KI - Künstliche Intelligenz (2018) 32:231–243
https://doi.org/10.1007/s13218-018-0526-z

TECHNICAL CONTRIBUTION

Iterative Keystroke Continuous Authentication: A Time Series Based Approach

Abdullah Alshehri[1] · Frans Coenen[1] · Danushka Bollegala[1]

Published online: 18 January 2018

Abstract
Keyboard typing patterns are a form of behavioural biometric that can be usefully employed for the purpose of user authentication. The technique has been extensively investigated with respect to the typing of fixed texts such as passwords and pin numbers, so-called static authentication. The typical approach is to compare a current "typing sample" with a typing template expressed in terms of a feature vector comprised of keystroke dynamics. The feature vector approach has also been applied in the context of continuous authentication where features are extracted from free typing. However, the use of feature vectors for keystroke continuous authentication entails a number of disadvantages, mostly associated with the size of the feature vectors and their generation, which need to capture a large number of features to be effective; thus making the technique unsuitable for iterative (real-time) authentication as would be required in the case of, for example, online assessments. To address this issue, a mechanism whereby iterative real-time keystroke continuous authentication can be achieved is proposed, by considering typing behaviour as a form of time series, that avoids the disadvantages associated with the feature vector approach. The reported experimental results show a significantly improved performance using the proposed method in comparison with the feature vector based technique.

Keywords Keystroke Data Streams · Keystroke Time Series · Continuous Authentication

1 Introduction

The increasing prevalence of internet-facilitated distance learning (eLearning, Massive Open Online Courses and so on) has raised a requirement for techniques whereby the claimed identity of remote users can be authenticated. One frequently used authentication mechanism is through the use of usernames and passwords [1]; thus "once only" authentication. However, in the case of online assessments and exams (and other applications) there is a requirement to monitor the identity of users throughout the course of an entire assessment, thus "iterative" continuous authentication. One solution is to use some form of biometric such as continuous iris recognition or fingerprint recognition, but this requires specialist equipment and technology not readily available to the typical distance learner.

Keystroke dynamics (typing patterns) are a promising behavioural biometric for continuous authentication; it has been shown that individuals have distinctive keyboard usage styles [2, 3]. Early work of the usage of typing patterns for authentication was directed at the static context. For example, static authentication with respect to the typing pattern, the rhythm, found when users typed in their credentials (password and username, or pin number) [4–7]. This can be referred to as Keystroke Static Authentication (KSA), static in the sense that the keys being pressed are predefined, or simply fixed. The alternative is Keystroke Continuous Authentication (KCA) where we are interested in patterns resulting from the typing of free text [3, 8–11]. It is the later we are interested in when monitoring distance learners undertaking online assessments and exams. However, iterative KCA is a non-trivial task whereby the adopted system has to recognise typing patterns as typing progresses and regardless of what text is actually being typed. It should also be noted that keystroke dynamics, although frequently used

✉ Abdullah Alshehri
a.a.alshehri@liverpool.ac.uk

Frans Coenen
coenen@liverpool.ac.uk

Danushka Bollegala
danushka.bollegala@liverpool.ac.uk

1 University of Liverpool, Liverpool L69 3BX, UK

for KSA and KCA, has also been used in other contexts, such as detecting keyboard user emotions [12]

Biometric techniques, in general, comprise a two stage process [13]: (1) enrolment and (2) verification. The first is concerned with the creation of an enrolment database, and the second with using that database for authentication or identification purposes. Typically, the study of keystroke dynamics, as a biometric, is concerned with the timing information generated from key presses and releases. The basic timing information used is: (1) flight time (F^t), the time between n consecutive key presses; and (2) hold down time (HD^t), the length of time between a key press and a key release. Note that in the context of flight time, when $n = 1$ we talk of monographs, when $n = 2$ we talk of digraphs, when $n = 3$ we talk of trigraphs, and so on. Instead of flight time the terms *latency* [5] and *duration* [14] are also sometimes used in the literature. This timing information is utilised to construct distinctive typing biometric templates for both enrolment and verification purposes. In the context of KSA and KCA such templates are also referred to as signatures [11] or reference profiles [15].

To date, the most common mechanism used to build individual templates is by constructing feature vectors based on keystroke timing information, regardless of whether KSA or KCA is being considered. Such feature vectors usually comprise statistical values, for example, the average and standard deviation of hold or flight times. In the context of KSA, the template is constructed for a fixed (static) text; enrolled users are usually requested to repeatedly type the fixed text several times. A previously unseen typing profile can then easily be verified through comparison with the stored typing patterns using some statistical mechanism. In the context of KCA, however, the construction of templates is more challenging. The is because, by definition, the text to be considered is unstructured; we do not know in advance the expected sequence of key presses. This, in turn, means that our templates need to be more generic, and consequently more sophisticated. A common mechanism for defining KCA feature vector templates is to identify statistical details concerning the most frequently occurring sequences of keystrokes, n-graphs [8, 9]. For example, if `ea` and `th` are the most frequently occurring digraphs in a sample provided by a user, a typing template can be constructed using the means and standard deviations of the flight times for these digraphs. Consequently, for verification purposes, whenever a typing sample is received we can search for templates whose statical similarity (matching score) falls within a global threshold; if found the sample is recognised as a real sample, otherwise it is rejected. A criticism directed at this approach is that the template profile might not feature the same frequently occurring n-graphs as the samples to be authenticated, which in turn can lead to poor verification rates. A suggested solution is to increase the number of training n-graphs considered to cover all possibilities, which in turn means that the user needs to be asked to provide more samples. An obvious question is how many n-graphs do we require to ensure that a template is sufficiently robust? Whatever the answer, the number of n-graphs, and hence the size of the required sample, is significant.

In this paper, an alternative approach to representing typing patterns, in the context of free text, for iterative KCA, is proposed. The approach uses all information from all keystrokes and not just selected n-graphs (thus a global perspective), without taking into consideration the actual keys pressed (what might be termed the local perspective). More specifically, typing activity is conceptualised in terms of a continuous data stream, a *time series*, comprised of a sequence of press-and-release temporal events $\{p_1, p_2, \dots\}$. Thus we can generate subsequences of keystroke time series. The intuition is that these time series will feature keystroke dynamic patterns unique to individual users. It is therefore conjectured that the shape of keystroke time series can provide insights to typing patterns, from free text, which can then be used for the purpose of iterative KCA. Thus by comparing a previously unseen keystroke time series, that is claimed to belong to a particular user, with stored reference templates (profiles) held in an enrolment database, that are known to belong to the claimed user, authentication can take place.

