Pattern Recognition Letters 148 (2021) 61–67

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Identity authentication based on keystroke dynamics for mobile device users

Zhigang Gao[a], Wenjie Diao[a], Yucai Huang[a], Ruichao Xu[a], Huijuan Lu[b], Jianhui Zhang[a,*]

[a] *College of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China*
[b] *Key Laboratory of Electromagnetic Wave Information Technology and Metrology of Zhejiang Province, College of Information Engineering, China Jiliang University, Hangzhou 310018, China*

ARTICLE INFO

Article history:
Received 21 October 2020
Revised 30 March 2021
Accepted 22 April 2021
Available online 6 May 2021

Keywords:
Identity authentication
Keystroke dynamics
Data cleaning
Centroid fluctuation ranges

ABSTRACT

User authentication of mobile devices is an important means to protect data security and user privacy. Low overhead, high accuracy and continuous authentication are challenging problems in user authentication. Aiming at the disadvantages of current authentication methods based on keystroke dynamics such as low accuracy and one-time authentication, this paper proposes a new authentication method named UIKI (User Identity authentication method based on clusters of Keystroke time Intervals). UIKI consists of the valid user modeling phase and the runtime user authentication phase. In the valid user modeling phase, UIKI uses a clustering algorithm to find the stable centroids and the fluctuation range of a valid user's centroid positions. In the runtime user authentication phase, UIKI clusters the data to be authenticated with the stable centroids as the initial centroids, and compares the results of clustering with the stable centroids and the centroid fluctuation range of the valid user to determine whether the input data are from the valid user. Experimental results prove that UIKI has the average FAR (False Accept Rate) of 0.082 and the average FRR (False Reject Rate) of 0.052, which can effectively authenticate a user's identity. UIKI has the advantages of high accuracy, low overhead and continuous authentication, which is suitable for the user authentication of mobile devices.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

With the development of society, mobile devices such as mobile phones, Pads, etc., have played increasingly important roles in our lives, and they can provide services for users anytime and anywhere. Because people are increasingly used to dealing with their daily affairs by mobile applications, more and more privacy data are stored in mobile devices, for example, the passwords of bank accounts, transaction information, messages, and private photos. With the increasing disclosure events of privacy data, the protection of privacy data is becoming the attention focus of people. How to store users' privacy data safely has become an important issue which needs to be solved during the usage of mobile devices.

User authentication is a common means to protect users' privacy data or improve the security [1]. Some common mechanisms such as digital passwords and sliding passwords have been integrated in Android systems or iOS systems. However, most people are used to setting a simple password with lower strength for their convenience [2]. Therefore, unauthorized users may guess the password and it causes information disclosure. In recent years, some biological authentication means have emerged, such as fingerprint identification [3], face recognition [4], and pupil recognition [5], or the combination of multiple recognition techniques [6–8], etc. Although these means solve the problem that simple passwords incur, they often spend high costs in collecting the biological features by special sensors. Moreover, they are easy to reveal the privacy of users, and difficult to carry out safety protection continuously.

With regard to the above problems, users can collect the data of touch screens [9–11], and then use machine learning techniques to perform authentication. This means can not only protect the user's privacy, but also remove the requirement for additional hardware. What is more, the authentication process is transparent to users. However, when we use this means to carry out identification, it has some limitation in application scenarios. For example, when a user is walking or sitting, the values of data change greatly because the biological features are diverse in different scenarios, which leads to low recognition accuracy.

When using mobile devices, users usually frequently use virtual keyboards in fixed gestures and scenarios. Some scholars have researched the method of keystroke dynamics based on PC key-

* Corresponding author.
E-mail address: jh_zhang@hdu.edu.cn (J. Zhang).

https://doi.org/10.1016/j.patrec.2021.04.019
0167-8655/© 2021 Elsevier B.V. All rights reserved.

boards [12–14]. Because of the difference in usage habits of keyboards, such as the typing speeds, typing gestures, etc., users will generate different keystroke feature data. Therefore, we can collect keystroke data to recognize users' identities. In this paper, we find that users' gestures are relatively stable and there is a low correlation between keystroke dynamic features and input scenarios when users use input data from keyboards. Because the biological features are stable and easy to extract during keyboard inputs, we can collect the data of keystroke dynamics to conduct identity authentication. However, most of the existing research based on free text focuses on *n*-graph, it is difficult to improve the accuracy and it often takes much time to build user models. Aiming at the above problems, this paper proposes a method named UIKI, it is a new method of authenticating the identity of a user, and has the advantage of high recognition rate and continuous authentication. The contributions of this paper are twofold. First, we propose a valid user modeling method which combines stable centroids and centroid fluctuation ranges. Second, we present a fixed centroid clustering and user authentication method. Therefore, UIKI can achieve continuous user authentication with high accuracy.

