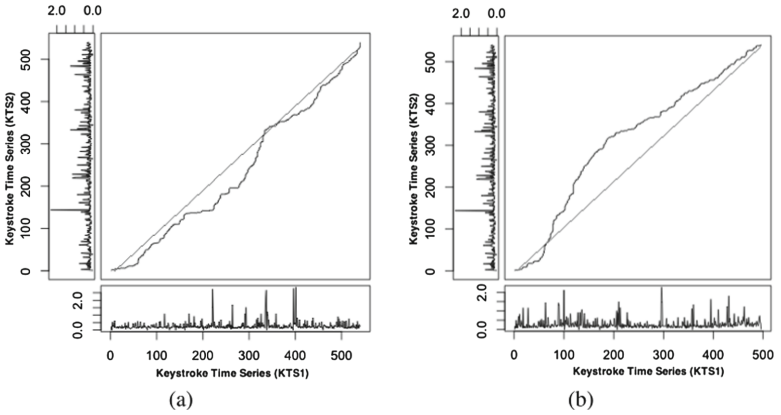

Fig. 2 Application of DTW: **a** Depicts WP for same subject where **b** Illustrates WP for different subjects

effectiveness of this mechanism this section presents the results obtained from a series of experiments conducted using DTW to compare keyboard time series. Two sets of experiments were conducted using a collection of 51 keyboard time series associated with 17 different subjects. The first set of experiments used DTW to compare time series in the collection; the objective was to determine how effectively DTW could be used to distinguish between time series. The second set of experiments compared the operation of the DTW technique with that when using a statistical feature vector based approach akin to that used in earlier work on KCA (see Sect. 2). The evaluation metrics used were: (i) accuracy, (ii) False Rejection Rate (FRR) (iii) False Acceptance Rate (FAR) and Mean Reciprocal Rank (MRR). The remainder of this evaluation section is organised as follows. We commence, Sect. 6.1, with a discussion of the data collection process. The outcomes from the experiments conducted to analyse the operation of the use of DTW within an overall KCA process are reported on in Sect. 6.2. A summary of the results obtained comparing the usage of DTW with the feature vector style approach, found in earlier work on KCA, is then presented in Sect. 6.3. Some discussion is present in Sect. 6.4.

6.1 Data Collection

Keystroke timing data was collected (in milliseconds) using a Web-Based Keystroke Timestamp Recorder (WBKTR) developed by the authors in JavaScript.[2] An HTML "front end" was used and subjects asked to provide answers to discussion questions. The idea was to mimic the situation where students are conducting online assessments. We therefore wished to avoid imposing constraints such as asking the subjects

[2]The interface can be found at: http://cgi.csc.liv.ac.uk/~hsaalshe/WBKTR3.html.

to use a specific keyboard or operating system. The idea was to allow subjects to type in the same manner as they would given an on line learning environment, in other words using different platforms, browsers and so on. JavaScript was used to facilitate the collection of data because of its robust, cross-platform, operating characteristics. This also offered the advantage that no third-party plug-ins were required to enable WBKTR to work. Another advantage of using JavaScript was that it avoided any adverse effect that might result from network delay when passing data to the "home" server, which might have affected the accuracy of recorded times, because the script function works at the end user station to record time stamp data within the current limitations of the accuracy end users's computer clock. The ability to paste text was disabled.

The subjects recruited were students and instructors working on online programmes (thus a mixture of ages). The data was collected anonymously. Additional information concerning the subjects was not recorded (such as gender and/or age). This was a deliberate decision so as minimise the resource required by subjects providing the data. Also because this data was not required, we are interested in comparing user typing patterns with themselves, not in drawing any conclusions about the nature of keyboard usage behaviour in the context of (say) age or gender. The subjects were also asked to type at least 100 words in response to each of three discussion questions (with no maximum limitation) so that adequate numbers of keystrokes could be collected. In [9] it was suggested that 100 keystrokes was sufficient for KCA (for convenience the WBKTR environment included a scripting function to count the number of words per question). Samples with a total number of keystrokes of less than 300 would have been discarded. During each session a JavaScript tool, with JQuery, transparently operated in the background to record the sequencing events KN and the flight time F^t between those events. A PHP script was used to store the identified attributes in the form of a plain text file for each subject. Once the keystroke data had been collected, time series were generated of the form described in Sect. 4 above. In this manner data from a total of 17 subjects were collected, three keyboard time series per subject, thus $17 \times 3 = 51$ time series in total. The data was used to create three data sets such that each data set corresponded to a discussion question. In the following the data sets are referred to using the letters (a), (b) and (c).