Generally speaking, there are various ways whereby time series can be compared. With respect to the work presented in this paper Dynamic Time Warping (DTW) was adopted for reasons that will become clear later in the paper. For evaluation purposes three datasets were used: (1) the ACB dataset generated by the authors, (2) the GP dataset from [10] and (3) the VHHS dataset from [16]. It should also be noted that the proposed mechanism presented in this paper is an extension of previous work conducted by the authors [17, 18]. The distinction between the work presented in [17, 18] and that presented here, is that, although the idea of using time series for KCA was proposed in both [17, 18] (multivariate time series in the case of the later), in both papers the idea was only considered in the once-only context, not in the iterative context as in the case of this paper.

The remainder of this paper is structured as follows. In Sect. 2 some related work concerning KCA is presented. This is followed in Sect. 3 with some definitions and a formalism concerning keystroke time series analysis. Section 4 discusses the DTW process, the mechanism adopted with respect to the work presented in this paper for comparing keystroke time series. The proposed iterative KCA process is then presented in Sect. 5. The evaluation of the proposed approach is reported on and analysed in Sect. 6. Finally, the paper is concluded with a summary and some recommendations for future work in Sect. 7.

2 Previous Work

From the literature, and as noted above, we can broadly identify two categories of keystroke authentication defined as follows:

1. *Keystroke static authentication* (KSA) Authentication directed at static (predefined) text such as passwords, user names, and pin numbers.
2. *Keystroke continuous authentication* (KCA) Authentication direct at continuous (free) text.

The definitions are consistent with the definitions frequently used in the literature, although not in all cases. The work presented in this paper is directed at KCA. In this paper, we also use the terms *once only* and *iterative* to describe the nature of the authentication. The term once only is used to refer to "one-time-only" authentication applied once typing has been completed, regardless of whether we are considering KSA or KCA. The term iterative then refers to a process where we are repeatedly conducting the authentication whilst typing is taking place, preferably in real-time, as required when monitoring online assessments and examinations. It does not make sense to apply iterative authentication in the context of KSA; thus the term iterative authentication is assumed to apply to KCA only; we, therefore, talk of iterative KCA. From the literature, however, most existing work on KCA has been directed at once only authentication, authentication conducted when typing has been completed.

A distinction should also be made between keystroke *user identification* and keystroke *user authentication*. In the first case, the objective is to identify a user, typically with respect to an enrolment database of user keystroke templates (profiles). For example, in an access control situation where all users key-in the same access code, we might wish to apply KSA to identify the user as an extra security precaution. If the keyboard pattern is not in our database access can be denied. In the case of user authentication, we wish to check that the user is who they say they are, not to identify them. For example, in the case of a banking scenario, we might wish to apply KSA in the context of an entered pin number which, it is claimed, belongs to a particular user, again as an extra security precaution. There is thus a distinction between user identification and authentication. The work presented in this paper is directed at user authentication, particularly in the context of users undertaking online assessments and examinations. Therefore, in this paper, we are considering iterative user authentication using KCA.

As noted from the introduction to this paper, most existing work on KSA and KCA, uses a feature vectors representation where the features stored are statistical quantitative equivalents of typing features (hold time and flight time). For the purpose of authentication, the similarity between such vectors can be ascertained by measuring the "distance" between vectors, for example using the cosine similarity measure. However, as suggested in this paper, this approach may not be the ideal method for representing free text typing patterns.

Much existing work on KCA (and KSA) is also directed at one time only identification and authentication; there has been very little work directed at iterative KCA. Some notable exceptions where iterative KCA has been considered can be found in [8–11]. In [8] a training set was used to generate a feature vector represented typing template repository which was then used to "identify" users (see also [19]). The feature vectors were constructed by computing the flight time means of all digraphs that featured in the training set. The iterative user identification was then conducted by repeatedly generating "test" feature vectors for a given user, one every minute, and comparing with the stored templates. If a statistically similar match was found this indicated the typer's (user's) identity. For evaluation purposes a kNN approach was used where $k = 1$. Experiments were also conducted using a number of different methods for comparing feature vectors: Euclidean distance, Mahalanobis, probability, and weighted probability. The disadvantage of the approach, however, was the size of the feature vectors (a great number of digraphs were required) and the number of stored templates. To minimise the search complexity, the authors proposed a clustering mechanism so only the most relevant cluster had to be searched in detail. However, this then meant that re-clustering was required every time a new user was added. The overall reported accuracy, in the context of user identification and iterative KCA, was 23%; not, it is argued here, a good result.

In [9] digraph latency (flight time) was used for the construction of feature vectors. Each feature vector was generated by considering the first 500 digraphs and trigraphs in the input typing sample, and the most frequently occurring 2000 keywords in the English language and determining the associated latency (flight) times. Valid latency times had to be within the range 10–750 ms. The mean and standard deviation (SD) of each digraph, trigraph and keyword were calculated and n-graphs with SD values in the top and bottom 10% pruned so as to remove n-graphs that had large SD's. During authentication, potential imposter samples were compared with a stored template and an "alert criterion" adjusted accordingly. A deviation (threshold) value was then used to identify imposters. For evaluation of the process, a simulated environment was used. The metric used to measure the performance of the system was the False Acceptance Rate (FAR). Experiments were conducted using digraphs, trigraphs, and keywords independently and

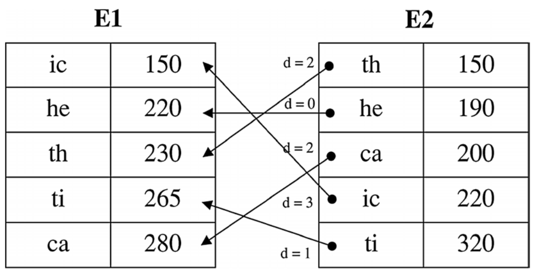

Fig. 1 An example showing how the similarity is computed using the degree of disorder as proposed in [10] (the figure has been taken from the original study [10])

in combination. Best results were obtained using digraphs. The reason trigraphs and keywords did not work well was because trigraphs did not appear as frequently as digraphs, and many keywords did not appear at all.