The rest of this paper is organized as follows. Section 2 introduces the related works in the field of user authentication based on keystroke dynamics. Section 3 describes the implementation process of UIKI in detail. Section 4 gives the experiment comparison and analyses. Finally, we summarize this paper in Section 5.

2. Related works

Keystroke dynamic authentication is a means to verify the identities of users, which collects the keystroke characteristics of users and then derives their keystroke behavior patterns. The existing identity authentication methods based on keystroke dynamics can be classified into two categories, i.e., one is based on fixed text and the other is based on free text.

The authentication based on fixed text is characterized by fixed text input (such as passwords). It is often embedded in the login system as an auxiliary authentication scheme, and has the characteristics of simple modeling process, and fast authentication speed, etc. De Luca et al. [15] used the pressure and input speeds when entering passwords as auxiliary features for identity authentication. Giuffrida et al. [10] improved the effect of keystroke dynamic authentication by collecting the data from the acceleration sensor when users input the fixed text. Janakiraman and Sim [16] extracted a valid user's characteristics to recognize the user's identity when the user inputs fixed words.

The authentication based on free text is characterized by no restrictions on the input text. Compared to the authentication based on fixed text, users can input any text, and have more flexibility. It is suitable to the scenario where text input is necessary, such as e-mails, short messages or chatting. Al Solami et al. [17] proposed to use the directed graph to measure the distance between input samples and the ones stored in the database. Tsai and Shih [18] proposed a new keystroke clusters map (KC-Map) based on the keystroke time features, and it could effectively improve the accuracy of identity authentication when using free text. Messerman et al. [14] proposed an improved scheme to adaptively detect the changes of users. Zhou et al. [19] presented a method for describing and calculating distinct user features according to their dynamic connections and influential behaviors. This method can distinguish four kinds of specific users in order to support the information transmission in the social network.

In this paper, we extend the method presented by Tsai and Shih [18], propose UIKI, and implement the UIKI algorithm in a KDA (Keystroke Dynamic Authentication) system named UIKI KDA, which is a letter-related keystroke dynamic authentication system based on free text in order to meet the requirement of high-precision continuous user authentication. UIKI KDA includes the functions such as collecting the data of valid users, modeling valid users, and authenticating runtime users, all of which are implemented on mobile devices.

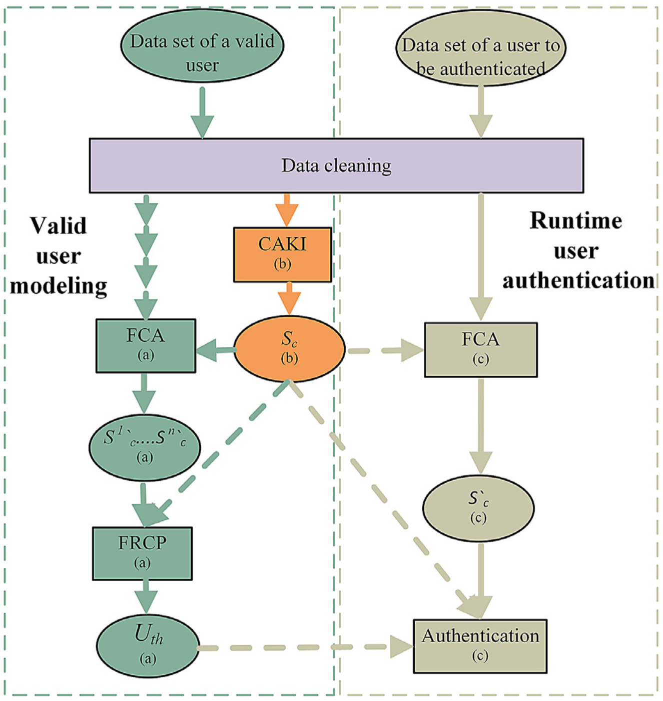

Fig. 1. Workflow of UIKI. The left half part is the valid user modeling phase, and the right half part is the runtime user authentication phase. Moreover, it indicates the different workflows with different colors and numbers.