6.2 *Effectiveness of DTW for Keyboard Usage Authentication*

From the foregoing three keyboard time series data sets were collected. For the set of experiments used to determine the effectiveness of DTW each time series in each data set was compared with the time series in each other data set pair: (i) $a. \vee \{b, c\}$, (ii) $b. \vee \{a, c\}$ and (iii) $c. \vee \{a, b\}$.

For each experiment we refer to the first data set as data set 1, and the two comparator datasets as data sets 2 and 3. In each case we have 17 subjects numbered from 1 to 17. Consequently individual time series can be referenced using the notation

Table 1 Ranked average recorded *WD* values for DTW analysis ($a \vee b, c$), *correct matches* highlighted in bold font

	S1	S2	S3	S4	...	S13	S14	S15	S16	S17
All samples $(m-1)$ in the dataset a	0.042	0.084	0.057	0.064	...	**0.069**	**0.040**	0.035	**0.087**	**0.030**
	0.045	**0.085**	0.060	0.067	...	0.073	0.041	0.040	0.091	0.042
	0.048	0.090	0.062	**0.068**	...	0.074	0.042	0.041	0.093	0.042
	0.049	0.090	0.063	0.070	...	0.075	0.046	0.042	0.094	0.043
	0.050	0.094	0.064	0.071	...	0.076	0.048	0.042	0.095	0.044
	0.051	0.098	0.065	0.073	...	0.076	0.048	0.043	0.097	0.045
	0.053	0.099	0.065	0.075	...	0.078	0.049	0.043	0.099	0.045
	0.053	0.099	0.066	0.075	...	0.080	0.050	0.045	0.100	0.046
	0.055	0.100	0.067	0.078	...	0.081	0.054	0.045	0.101	0.046
	0.051	0.101	**0.068**	0.080	...	0.082	0.058	**0.047**	0.110	0.048
	...	...	...	...	...	...	...	...	...	...
Rank	**4**	**2**	**10**	**3**	...	**1**	**1**	**10**	**1**	**1**

Table 2 *WD* rankings for subjects when compared to themselves across datasets

Subjects	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17
Ranking list in Group a	4	2	10	3	4	5	5	2	4	2	3	10	1	1	10	1	1
Ranking list in Group b	1	4	10	15	3	3	2	2	2	2	6	3	1	1	4	1	1
Ranking list in Group c	1	10	10	5	1	5	1	3	10	1	3	6	1	2	7	5	3

s_{ij} where i is the data set identifier and j is the subject number. For each time series s_{1j} we compared with all time series in data sets 2 and 3 and a set of warping distance *WD* values obtained. For each pair of comparisons an average *WD* value was obtained and these values were then ranked in ascending order. Thus each comparison has a rank value r. The ranking outcome is shown in Table 1. Note that because of space limitation, the complete table is not shown in its entirely. In the table the rows and columns represent the subjects featured in the data sets. The values highlighted in bold font are the values where a subject is compared to itself. Ideally we would wish this comparison to be ranked first (recall that $WD = 0$ indicates an exact match). The last row in the table gives the ranking r' of each desired match (highlighted in bold font). All ranking values of corresponding samples of the same subject are listed in Table 2 with subjects represented by columns and data sets by the rows.

With respect to the above, the overall accuracy was computed as the ratio between the number of *incorrect matches* ℓ ranked prior to a *correct match* ($\ell = \sum_{i=1}^{i=m} \sum_{j=1}^{j=n} (r'_{ij} - 1)$ where: (i) m is the number of data sets, (ii) n is the number of subjects and (iii) r'_{ij} is the ranking of the desired match for subject s_{ij}). The total number of comparisons τ is given by $\tau = m \times n$. Thus, with respect to the results presented in Tables 1 and 2, $\ell = 52$ and $\tau = 867$, giving an accuracy of 94.00 % ($\frac{867-52}{867} \times 100 = 94.00$).

The False Rejection Rate (FRR) and False Acceptance Rate (FAR) were also calculated with respect to the outcomes of the time series analysis of typing patterns presented above. According to the European Standard for access control, the acceptable value for FRR is 1 %, and that for FAR is 0.001 % [16]. Thus, we used the FRR and FAR metrics to measure how far the operation of our proposed method, as a biometric authentication method, compared with this standard. For each average *WD* comparison, in each dataset grouping, we calculated FRR by computing the number of subjects n where the corresponding desired rank r' did not equal to 1, $\sum_{j=1}^{j=n} r'_j \neq 1$. If the equivalent sample's rank is not equal to 1, this means that the sample has been falsely rejected. In contrast, FAR is calculated by computing the number of subjects that are ranked higher than the desired sample. The subjects ranked before the desired subject can be considered to have been accepted as real users. Average FRR and FAR values, across the three data set groupings, of 5.99 % and 1.48 % were obtained.