The study presented in [10] is one of the most promising studies that deal with free text analysis. In this study a feature vector representation was again used, however, in this case using the latency values (flight times) associated with the entire shared set of n-graphs included in the evaluation dataset. The similarity between two typing samples was determined as follows. The latency times of all shared n-graphs in the two samples were extracted, and ordered (in ascending order of latency time) in two arrays. The difference between the order numbering of each n-graph in each array, the counterpart distance d, was then computed and summed to give a *degree of disorder* value.[1] This was then used as a similarity measure, the smaller the degree of disorder the more similar the two typing samples. The process is illustrated in Fig. 1 (taken from [10]). The figure shows five digraphs, `ic-he-th-ti-ca`, that feature across two typing samples E_1 and E_2. The digraphs are ordered according to latency time. The associated counterpart distances are then $\{2, 0, 2, 3, 1\}$ respectively. The similarity, *sim*, between the two samples can then be computed as:

$$sim(E1, E2) = \frac{\sum_{i=1}^{n} d_i}{(n^2 - 1)/2},$$

where n is the number of shared digraphs (five in the example). In the study, the authentication process was evaluated by comparing a new sample to a collection of samples belonging to each enrolled users. For each user, an average degree of disorder value was obtained and the user with the lowest average value selected. This will be a computationally expensive process given an enrolment database of any size. In the reported evaluation, 600 reference templates were considered (generated from 40 users, each with 15 sample one of which was used as a previously unseen sample); the time taken for a single match was thus substantial. Moreover, each typing sample comprised between 700 and 900 keystrokes, so the average template size of each user consisted of 11,200 ($14 \times 800 = 11200$) keystrokes. Another disadvantage of the approach was that a large number of n-graphs was required so that accurate authentication results could be obtained.

In [11] a mechanism was proposed to mitigate against the expense (in terms of time and size) of pattern extraction and template construction for iterative KCA. The idea was to use an Artificial Neural Network (ANN) to predict missing n-graphs based on the available data that subjects provided. A feature vector representation was again used. The features used were key-down time and average digraph and monograph flight time. The ANN classifier was then used to build a prediction model with which to conduct user authentication. This mechanism worked reasonably well in a controlled experimental setting; typing of the same text using the same keyboard layout in an allocated environment. However, this is not the situation that will be encountered in the context of the real world, where iterative KCA is expected to operate.

From the above, it can be observed that most existing KCA studies have been directed at the usage of quantitative statistical measures to represent n-graph timing information which has been encapsulated in a feature vector format. A general criticism of this principle is the number of features that need to be included. Some of the previous work summarised above seeks to mitigate against this in various ways; a consequent argument is that not all the available data is used. Therefore, it is conjectured that representing keystroke dynamics using time series (streams) can lead to more effective KCA. To the best knowledge of the authors, there has been no prior work in the literature, other than that of the authors [17, 18], that has considered the concept of a time series representation for iterative KCA. The only reference that the authors are aware of is [20] where a streaming algorithm is introduced for which a potential suggested application domain is KCA.

It can be argued that the proposed time series based method does not use all the available data in the sense that the actual keys that have been pressed are not recorded. The criticism that not all the available data is used may thus also be directed at the proposed system. However, the excellent results obtained (see Sect. 6) indicate the effectiveness of the proposed technique whilst, at the same time, realising efficiency advantages over the previously reported work on KCA; this suggests that in the case of the proposed system the fact that information concerning the actual keys is not used provides efficiency benefits that outweigh any argued disadvantage of not using all the data. In summary, the distinction, between the proposed time series based method and previous work, can be said to be that the proposed method

[1] An idea inspired by Spearman's rank correlation coefficient.

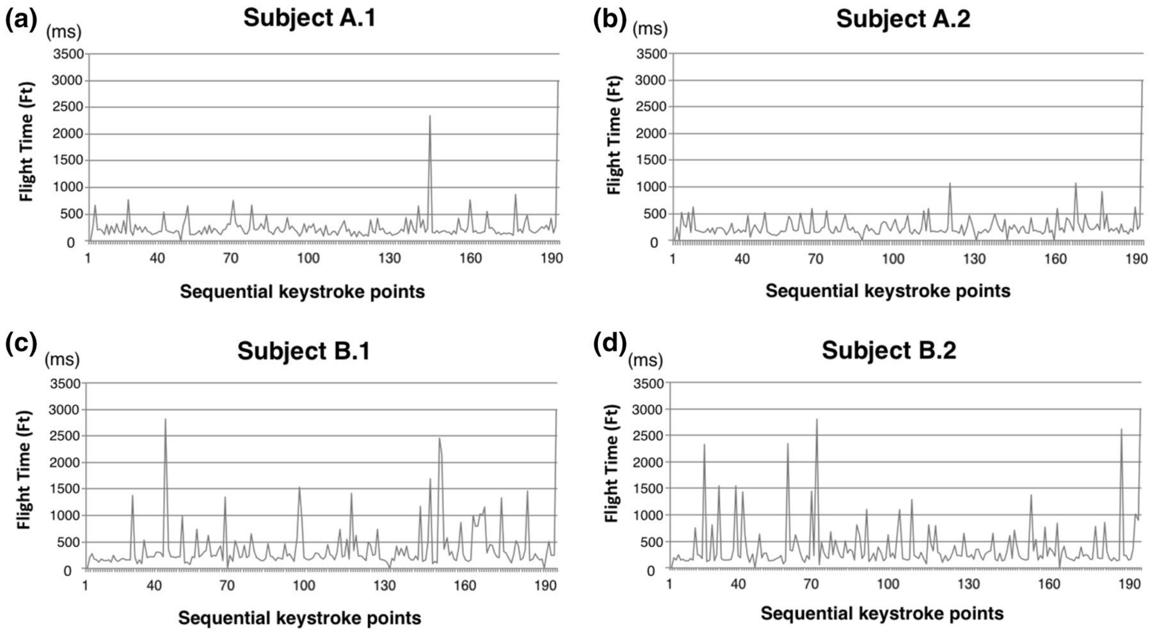

Fig. 2 Examples of keystroke time series: **(a)** and **(b)** time series for Subject A writing two different texts; **(c)** and **(d)** time series for Subject B writing two different texts

uses what might be termed "global features", whilst the previous methods use what might be termed "local features".

3 Keystroke Time Series Representation

Keystroke dynamics are key press-and-release temporal events. In combination, these events describe characteristic typing patterns.

Definition 1 A keystroke time series K is an ordered discrete sequence of points p; $K = \{p_1, p_2, \dots, p_n\}$ where n is the length of the entire series, and each point p_i is a keystroke event.

For each keystroke press we can obtain four timing values: (1) Key-down time KD^t, (2) Key-up time KU^t, (3****) Key-hold time KH^t and (4) flight-time F^t. For any keystroke i, KH^t_i can be calculated using $KH^t_i = KU^t_i - KD^t_i$. The value for F^t_i can then be obtained from $F^t_i = KU^t_{i-1} - KD^t_i$. Since KH^t and F^t incorporate other values these are considered the most important keystroke features. Thus any keystroke can be described by a tuple of the form $\langle KH^t, F^t \rangle$. Using both values together in a time series representation results in a multidimensional time series; consequently, in this paper, only F^t was used. The issue of multidimensional keystroke time series will be considered in future work for iterative KCA. Thus, in this paper, $K = \{F^t_1, F^t_2, \dots\}$, a series of flight time points.