3. Implementation of UIKI

The workflow of UIKI is shown in Fig. 1, which includes the valid user modeling phase (which is shown on the left half part of Fig. 1) and the runtime user authentication phase (which is shown on the right half part of Fig. 1).

The goal of the valid user modeling phase is to extract the features which can be used for user authentication from the data set of the valid user. The goal of the runtime user authentication phase is to use the continuous input data when a user uses a mobile phone to judge whether the current user is a valid user or not. As shown in Fig. 1, data cleaning is a necessary process for both the valid user modeling phase and the runtime user authentication phase. In the valid user modeling phase, the input data set is the keystroke data of the valid user (i.e., the training data). After performing data cleaning, the training data are divided into the left and right outputs. The outputs on the left are data segments with 300 data (it is set to 300 according to the usual word length limit in chatting.) in each group that will be processed along with the green modules with the marks of (*a*). First, the Fixed centroid Clustering Algorithm (FCA) calculates the centroid set $S_c^{i`}$ of *ith* data segment to obtain *n* centroid sets $S_c^{1`}$, $S_c^{2`}$, ..., $S_c^{n`}$. After that, UIKI uses the authentication based on Fluctuation Ranges of Centroid Positions (FRCP) to calculate the fluctuation range U_{th} of the centroids for the *n* centroids. After performing data cleaning, the output on the right is all the cleaned training data and they will be processed along with the orange modules with marks of (*b*). UIKI uses the Clustering Algorithm based on Keystroke time Intervals (CAKI) presented by this paper to calculate a stable centroid set S_c with optimized centroid offset distance.

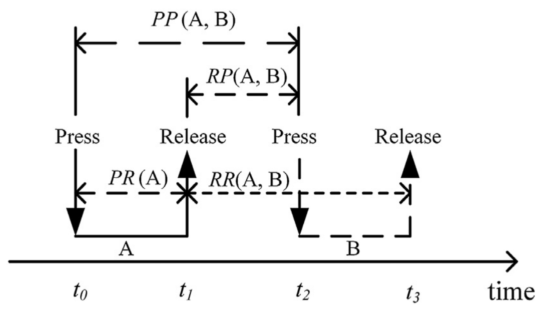

Fig. 2. Time of user keystrokes. It illustrates the time intervals of four event combinations, i.e., *PP, PR, RP*, and *RR*.

The workflow of the runtime user authentication phase is shown on the right half part of Fig. 1, which is processed by the light brown modules with marks of (*c*). When the UIKI KDA system is running, the user's keystroke data (i.e., the test data) are collected and cleaned, and the identity of the user is authenticated every 300 data. During authenticating, the UIKI KDA system first clusters the data with S_c as the initial centroids, and then the authenticator compares U_{th} with S_c based on the results of FCA clustering to determine whether the keystroke data are the data of a valid user and returns the authentication results. The process of user identity authentication works all the time, so a user's identity can be continuously authenticated.

3.1. Time characteristics of keystrokes

It will produce various biological characteristics during users typing. The characteristics of keystroke time have the advantages of not requiring specific hardware support and easy to obtain. Fig. 2 shows the events and their occurrence time when a user continuously types two characters "*A*" and "*B*". We can see that the user's keypress action consists of two steps, i.e., key press and key release.

Considering the time orders of key press and key release, we can get four event combinations, i.e., *PP, PR, RP*, and *RR*. As shown in Fig. 2, the event combination of *A* press and *B* press is defined as *PP(A, B)*, the event combination of *A* press and *A* release is defined as *PR(A)*, the event combination of *A* release and *B* press is defined as *RP(A, B)*, and the combination of *A* release and *B* release event is defined as *RR(A, B)*. The time interval between the two events is the time difference between the occurrences of the two events.