6.3 *Effectiveness of Feature Vector Approach for Keyboard Usage Authentication*

To compare the operation of our proposed approach with the statistical feature vector style of operation found in earlier work on KCA (see Sect. 2), the keystroke data was used to define a feature vector representation. This was done by calculating the average flight time $\mu(f^t)$ for the most frequently occurring di-graphs found in the data

$$\mu(f^t) = \frac{1}{d} \sum_{i=1}^{i=d} Ft_i \tag{3}$$

where d is the number of identified frequent di-graphs and F^t is the flight time value between the identified di-graphs. Each identified di-graph was thus a feature (dimension) in a feature space with the range of average $\mu(f^t)$ values as the values for the dimension. In this manner feature vectors could be generated for each sample. The resulting representation was thus similar to that found in more traditional approaches to KCA [6, 9, 11, 13].

In the same manner as described in Sect. 6.2, we measured the similarity of each feature vector in the first data set with every other feature vector in the other two data sets using Cosine Similarity (CS) (note that this was done for all three data pairings as before). The CS values were calculated using Eq. 4, where $x \cdot y$ is the dot product

Table 3 Ranking lists of all samples in different groups when applying CS

Subjects	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17
Ranking list in Group *a*	5	20	8	13	20	2	20	10	25	20	14	15	1	3	8	1	1
Ranking list in Group *b*	5	18	15	13	15	2	20	11	25	18	20	10	2	3	8	8	5
Ranking list in Group *c*	4	19	7	7	20	8	18	3	15	20	20	13	1	8	10	5	1

between two feature vectors x and y, and $||x||$ ($||y||$) is the magnitude of the vector x (y). Note that in case of using CS, the feature vectors need to be of the same length (unlike in the case of DTW). As before averages for pairs of CS values were used, and as before the values were ranked but in this case in descending order ($CS = 1$ indicates a perfect match).

$$CS(x, y) = \frac{x \cdot y}{||x|| \times ||y||} \tag{4}$$

Table 3 presents an overview of the ranking results obtained using the feature vector approach in the same way as Table 2 presented an overview of the DTW rankings obtained. Detection accuracy was calculated in the same way as for the DTW experiments reported above. In this case $\ell = 163$ and, as before, $\tau = 867$. Thus an accuracy of 81.20 % ($\frac{867-163}{867} \times 100 = 81.20$). The average FRR and FAR values obtained using the feature vector representation, calculated as described above, 20.59 % and 1.69 % respectively.

6.4 Discussion

In the foregoing three sub-sections both the proposed DTW based and the established feature vector based approaches to keyboard authentication were analysed in terms of accuracy, FRR and FAR. A summary of the results obtained is presented in Table 4 with respect to the three data set combinations considered. Mean values are included at the bottom of the table together with their associated Standard Deviation (SD). The table also includes Mean Reciprocal Rank (MRR) values for each approach and data set combination. MRR is an alternative evaluation measure that can be used to indicate the effectiveness of authentication systems [17]. MRR is a standard evaluation measure used in Information Retrieval (IR). It is a measure of how close the position of a desired result (identification) is to the top of a ranked list. MRR is calculated as follows:

Table 4 Results obtained by representing keystroke features in the two approaches: (i) *Time series*, and (ii) *Feature vector*

Representation method	Time series				Feature vector			
	Metrics							
Dataset	FRR (%)	FAR (%)	MRR	Acc (%)	FRR (%)	FAR (%)	MRR	Acc (%)
$Group(a) \rightarrow a. \vee \{b, c\}$	6.11	1.52	0.438	93.88	20.58	1.64	0.283	79.41
$Group(b) \rightarrow b. \vee \{a, c\}$	5.17	1.41	0.520	94.82	21.64	1.76	0.155	78.35
$Group(c) \rightarrow c. \vee \{a, b\}$	6.70	1.51	0.454	93.29	19.17	1.64	0.225	80.82
Mean	**5.99**	**1.48**	**0.471**	**94.00**	20.59	1.69	0.221	79.53
SD	**0.77**	**0.06**	**0.04**	**0.77**	1.24	0.07	0.06	1.24

$$MRR = \frac{1}{|Q|} \cdot \sum_{i=1}^{|Q|} \frac{1}{r_i} \tag{5}$$

where: (i) Q is a set of queries (in our case queries as to whether we have the correct subject or not), and (ii) r_i is the generated rank of the desired response to Q_i. An MRR of 1.00 would indicate that all the results considered are correct, thus we wish to maximise MRR. With reference to Table 4, the average MRR with respect to the proposed time series based approach to KCA proposed combination, is 0.471; while for the feature vector based approach it is 0.221. Returning to Table 4, inspection of the results indicates that the proposed DTW based approach to keyboard authentication outperforms the exciting feature vector based approach by a significant margin.