Definition 2 A keystroke time series subsequence s, of length l, is a subsequence of K that starts at the point p_i within K and ends at point p_{i+l-1}, thus $s = \{p_i, p_{i+1}, \dots, p_{i+l-1}\}$.

A subsequence s of K is indicated using the notation $s \preceq K$ ($\forall p_i \in s, \exists p_j \in K$ such that $p_i \equiv p_j$).

Definition 3 A user template (profile) $\mathcal{U}$ is a set of m previously recorded keystroke time series subsequences $\mathcal{U} = \{s_1, s_2, \dots, s_m\}$.

The template $\mathcal{U}$ will be stored in what in the field of biometrics is called a user enrolment database. The usage of $\mathcal{U}$ will become clear later in this paper.

Figure 2 gives four example time series, using flight time (F^t), for continuous (free) texts, two for subject A and two for subject B, taken from the ACB evaluation dataset generated by the authors and used with respect to the evaluation reported on later in this paper in Sect. 6. From the figure, it can be observed that the keystroke time series belonging to the same subject have clear similarities despite the series being related to different texts. In contrast, the keystroke time series associated with different subjects have clear dissimilarities.

The fundamental iterative KCA process presented in this paper operates as follows. On start up an initial subsequence s_1 is compared with the user profile $\mathcal{U}$ so as to determine whether the user is who they say they are. The decision is made using a similarity threshold value tailored to the user's typical typing behaviour (further detail concerning the generation of this threshold is presented later in Sect. 5.3). Assuming a positive result, each subsequent subsequence s_k (where $k > 1$) is compared with the preceding, previously collected, subsequence s_{k-1}; in this way changes in typing behaviour (if any) can be detected.

4 Measuring Keystroke Time Series Similarity

The most significant element of the proposed iterative KCA mechanism presented in this paper is the process whereby pairs of keystroke time series subsequences are compared, either with the keystroke time series subsequences held in $\mathcal{U}$ or previously identified subsequences in the current data stream. Given two keystroke time series subsequence s_1 and s_2 of the same length, the simplest way to define their similarity is in terms of the Euclidean Distances (ED) between each point in s_1 and the corresponding point in s_2. However, the ED measurement does not consider the "offsets" (phase and amplitude differences) that might exist in a given time series pair. This can be illustrated by considering the subsequences given in Fig. 2a, b. Inspection of these keystroke time series indicates that "shapelets", or simply the shape, within the two series are similar, but that the "peaks" and "troughs" are offset to one another. ED measurement will not capture this noticeable similarity.

To cope with this issue, a DTW mechanism has been adopted instead. This is a well-established method [21, 22], which has been used effectively to find the similarity between pairs of point (time) series. It has been adopted in many domains such as speech recognition [23], time series data mining [24, 25] and pattern recognition [26, 27]. DTW serves to warp the linearity of sequences (even of different lengths) so that any phase shifting can be taken into consideration.

In more detail, the operation of DTW can best be described by considering two time series $s_1 = \{p_1, p_2, \dots, p_i, \dots, p_x\}$

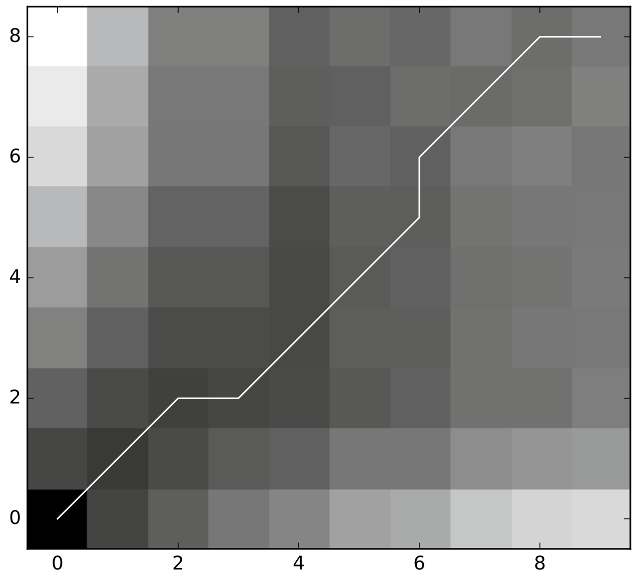

Fig. 3 Example minimum WP obtained using DTW applied to two keystroke time series subsequences generated by the same subject writing different texts

and $s_2 = \{q_1, q_2, \dots, q_j, \dots, q_y\}$, where x and y are the lengths of the two series respectively. In the case of the proposed KCA mechanism presented in this paper the time series are keystroke time series and the values represented by each point $p_i \in s_1$ and each point $q_j \in s_2$ are flight time values (F^t). A matrix M of size $(x-1) \times (y-1)$ is then constructed whereby the value held at each cell $m_{i,j} \in M$ is the distance from point $p_i \in s_1$ to point $q_j \in s_2$:

$$m_{i,j} = \sqrt{(p_i - q_j)^2}. \tag{1}$$

The matrix M is used to determine a minimum *warping distance* (*wd*) which is then used as a similarity measure. A *wd* is the accumulated sum of the values associated with a *Warping Path* (*WP*) from cell $m_{0,0}$ to cell $m_{x-1,y-1}$. A warping path is a sequence of cell locations, $WP = \{w_1, w_2, \dots, w_i\}$, such that given $w_k = m_{i,j}$ the follow on location is either $m_{i+1,j}$, $m_{i,j+1}$ or $m_{i+1,j+1}$. The *wd* associated with a particular *WD* is then the sum of the values held at the locations in *WD*:

$$wd = \sum_{i=1}^{|WP|} w_i \in WP. \tag{2}$$

To arrive at a minimum *wp*, for each location the following location is chosen so as to minimise the accumulated *wd*.

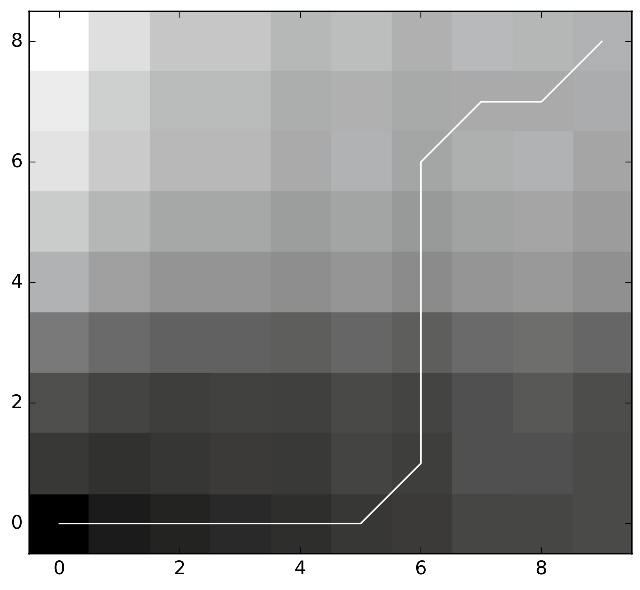

Fig. 4 Example minimum WP obtained using DTW applied to two keystroke time series subsequences generated by the different subjects writing the same texts

The “best” warping path is thus that which serves to minimise the distance from $m_{0,0}$ to $m_{x-1,y-1}$. The minimum *wd* for a pair of time series can, therefore, be interpreted as an indicator of the similarity between the two time series. Note that if $wd = 0$ the two keystroke time series in question will be identical. Further detail concerning the DTW mechanism can be found in [21, 22].