The existing studies proved [20] that the period *PP* contained more user biological features, and can be used to achieve better identification effects. Therefore, the keystroke time of *PP* is selected as the input data used in this paper. In this paper, clustering is based on letters. In other words, it calculates the average value of *PP* for each letter, and then clusters the letters with the same time interval into the same cluster. In the following part of this paper, we do not distinguish between uppercase and lowercase letters of keystrokes. For example, "*A*" and "*a*" are considered as the same letter. In this paper, T_i denotes the time interval of the *ith PP* event, which is defined by

$$T_i = t_{2i+2} - t_{2i} \tag{1}$$

Where t_{2i} is the time instant when the *ith* key press occurs. For example, in Fig. 2, the letter pressed by the zeroth keystroke is "A" at t_0, and the letter pressed by the first keystroke is "B" at t_2. Therefore, $T_0 = t_2 - t_0$. The set of all T_i is called the keystroke interval set *T*. In this paper, a data refers to the input data corresponding to a *PP* event, which is the value of a T_i.

```
Algorithm 1:. Data Cleaning.
Input: TA, and the length n of TA
1. Copy the data from TA to TB;
2. Sort the data in TB in ascending order;
3. $q1 = \frac{n+1}{4}$ ; // get the first quartile of TB
4. $q2 = \frac{3(n+1)}{4}$ ; // get the third quartile of TB
5. Copy the data between TB[q1] and TB[q2] to TA;
6. Output: TA
```

3.2. Data cleaning

Mobile devices such as smartphones and Pads do not have stringent interaction time limits. Therefore, some of the collected raw data will have large deviations due to interference from various factors. In the runtime user authentication phase, abnormal data will lead to a large distortion of behavioral features, which might result in incorrect authentication results. We found that only when a user continuously inputs data, T_i will include the user's behavior features. That is to say, only part of the data collected by the UIKI KDA system can reflect the user's behavioral features. Otherwise, if the user is interrupted during typing, for example, when someone comes to speak to him, there will be a too large T_i. Similarly, if the user makes a mistake during typing, T_i will be too small to be used for authentication. Gentner [21] found that the T_i value of expert users was about 96 ms, while that of novices was about 825 ms, and the normal value of T_i should be distributed in a small range decided by the characteristic of a user. Therefore, it is necessary to clean the raw data to eliminate the interference of abnormal data. Additionally, we find that the distribution of the data in set *T* is concentrated in the middle, and the middle part of data contains more user features. The middle refers to the data between the first quartile and the third one (i.e., the middle 50% data after the data is sorted in ascending order).

We present a data cleaning method by calculating the quartiles of *T*. The pseudo code of the data cleaning algorithm is shown in Algorithm 1. We assume that the data in *T* are stored in an array *TA* in ascending order, and *TB* is an auxiliary array. Data cleaning has no negative impact on the results of runtime user authentication except it extends the time interval for user authentication because of longer data collecting time. On the contrary, it eliminates the input abnormal data, and makes the runtime user authentication more accurate.

In Algorithm 1, Line 1 copies the data from *TA* to *TB*. Line 2 sorts the data in *TB* in ascending order of T_i. Line 3 and line 4 assign the first and the third quartiles to variables *q1* and *q2*, respectively. Line 5 copies the data in *TB* which are less than *TB[q2]* and more than *TB[q1]* to *TA*. Finally, line 6 outputs the results. After *TA* is processed by Algorithm 1, we obtain the cleaned data, which can be used in the subsequent valid user modeling phase and the runtime user authentication phase.

3.3. Clustering algorithm based on keystroke time intervals

In CAKI, it uses the *K*-means algorithm to perform data clustering in order to classify all letters into *k* clusters ($1 \le k \le 26$, $k \in N$), and then obtains the stable centroid set S_c. Only the alphabet letters are selected as keystroke data in order to obtain higher differentiation.

Before calling CAKI, we first calculate the average key press time of each letter by

$$\overline{T}(l_i) = \frac{1}{N_i} \sum_{k=1}^{N_i} T(l_i^k) \tag{2}$$

Algorithm 2:. CAKI algorithm.

```
Input: $TM$, the cluster number $k$ and the threshold $n$
Output: User cluster centroid vector $S_c = \{c_1, c_2, ... c_k\}$
1. Initialize: Set $sd = Max$, $flag = 0$; // set the initial variables
2. do
3.     Randomly choose $k$ non-null elements from $TM$ as the initial centroids $c_1, c_2, ... c_k$ of $C_1, C_2, ... C_k$;
4.     repeat
5.         $c_1^`, c_2^`, ..., c_k^` = c_1, c_2, ..., c_k$;
6.         for each $\overline{T}(l_i) \in TM$ and $\overline{T}(l_i) \neq$ null do
7.             Calculate the distance between $\overline{T}(l_i)$ and $c_1, c_2, ... c_k$ respectively;
8.             Assign $\overline{T}(l_i)$ to the cluster $C_i$ with the minimum distance between $\overline{T}(l_i)$ and $c_i$;
9.         end for
10.        Recalculate the new cluster centers $c_1, c_2, ... c_k$;
11.    until $\sum_{i=1}^{k} |c_i^` - c_i| <= \delta$;
12.    $D = 0$;
13.    for each cluster $C_i$ in $C_1, C_2, ... C_k$ do
14.        $D_i = 0$;
15.        for each element $\overline{T}(l_j)$ in $C_i$ do
16.            $D_i = D_i + |\overline{T}(l_j) - c_i|$
17.        end for
18.        $D = D + D_i$;
19.    end for
20.    if $sd > D$ then
21.        $sd = D$;
22.        $S_c = \{c_1, c_2, ... c_k\}$;
23.        $flag = 0$;
24.    else
25.        $flag$++;
26.    end if
27. while $flag <= n$
28. return $S_c$;
```