7 Conclusion

An approach to KCA using time series analysis has been presented that takes into consideration the ordering of keystrokes. The process operates by representing keystroke timing attributes as discrete points in a time series where each point has a timestamp of some kind and an attribute value. The proposed representation used a sequential keypress numbering system as the time stamp, and flight time as the attribute (distance between key presses). DTW was adopted as the time series comparison mechanism. For evaluation purposes data was collected anonymously using a bespoke web-based tool designed to mimic the process of conducting online assessments (responding to discussion questions). The evaluation was conducted by comparing every subject to every other subject to determine whether we could distinguish between the two using: (i) the proposed technique and (ii) a feature vector based approach akin to

that used in established work on KCA. With respect to the first set of experiments, an overall accuracy of 94.00 % was obtained. This compared very favourably with an accuracy of 79.53 %, obtained with respect to the second set of experiments. The results demonstrated that the proposed time series based approach to KCA had significant potential benefit in the context of user authentication with respect to online assessments such as those used in online learning and MOOCS. The authors belief that further improvement can be realised by considering n-dimensional time series (time series that consider more than one keystroke attribute). Future work will also be directed at confirming the findings using larger datasets.

Acknowledgments We would like to express our thanks to those who participated in collecting the data and to Laureate Online Education b.v. for their support.

References

1. Gaines, R.S., Lisowski, W., Press, S.J., Shapiro, N.: Authentication by keystroke timing: some preliminary results, no. RAND-R-2526-NSF. RAND Corp., Santa Monica, CA (1980)
2. Bleha, S., Slivinsky, C., Hussien, B.: Computer-access security systems using keystroke dynamics. IEEE Trans. Pattern Anal. Mach. Intell. **12**(12), 1217–1222 (1990)
3. Joyce, R., Gupta, G.: Identity authentication based on keystroke latencies. Commun. ACM **33**(2), 168–176 (1990)
4. Ogihara, A., Matsumuar, H., Shiozaki, A.: Biometric verification using keystroke motion and key press timing for atm user authentication. In: Intelligent Signal Processing and Communications, 2006. ISPACS'06, pp. 223–226. IEEE
5. Syed, Z., Banerjee, S., Cukic, B.: Normalizing variations in feature vector structure in keystroke dynamics authentication systems. Softw. Qual. J. 1–21 (2014)
6. Ahmed, A.A., Traore, I.: Biometric recognition based on free-text keystroke dynamics. IEEE Trans. Cybern. **44**(4), 458–472 (2014)
7. Bours, P.: Continuous keystroke dynamics: a different perspective towards biometric evaluation. Inf. Secur. Tech. Rep. **17**(1), 36–43 (2012)
8. e Silva, S.R.D.L., Roisenberg, M.: Continuous authentication by keystroke dynamics using committee machines. In: Intelligence and Security Informatics, pp. 686–687. Springer, Berlin (2006)
9. Gunetti, D., Picardi, C.: Keystroke analysis of free text. ACM Trans. Inf. Syst. Secur. (TISSEC) **8**(3), 312–347 (2005)
10. Messerman, A., Mustafic, T., Camtepe, S.A., Albayrak, S.: Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics. In: 2011 International Joint Conference on Biometrics (IJCB), pp. 1–8. IEEE (2011)
11. Shepherd, S.J.: Continuous authentication by analysis of keyboard typing characteristics. In: European Convention on Security and Detection, 1995, pp. 111–114. IET (1995)
12. Richardson, A., Kaminka, G.A., Kraus, S.: REEF: resolving length bias in frequent sequence mining using sampling. Int. J. Adv. Intell. Syst. **7**(1), 2 (2014)
13. Dowland, P.S., Furnell, S.M.: A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In: Security and Protection in Information Processing Systems, pp. 275–289. Springer, US (2004)
14. Rabiner, L., Juang, B.H.: Fundamentals of speech recognition. Prentice Hall (1993)
15. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD Workshop, vol. 10, no. 16, pp. 359–370

16. Polemi, D.: Biometric techniques: review and evaluation of biometric techniques for identification and authentication, including an appraisal of the areas where they are most applicable. Reported prepared for the European Commision DG XIIIC, 4 (1997)
17. Craswell, N.: Mean reciprocal rank. In: Encyclopedia of Database Systems, pp. 1703–1703. Springer, US (2009)