For ease of understanding, the DTW process is illustrated in Figs. 3 and 4. Figure 3 shows the warping path that results when the DTW process is used to compare two keystroke time series subsequences produced by the same subject but writing different texts, whilst Fig. 4 shows the warping path that results when the DTW process is used to compare two keystroke time series subsequences produced by different subjects but writing the same texts. The white line included in the figures indicates the minimum WP. The distinction between the generated WPs can be observed from the inspection of the figures. It is also worth noting here that the DTW concept has similarities with Levenshtein Distance calculation [28], also referred to as Edit Distance, used for measuring the similarity between two strings. However, in this case, the values used are the number of deletions, insertions or substitutions required to transform the first string into the other.

5 The Keystroke Continuous Authentication Process

The proposed iterative KCA process, founded on the idea of time series analysis as described above, is presented in this section. The fundamental idea is that, as typing proceeds, we repeatedly collate keystroke usage subsequences of length ω and compare, on start up, with the subsequences in $\mathcal{U}$; and then, once the process is underway, with the previously collected subsequence to the current subsequence.

The basic process is given by the pseudo code in Algorithm 1. The algorithm takes as input: (1) the frequency f with which keystroke time series subsequences are collected, (2) the length ω of a collected subsequence and (3) a similarity threshold σ value. Note that f can be set so that subsequences (windows) overlap ($f < \omega$), subsequences abut ($f = \omega$) or subsequences are spaced ($f > \omega$). For the evaluation presented in Sect. 6 $f = \omega$ was used. More specifically a range of ω values was considered from 25 to 150 incrementing in steps of 25 ($\{25, 50, 75, 100, 125, 150\}$).

On start up the sets s_1 and s_2 are initialised to $\emptyset$ (lines 1 and 2), a keystroke counter t is initialised with the value 0 (line 3) and a Boolean flag *isCollectingTS*, indicating whether a subsequence is in the process of being collected or not, initialise to the value *false*. The procedure then operates on a continuous loop. On each iteration a single key press is processed; there are five options: (1) stop processing as the end of the data stream has been reached (lines 7 and 8), (2) start collating a new subsequence (lines 9 to 11), (3) subsequence collation is complete therefore test the subsequence (lines 12 to 23), (4) add the current keystroke to the current subsequence (lines 24 and 25) or (5) do nothing (the process is between subsequence collection phases). At the end of each iteration, the keystroke counter is incremented by one (line 27).

Whenever a subsequence of length ω has been obtained the algorithm tests whether the subsequence contains noise using the function *noiseReduction*() (line 13), how this operates is described in greater detail in Sect. 5.1 below. If there is no previous subsequence the current subsequence is compared to the content of $\mathcal{U}$ to confirm the claimed identity of the user (lines 14 and 15). How this is achieved is described in further detail in Sect. 5.2. Otherwise DTW is applied using the function $dtw(s_1, s_2)$ (line 17). As noted above, DTW generates a warping distance *wd*, if this distance is greater than the threshold σ this is “highlighted” (lines 18 and 19). How this threshold is determined is described in Sect. 5.3 below.

Algorithm 1 Itterative KCA process

```
Input: $f$ = sampling frequency, $\omega$ = time series subsequence length,
    $\sigma$ = similarity threshold
Output: Similarity "highlight" reporting
 1: $s_1 = \emptyset$
 2: $s_2 = \emptyset$
 3: $t = 0$
 4: $isCollectingTS = false$
 5: loop
        ▷ Get next point
 6:     $p_t$ = Next point in data stream
        ▷ End of input stream
 7:     if $p_t$ == End of data stream marker then
 8:         Exit loop
        ▷ Start collecting a subsequence
 9:     else if remainder $\frac{t}{f} \equiv 0$ then
10:         $s_2 = \{p_t\}$
11:         $isCollectingTS = true$
        ▷ Conduct comparison
12:     else if remainder $\frac{t}{\omega} \equiv 0$ then
13:         $s_2$ = noiseReduction($s_2$)
14:         if $s_1 = \emptyset$ then
15:             Compare $s_2$ with $\mathcal{U}$
16:         else
17:             $wd$ = dtw($s_1$,$s_2$)
18:             if wd $> \sigma$ then
19:                 highlight
20:             end if
21:         end if
22:         $s_1 = s_2$
23:         $isCollectingTS = false$
        ▷ Continue collecting subsequence
24:     else if isCollectingTS then
25:         $s_2 = s_2 \cup p_t$
26:     end if
        ▷ Increment counter
27:     $t = t + 1$
28: end loop
```

5.1 Noise Reduction

The keystroke dynamic used with respect to the work presented in this paper was flight time F^t. However, it is possible that a given flight time value is greater than normal because the subject has paused during his/her typing (an away from keyboard moment); indeed this occurs in the test data collected by the authors and used for the evaluation of the proposed approach as described in Sect. 6 below. Essentially such high values introduce unwanted "noise" into the iterative KCA process. To address this issue, a limit was placed on the F^t values in a given time series subsequence s_i using a second threshold value φ. In other words, given a specific F^t value in excess of φ, the value was reduced to φ. In the evaluation presented later in this paper a range of values for φ were considered ranging from 0.75 to 2.00s increasing in steps of 0.25s ($\{0.75, 1.00, 1.25, 1.50, 1.75, 2.00\}$). Referring back to Algorithm 1, noise reduction (adjustment) is conducted using the *noiseReduction()* function (line 13).

5.2 User Authentication on Start Up

On start up, as noted above, it is first necessary to confirm that the subject is who (s)he says (s)he is. This is done by comparing the first subsequence collected, s_1, with the relevant user profiles held in $\mathcal{U}$ (stored in a user enrolment database). This set of reference profiles is extracted from a sample typing profile K that is known to belong to the claimed subject and obtained previously. Note that K needs to be substantially greater than the maximum anticipated value for ω so that a number of subsequence reference profiles can be extracted $\{s_1, s_2, \dots\}$. The initial subsequence s_1 is compared with each subsequence held in $\mathcal{U}$, using DTW, and an average minimum wp distance $\bar{w}$ obtained. This is then compared to a σ threshold value calculated as described in the following section.