Where l_i denotes a letter, $\overline{T}(l_i)$ represents the average key press time of l_i, and N_i ($N_i > 0$, $N_i \in N$) is the number of times that the letter l_i appears in set T, $T(l_i^k)$ denotes the key press time of l_i in the kth keystroke.

$\overline{T}(l_i^k)$ will be set to *null* if the letter l_i does not appear in the pressing data. The results are stored in the array TM, $TM = [\overline{T}(l_1), \overline{T}(l_2), ..., \overline{T}(l_{26})]$, where $\overline{T}(l_i) \in R$.

After that, we can classify TM into k clusters and obtain the user's stable centroid set S_c by the CAKI algorithm.

The implementation process of the CAKI algorithm is shown in Algorithm 2. In Algorithm 2, Line 1 initializes the steady dispersion *sd* to a maximum value and sets a variable *flag* to count the times that *sd* keeps unchanged during iteration. *sd* is used for saving the minimum dispersion value. Lines 2 to 27 cluster the data of average key press time. Line 3 randomly selects k non-empty elements from TM as the centroids c_1, c_2,..., c_k of the initial clusters C_1, C_2,..., C_k. Lines 4 to 11 run a loop to classify the data in TM into k clusters by calculating the absolute value of the difference between the centroids and each element (the distance between $\overline{T}(l_i)$ and c_i is abs($\overline{T}(l_i)$-c_i)), then we can get the new centroid c_i by

$$c_i = \frac{1}{m} \sum_{j=1}^{m} \overline{T}(n_j) \tag{3}$$

Where m is the number of elements in the cluster C_i, and n_j is the jth letter in the cluster C_i.

The cluster process will finish when the sum of the difference between the old centroids and the new ones is smaller than the threshold δ. Line 12 initializes D to zero in order to calculate the sum of the dispersion values of all clusters, and Line 14 initializes D_i to zero in order to calculate the dispersion of each cluster. Lines 13 to 19 show the calculation process of D_i and D. Lines 20 to 26 determine whether this iteration obtains a smaller dispersion than ever before. If the positions of all centroids do not change during n consecutive iterations, the CAKI algorithm ends and outputs the stable cluster centroid set S_c with the optimized dispersion.

Algorithm 3:. FCA algorithm..

```
Input: $TM$, the cluster number $k$ and $S_c$,
Output: $S^`_c$
1. Set $S_c$ to the initial cluster controids of $C_1, C_2, .. C_k$;
2. for each $\overline{T}(l_i) \in TM$ do
3.     Calculate the distance between $\overline{T}(l_i)$ and $c_1, c_2, ... c_k$ respectively;
4.     Assign $\overline{T}(l_i)$ to the cluster $C_i$ with the minimum distance between $\overline{T}(l_i)$ and $c_i$;
5. end for
6. for each cluster $C_i$ in $C_1, C_2, \cdots, C_k$ do
7.     if $C_i$ is null then
8.         $c_i = 0$;
9.     else
10.        $c_i$ = the mean value of all elements in $C_i$;
11.    end if
12. end for
13. $S^`_c = \{c_1, c_2, ... c_k\}$;
14. return $S^`_c$;
```

3.4. Fixed centroid clustering algorithm

Generally speaking, the K-means algorithm will generate different clustering results for the same dataset when we choose different initial centroids due to its instability.

Under normal circumstances, a user presses the same letter at a stable interval, resulting in consistent results for keystroke data clustering. Because UIKI outputs S_c with the minimal dispersion during the valid user modeling phase, S_c is the clustering centroid of the user data in a stable state. In the runtime user authentication phase, when using the elements of S_c as the centroids to perform a one-time clustering for valid user data, the clustering results are steadily distributed across each cluster, and the newly generated centroids are not much different from S_c. When using the elements of S_c as the centroids to perform a one-time clustering for invalid user data, clustering results become extreme, often with large numbers of clusters empty. Therefore, when using one-time clustering, there is a far distance between the valid user cluster centroids and the invalid user cluster centroids, which is suitable for user authentication.

In this paper, we propose a fixed centroid clustering algorithm named FCA. By using the stable cluster centroids S_c as the initial centroids, we can obtain the centroid set $S‘_c$ by clustering the data to be authenticated.

The steps of FCA are shown in Algorithm 3. In Line 1, it sets S_c to the initial centroids of clusters. From Lines 2 to 5, it calculates the absolute value of the distance between the centroids and each element and classifies the data in TM into k clusters. From Lines 6 to 12, it calculates the average value of the elements in each cluster. If the cluster is empty, the centroid will be assigned to zero. At the end of FCA, it returns the cluster centroid set $S‘_c$.

3.5. Authentication based on fluctuation ranges of centroid positions

When authenticating the validity of input data in the runtime user authentication phase, we need to use the authentication features obtained in the valid user modeling phase. The authentication features used in this paper include two kinds, one is the stable centroid set of valid users S_c, and the other is the centroid fluctuation range U_{th}. In the previous section, we have proposed the CAKI algorithm to calculate S_c. In this section, we will present the FRCP algorithm to calculate U_{th}.

Assuming the length of the training data set of the valid user TM is n after they are cleaned. We divide TM into p segments with the length of m ($p = n/m$). For each segment, we first use the Eq. (2) to calculate the average key press time $\overline{T}(l_i)$ for each letter. Then, we run the FCA algorithm to calculate the centroid set S_c‘. Finally, we calculate the centroid offset distance *Dis* between S_c‘ and

S_c by the Eq. (4). Note that before using the Eq. (4), we first sort the data in S_c and $S‘_c$ in ascending order, and then subtract c and $c‘$ with the same number.

$$Dis = \sum_{i=1}^{k} \left|c_i - c_i^{‘}\right|,\ c \in S_c,\ c^{‘} \in S_c^{‘} \tag{4}$$

Assuming the offset distances Dis_1, Dis_2, …, Dis_p of the p-segment data are stored in an array *F-Dis*. U_{th} represents the fluctuation range of the valid user's centroids in *F-Dis* when the confidence interval is set to p ($0 < p < 1$). This paper proposes the following method to calculate U_{th}. First, we sort the values in *F-Dis* in ascending order, and then we calculate its lower quartile and upper quartile according to the Eqs. (5) and (6), respectively.

$$p_L = \frac{1-p}{2} \tag{5}$$

$$p_H = p + p_L \tag{6}$$

After obtaining the quartiles, we calculate the lower bound Dis_L and the upper bound Dis_H of U_{th} by the Eqs. (7) and (8), respectively.

$$Dis_L = F{-}Dis[p_L \times (n+1)] \tag{7}$$

$$Dis_H = F{-}Dis[p_H \times (n+1)] \tag{8}$$

Where n denotes the length of *F-Dis*. The fluctuation range of the valid user's centroids is $U_{th} = [Dis_L, Dis_H]$.

3.6. Authenticator

In this paper, we use U_{th} and S_c as the authentication features. In the runtime user authentication phase, after each segment of an unauthenticated user keystroke data are collected and cleaned, FCA will be used for calculating $S‘_c$, and then *Dis* between $S‘_c$ and S_c will be calculated by using the Eq. (4). After obtaining $S‘_c$ and *Dis*, the authenticator will count the number of zero in $S‘_c$. If the number of zero accounts for half of the total number of centroids in $S‘_c$, the user will be authenticated as an invalid one. Otherwise, the authenticator will use the Eq. (9) to do further authentication. In the Eq. (9), if $Dis_L \le Dis \le Dis_H$, the user is a valid one, otherwise the user is an impostor.

$$user = \begin{cases} \text{Valid User if } Dis_L \le Dis \le Dis_H \\ \text{Impostor otherwise} \end{cases} \tag{9}$$

4. Experiments and analyses

4.1. Experiment setups

We implemented the UIKI algorithm by Java programming language in the UIKI KDA system. The mobile devices we used for the following experiments are HUAWEI Honor 9 and HUAWEI Honor 6 mobile phones. We recruited 20 volunteers, 15 of whom were familiar with smartphones with user IDs from 1 to 15 (including 6 women and 9 men with the age mean of 23.8 years old and the age standard deviation of 4.2), and 5 of whom were unfamiliar with smartphones with user IDs from 16 to 20 (including 2 women and 3 men with the age mean of 28.8 and the age standard deviation of 1.3). They are between 20 and 30 years old.