5.3 Threshold Value Calculation

Rather than using a global value for σ, a unique value is calculated for each user. This is done by comparing each profile in $\mathcal{U}$ with every other reference profile in $\mathcal{U}$ using DTW. In this manner a set of warping distance values $WD = \{wd_1, wd_2, \dots\}$ is collected. The average value of WD then becomes the selected value for σ. The σ value derived in this manner is then used with respect to the iterative KCA process. Note that averaging warping distances of a set of time series result in obtaining a representative warping value for this set; thus it can lead to efficient and accurate classification [29].

6 Evaluation

The evaluation of the proposed time series-based approach to iterative KCA is presented in this section. Experiments were conducted to: (1) evaluate the processing time required to generate a set of user profiles $\mathcal{U}$ for each enrolled user, (2) determine how well the proposed approach performed in terms of the detection of impersonators, and (3) conduct a comparison of the proposed approach with the traditional feature vector representation approach where the vectors comprise values obtained using n-graphs. The reported experiments were conducted using three datasets; these are thus presented first in Sect. 6.1. The experimental set up is then presented in Sect. 6.2. The metrics used for the evaluation were: (1) False Match Rate (FMR), (2) False Non-Match Rate (FNMR), and (3) authentication accuracy (Acc.). Note that FMR and FNMR are the standard metrics used to measure the performance of Biometric systems [13], although some researchers, in the literature, have used the terms FAR (False Acceptance Rate) and FRR (False Rejection Rate) instead. The experimental results obtained, with

Table 1 Summary of datasets

Dataset	# Subjects	Environment	Language	# Samples/subject	Collection duration	Avg. length/subject	SD
ACB	30	Free	English	3	1 day	4625	1207
GP	31	Free	Italian	15	15 days to 6 months	7157	1095
VHHS	39	Lab.	English	2	2 days to 11 months	4853	1021

respect to the above evaluation objectives, are presented in the Sects. 6.3–6.5.

6.1 Datasets

The experiments presented here were conducted using three datasets: (1) ACB, collected by the authors, (2) GP obtained from [10], and (3) VHHS obtained from [16] (the identifying acronyms are made up of the relevant author surnames). Note that each dataset comprised keystroke dynamics from free text sessions obtained from volunteer keyboard users; however, each featured differing characteristics, thus different: (1) numbers of subjects, (2) lengths of typed samples, (3) subject matter (static text and/or free text) and (4) environments in which the samples were collected (laboratory or free session). Each dataset is described in more detail below.

1. The ACB dataset was collected using undergraduates, postgraduates, and staff. A total of 30 subjects participated in providing answers (in English) to general questions so as to simulate the way that online assessments might be run. The subjects were allowed to undertake the exercise in their own time using whatever keyboard they had at hand. Each subject was requested to provide 3 samples.
2. The original GP dataset described in [10] comprised 40 subjects; however, only 31 of these were included in the publicly available dataset. Each recruited subject provides a total of 15 samples; each provided on a different day over a period of six months.
3. The VHHS dataset [16] consisted of both fixed text and free text tasks, only the free text tasks were considered with respect to the evaluation reported here. Subjects were asked to provide two free typing samples on two different days over a period of eleven months; however, the majority of subjects had provided the two samples over a period of one to two months. Each sample was produced by free typing responses to questions.

A summary of the datasets is presented in Table 1. The summary includes statistical values concerning the average length of the collected samples and the associated Standard Deviation (SD).

6.2 Experimental Setup

For the evaluation, for each subject in each dataset, the records were split into two so that one-half could be used to create individual typing templates $\mathcal{U}$ and the other to simulate a typing stream. Recall that for the proposed iterative KCA, $\mathcal{U}$ is used to: (i) derive a value for σ and (ii) for "start-up authentication". Although the data was collected in advance, to simulate a typing stream a "test harness" was constructed whereby the keystroke data was released in a manner that simulated real-time keyboard activity. Real-life experiments were not conducted as the authors wished to repeatedly use the same keyboard input data so that comparisons could be made. Wherever applicable twofold cross validation was conducted (using different halves of the keystroke time series to generate $\mathcal{U}$), hence results presented below are average results from two cross validations. For comparison with the established, feature vector based techniques, the techniques were re-implemented and re-run. The implementation language used, in all cases, was Java. All experiments were conducted using a 2.7 GHz Intel i5 processor with 16 GB RAM.

6.3 Generation of User Template and Threshold Calculation

A user template $\mathcal{U}$ is required prior to any iterative KCA. This template is also used to calculate a bespoke similarity threshold value σ. Table 2 gives the run-time values (seconds) required to generate the user template (the enrolment database), and calculate the associate σ values in each case, using a range of ω values. The "per subject" values were obtained by dividing the total run time with the number of subjects (records) in the dataset from Table 1. From the table, it can be seen that, regardless of the dataset used, processing time increased with ω. This was to be expected because the computation time required by the DTW would increase as the size of the subsequences (ω) considered increased (even though there might be less of them). There are well-known solutions in the literature to mitigate against the complexity of DTW [30–32], although no such mitigation was applied with respect to the experiments reported here (this is left to future work).

Table 2 The time taken (seconds) to generate user template $\mathcal{U}$ and associated σ values

ω	Entire dataset			Average per subject		
	ACB	GP	VHHS	ACB	GP	VHHS
25	0.629	0.918	0.895	0.021	0.030	0.023
50	1.356	1.761	1.716	0.045	0.057	0.044
75	2.316	2.76o	2.476	0.077	0.089	0.064
100	3.243	3.696	3.333	0.108	0.119	0.086
125	4.022	4.034	3.998	0.134	0.130	0.103
150	4.324	4.727	4.534	0.144	0.153	0.116

From the table, the speed at which user template is generated illustrates the efficiency of the time series based approach in comparison with established feature vector based approaches founded on *n*-graphs statistics during the enrolment and verfication stages. This is because using the proposed method there is no need to search through the time series to identify the *n*-graphs of interest or any requirement for the subsequent statistical feature calculation. In the case of the method reported on in [10] (see Sect. 2), the run-time for verifying a single typing sample was given as 140s. Using the same dataset, and the proposed approach, a single verfication takes less than 0.153s to compute (in the worst case). A considerable speed up although it is acknowledged that the experiments reported in [10] were conducted in 2005 when processing power was not what it is today.