In the following experiments, two kinds of data are used in the valid user modeling phase and the runtime user authentication phase. The first one is the user comments on Twitter (Twitter is a popular social platform where people express their thoughts within the spoken character limit.), called *DS1*, and the second one is the public dataset provided by Belman and Phoha [22], called *DS2*. Based on the difference in data volume and user number, this paper uses DS1 together with DS2 to verify the effectiveness of UIKI. The data used in the following experiments is summarized in Table 1. *DS1* is collected from one user, which means every user has a *DS1. DS2* is the data of 40 users which are randomly selected from [22]. In the runtime user authentication phase, the data in *DS1* and *DS2* are randomly mixed according to data segments with the length of 300.

Table 1
Experiment data.

Valid User Modeling	Runtime User Authentication	
DS1	**DS2**	**DS1**
12K	**180K**	**20K**

We perform authentication once every 300 data according to the limit of user comment length on the Twitter website. With regard to the cluster centroid number k, the experiment results show that when $k = 4$, UIKI achieves the best authentication effect and the highest recognition ratio. Additionally, we set the confidence interval p of valid users to 95%, which means that 5% of valid users might be misclassified. Note that p can be tuned according to different users. It is recommended that a slightly larger p is set for users who are familiar with smartphones because their centroid floating ranges are relatively stable, and vice versa. We use two error ratios, FAR and FRR, to evaluate the authentication accuracy of the UIKI KDA system.

4.2. T-means classifier

In order to compare with UIKI, we construct a statistical classifier called T-means by using the mean and variance of T_i. Different from the CAKI algorithm, the T-means authentication method does not need to calculate the mean of each letter, and it only calculates the mean μ_0 and variance σ_0 of each segment data. They are defined as

$$\mu_0 = \frac{1}{m}\sum_{i=1}^{m} T_i \tag{10}$$

$$\sigma_0 = \frac{1}{m}\sum_{i=1}^{m} |T_i - \mu_0| \tag{11}$$

Where m represents the length of data.

Similar to UIKI, for a given valid user training data with the length of n, T-means uses the Eqs. (12) and (13) to calculate the mean and variance of the key press data, respectively, and derives the user authentication features.

$$\mu_H = \mu_0 + \sigma_0 \tag{12}$$

$$\mu_L = \mu_0 - \sigma_0 \tag{13}$$

The centroid fluctuation range of valid users is denoted as $U_{th} = [\mu_L, \mu_H]$. After obtaining the valid user authentication features, T-means uses the Eq. (9) to classify the users to be authenticated based on the mean value μ of the test data.

4.3. Experiment results and analyses

4.3.1. Influence of data cleaning

Fig. 3 shows the FAR and FRR without data cleaning. As shown in Fig. 3, the minimum and maximum values of FAR in Fig. 3 are 0.25 and 0.81, respectively. As shown in Fig. 3, in the absence of data cleaning, the authenticator cannot effectively carry out user authentication for the skilled users and unskilled ones. The reason is as follows. There usually are disturbances and mistakes when

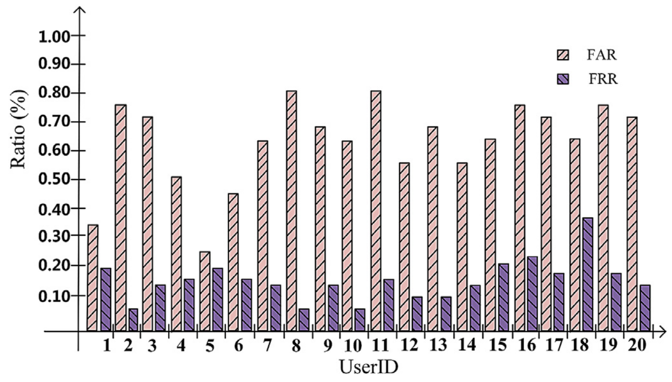

Fig. 3. User authentication results without data cleaning. It shown both FAR and FRR are high.