6.4 Subject Continuous Authentication

To evaluate the effectiveness of user authentication using the proposed approach, for each dataset and each subject, the continuous typing process was simulated by presenting the keystroke dynamics for each subject in the form of a data stream. In each case, the data stream was appended to a randomly selected second data stream from another subject. The idea is to simulate one subject being impersonated by another halfway through a typing session. For every comparison of every subsequence s_i with a previously stored subsequence s_{i-1} (line 17 in Algorithm 1, and both of length ω) we recorded whether this was a True Positive (TP), False Positive (FP), False Negative (FN) or True Negative (TN). In this manner a *confusion matrix* was built up from which accuracy (Acc.), False Match Rate (FMR) and False Non-Match Rate (FNMR) could be calculated (using Eqs. 3, 4 and 5 below).

$$Acc = \frac{TP + TN}{TP + FP + FN + TN}, \tag{3}$$

$$FMR = \frac{FP}{FP + TN}, \tag{4}$$

$$FNMR = \frac{FN}{FN + TP}. \tag{5}$$

The experiments were run using ranges of ω and φ values of $\{25, 50, 75, 100, 125, 150\}$ and $\{0.75, 1.00, 1.25, 1.50, 1.75, 2.00\}$ respectively. Each experiment was also run twice, each time using a different half of the data for authentication (the other half being used for user profile generation).

The FMR and FNMR results are presented in Tables 3 and 4 with respect to each dataset; the best-recorded value, in each case, is presented in bold font. From the table, it can be seen that the best FMR and FNMR values were obtained using ω settings of 100, 125 and 150, and $\varphi = 1.25$. It can also be observed that the ω parameter had less effect on the final result than the φ parameter. This is interesting because it indicates that user authentication can be conducted using very small windows (25 keystrokes) although it is better to use some 100 keystrokes. The ω value used, as shown in Table 2, is what dictates the efficiency of the approach; the higher the ω value becomes the less efficient the comparison.

With respect to accuracy, ω and φ values of 100 and 1.25 tended to produce best results for all three data sets. The best accuracy results obtained were then 98.00, 99.00 and 96.37% for ACB, GP and VHHS datasets respectively (see Table 5).

6.5 Comparison with Feature Vector Approach

This section reports on the results obtained when the operation of the proposed iterative KCA method was compared with a feature vector based approach of the form frequently referenced in the literature. As noted in Sect. 2, there are a number of reports where a feature vectors representation, made up of keystroke dynamics related to *n*-graphs, have been applied to the KCA problem, although not in an iterative manner (authentication was conducted on typing completion). Of these, the approach described in [10] was selected for the comparison reported on here because: (i) the study obtained, to the best knowledge of the authors, the best reported FMR and FNMR (although referred to as FAR and FRR) results so far; (ii) the dataset used for the study was publicly available (although with some records missing); and (iii) the approach was well explained in the literature, therefore it was easy to reproduce. Because the study reported on in [10] was conducted in 2005 it was considered

Table 3 The obtained **FMR** values (%) using different settings of ω and φ

ω	φ					
	0.75	1.00	1.25	1.50	1.75	2.00
ACB						
25	0.668	0.658	0.598	0.058	0.737	1.011
50	0.665	0.655	0.595	0.055	0.734	0.948
75	0.661	0.651	0.591	0.051	0.730	0.944
100	0.656	0.646	**0.580**	0.590	0.725	0.939
125	0.655	0.645	0.599	0.045	0.724	0.938
150	0.648	0.638	0.586	0.038	0.717	0.931
GP						
25	0.742	0.064	0.032	0.040	0.041	0.042
50	0.743	0.065	0.033	0.041	0.042	0.043
75	0.740	0.062	0.035	0.038	0.043	0.040
100	0.741	0.063	**0.031**	0.033	0.038	0.035
125	0.741	0.063	**0.031**	0.033	0.033	0.036
150	0.745	0.067	0.035	0.034	0.034	0.036
VHHS						
25	2.267	1.228	1.048	1.078	1.071	1.092
50	2.178	1.225	1.045	1.075	1.068	1.089
75	2.177	1.227	1.047	1.077	1.070	1.091
100	2.177	1.225	1.045	1.075	1.068	1.089
125	2.175	1.228	**1.040**	1.078	1.071	1.092
150	2.175	1.227	1.047	1.077	1.070	1.091

Table 4 The obtained **FNMR** values (%) using different settings of ω and φ

ω	φ					
	0.75	1.00	1.25	1.50	1.75	2.00
ACB						
25	2.000	2.070	2.025	2.030	2.025	2.029
50	2.020	2.090	1.990	1.980	1.980	1.990
75	1.952	2.022	1.990	1.980	1.984	1.960
100	1.989	2.059	**1.970**	1.972	1.980	1.980
125	1.995	2.065	1.979	1.979	1.979	1.980
150	1.980	2.050	1.978	1.976	1.979	1.981
GP						
25	1.456	1.871	1.101	1.160	1.161	1.136
50	1.451	1.866	1.096	1.155	1.156	1.131
75	1.450	1.865	1.096	1.154	1.155	1.130
100	1.445	1.860	1.095	1.149	1.150	1.125
125	1.457	1.872	1.102	1.161	1.162	1.137
150	1.449	1.864	**1.094**	1.153	1.154	1.129
VHHS						
25	3.033	2.131	2.052	2.077	2.068	2.060
50	3.035	2.133	2.051	2.079	2.070	2.062
75	3.027	2.125	2.052	2.071	2.062	2.057
100	3.030	2.128	2.050	2.074	2.065	2.053
125	3.027	2.125	**2.049**	2.071	2.062	2.054
150	3.031	2.129	2.050	2.075	2.066	2.055

Table 5 Comparison of proposed time series-based KCA with n-graphs feature vector-based KCA taken from [10]

Dataset	Proposed time series-based KCA			Method of [10] (baseline)		
	FMR	FNMR	Acc.	FMR	FNMR	Acc.
ACB	0.58	1.97	98.02	11.10	8.60	80.82
GP	0.03	1.09	99.00	6.05	2.06	90.15
VHHS	1.02	2.04	96.37	9.09	7.13	89.15
Average	0.54	1.64	97.80	8.75	5.93	83.71

inappropriate to compare directly with the results reported, instead, the approach was re-implemented and re-run in an iterative manner, as described for the proposed system, so that meaningful comparisons could be made. The features used were the $\mathcal{F}^t$ values for all shared digraph, trigraphs and quadgraphs in each dataset. Authentication was otherwise conducted for each user as described in [10] (see also Sect. 2). The metrics used were, again, FMR, FNMR and Acc.

The results are presented in Table 5. For comparison purposes, the results for the proposed iterative KCA approach have been included in the table. From the table, it can be observed that the proposed time series-based approach to iterative KCA obtained a much better performance than the feature vector-based approach (the baseline approach), with respect to all three datasets. This, therefore, confirmed the hypothesis posed by the authors that a time series representation would serve to more effectively encapsulate keystroke dynamics than the feature vector based approach that has typically been adopted to date. It should also be noted here that the FMR and FNMR result obtained for the GP dataset, of 6.05 and 2.06% respectively, are worse than the 5.00% and 0.005% reported in the original study [10], despite using the same mechanism. It is conjectured that this is because of the change in the dataset size that had occurred since the original study and that reported here, where the original dataset consisted of 40 subjects, but the publicly available version only had 31 subjects (see Sect. 6.1). Also, in [10], an additional 165 samples that served as imposter samples were used.

7 Conclusion

In this paper, a novel mechanism for iterative KCA has been presented. The idea is to use sequences of keystroke dynamics in the form of time series and to monitor such time series, as a continuous data stream, by periodically extracting subsequences from these time series and authenticating the subsequences. In the proposed process, on start up, the first subsequence extracted for a given subject was compared to a set of reference profiles (time series) in the typing template; each subsequent subsequences s_i was then compared to its immediate predecessor subsequence s_{i-1}. In this manner iterative (repeated) subject authentication was undertaken. The initial advantage offered is that the time series approach is very efficient and thus well suited to iterative KCA. The window size (ω) used did not have to be very large, $\omega = 100$ produced best results although a window size of only $\omega = 25$ still produce good results. In terms of accuracy, a best overall accuracy of 97.8% was recorded compared to the best overall average accuracy of 83.7% when an established feature vector based technique was used (founded on the work presented in [10]). The mechanism whereby the σ threshold was calculated, unique to each individual, is also of interest, as is the noise reduction mechanism using the φ parameter. For future work, the authors intend to investigate the use of 3D time series for iterative KCA, made up of hold time and flight time.

Acknowledgements The authors would like to express their thanks to those who participated in the collection of the data and to Laureate Online Education b.v. for their support.

References

1. Bours P (2012) Continuous keystroke dynamics: A different perspective towards biometric evaluation. Inf Secur Tech Rep 17(1):36–43
2. Gaines RS, Lisowski W, Press SJ, Norman S (1980) Authentication by keystroke timing: some preliminary results. Technical report, DTIC Document. RAND Corp Santa Monica CA
3. Shepherd SJ (1995) Continuous authentication by analysis of keyboard typing characteristics. In: Security and Detection, European Convention on. IET, pp 111–114
4. Bleha S, Slivinsky C, Hussien B (1990) Computer-access security systems using keystroke dynamics. IEEE Trans Pattern Anal Mach Intell 12(12):1217–1222
5. Rick Joyce, Gopal Gupta (1990) Identity authentication based on keystroke latencies. Commun ACM 33(2):168–176
6. Ogihara A, Matsumuar H, Shiozaki A (2006) Biometric verification using keystroke motion and key press timing for atm user authentication. In: Intelligent Signal Processing and Communications. ISPACS'06. International Symposium on. IEEE, pp 223–226

7. Syed Z, Banerjee S, Cukic B (2014) Normalizing variations in feature vector structure in keystroke dynamics authentication systems. Softw Qual J:1–21
8. Monrose F, Rubin A (1997) Authentication via keystroke dynamics. In: Proceedings of the 4th ACM conference on Computer and communications security. ACM, New York, pp 48–56
9. Dowland PS, Furnell SM (2004) A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In: Security and Protection in Information Processing Systems. Springer, New York, pp 275–289
10. Gunetti D, Picardi C (2005) Keystroke analysis of free text. ACM Trans Inf Syst Secur (TISSEC) 8(3):312–347
11. Ahmed AA, Traore I (2014) Biometric recognition based on free-text keystroke dynamics. Cybern IEEE Trans 44(4):458–472
12. Raja M, Sigg S (2016) Applicability of rf-based methods for emotion recognition: a survey. In: 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops). IEEE, pp 1–6
13. Unar JA, Seng WC, Abbasi A (2014) A review of biometric technology along with trends and prospects. Pattern Recognit 47(8):2673–2688
14. Bergadano F, Gunetti D, Picardi C (2002) User authentication through keystroke dynamics. ACM Trans Inf Syst Secur (TISSEC) 5(4):367–397
15. Dowland PS, Furnell SM, Papadaki M (2002) Keystroke analysis as a method of advanced user authentication and response. In: Security in the Information Society. Springer, New York, pp 215–226
16. Vural E, Huang J, Hou D, Schuckers S (2014) Shared research dataset to support development of keystroke authentication. In: Biometrics (IJCB), 2014 IEEE International Joint Conference on. IEEE, pp 1–8
17. Alshehri A, Coenen F, Bollegala D (2016a) Towards keystroke continuous authentication using time series analytics. In: Proc. AI 2016, Research and Development in Intelligent Systems XXXIII. Springer, New York, pp 325–338
18. Alshehri A, Coenen F, Bollegala D (2016b) Keyboard usage authentication using time series analysis. In: International Conference on Big Data Analytics and Knowledge Discovery. Springer, New York, pp 239–252
19. Monrosea F, Rubinb AD (2000) Keystroke dynamics as a biometric for authentication. Future Gener Comput Syst 16(4):351–359
20. Richardson A, Kaminka GA, Kraus S (2014) Reef: Resolving length bias in frequent sequence mining using sampling. Int J Adv Intell Syst 7(1):2
21. Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining. ACM, New York, pp 359–370
22. Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. Proc VLDB Endowment 1(2):1542–1552
23. Rabiner L, Juang B-H (1993) Fundamentals of speech recognition. Prentice Hall, Upper Saddle River
24. Bagnall AJ, Janacek GJ (2004) Clustering time series from arma models with clipped data. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, pp 49–58
25. Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. Proc VLDB Endow 1(2):1542–1552
26. Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: KDD workshop. Seattle, pp 359–370
27. Wang X, Mueen A, Ding H, Trajcevski G, Scheuermann P, Keogh E (2013) Experimental comparison of representation methods and distance measures for time series data. Data Min Knowl Discov 26(2):275–309
28. Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. Sov Phys Dokl 10(8):707–710
29. Niennattrakul V, Ratanamahatana CA (2009) Shape averaging under time warping. In: Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology. ECTI-CON 2009. 6th International Conference on, vol 2. IEEE, pp 626–629
30. Itakura F (1975) Minimum prediction residual principle applied to speech recognition. IEEE Trans Acoust Speech Signal Process 23:52–72
31. Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization fro spoken word recognition. IEEE Trans Acoust Speech Signal Process 26:43–49
32. Silva DF, Batista GE (2016) Speeding up all-pairwise dynamic time warping matrix calculation. In: Proceedings of the 2016 SIAM International Conference on Data Mining. SIAM, pp 837–845