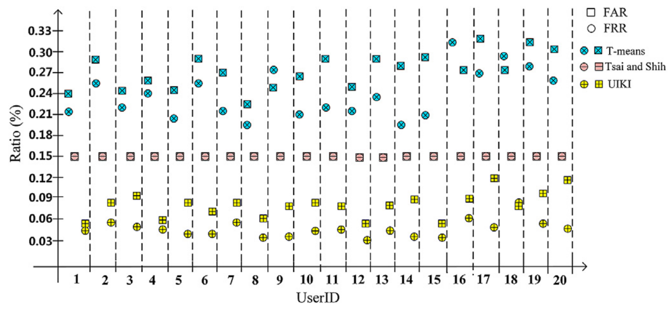

Fig. 4. Authentication accuracy comparisons. It shows UIKI has the least values in both FAR and FRR.

users input letters. Therefore, there will be many excessively large values of *PP* (e.g., some *PP* values are greater than 1000 ms), resulting in an excessively large mean value. At the same time, due to the user's keystrokes by mistakes, there will be many *PP* values that are too small (e.g., some are smaller than 50 ms). Too large or small data will cause adverse effects on the clustering results of the K-means algorithm, and eventually lead to an excessive U_{th} range. In the runtime user authentication phase, the authenticator will misclassify invalid users into valid ones when their data fall within the U_{th} range, or misclassify valid users into invalid ones when valid users have some extreme values that fall beyond the U_{th} range. In order to achieve correct authentication results, data cleaning is a necessary step in UIKI.

4.3.2. Comparison of authentication accuracy

We make a comparison between three methods, i.e., the T-means classifier, the authentication system proposed by Tsai and Shih [18] and UIKI. Fig. 4 shows the authentication results of 20 users. The horizontal axis denotes the user number, and the vertical axis shows the values of FARs and FRRs. From the distribution of FARs and FRRs, it can be concluded that the *T*-means classifier has the worst authentication accuracy, the method proposed by Tsai and Shih [18] performs better than that of *T*-means, and UIKI has the best accuracy. As shown in Fig. 4, FRRs are stable overall, floating around 0.05. Since we set 95% of the data to fall within the valid user range, which means 5% of the valid data will be mis-classified by default. Therefore, the FRRs of UIKI fluctuate around 0.05. In general, the FARs have smaller floating ranges. The FARs of user IDs between 1 and 15 is relatively stable with the average value of 0.076; The FARs of user IDs between 16 and 20 have larger floating ranges with the average value of 0.103. It shows that UIKI has a higher differentiation degree for the skilled users than that of the unskilled ones because the *PP* value distribution is more concentrated for the skilled users, resulting in the smaller FARs. In addition, although the FARs of UIKI fluctuate in a small range, it is still smaller than that of the other two methods.

T-means only considers the mean and variance of T_i, and ignores the relevance of specific letters. Although the method proposed by Tsai and Shih [18] takes the relevance of specific letters into consideration, they did not consider the impact of the different initial centroids on the clustering results. Additionally, they did not evaluate the dispersion of the clustering results. UIKI has higher authentication accuracy because of the following two reasons. First, we consider the relevance between the specific letters and the key press data. Second, we propose two methods to reduce the impact of the initial centroids. One is that we cluster iteratively in CAKI until we find a stable centroid with optimized dispersion. The other is that we use the stable centroid as the initial centroid in the runtime user authentication phase. In our experiments, the average value of FARs is 0.082, and the average value of FRRs is 0.052. The experiment results show that UIKI can effectively improve the authentication accuracy.

Moreover, we test the resource overhead such as the CPU utilization, memory footprints and running time consumption. The main functions of the UIKI KDA system include the user data collecting, the valid user modeling, and the runtime user authentication. The user data collecting accounts for most of the running time of the UIKI KDA system. The CPU utilization, the memory footprint and the CPU time consumption account for about 1%, 10 MB, and 20 ms, respectively. In the valid user modeling phase, the CPU utilization, the memory footprint and the CPU time consumption account for about 27%, 74 MB, and 13900 ms, respectively. In the runtime user authentication phase, the CPU utilization, the memory footprint and the CPU time consumption account for about 21%, 16 MB, and 1200 ms, respectively. Therefore, the UIKI KDA system consumes more resources only during the valid user modeling. However, it only carries out the valid user modeling once. Moreover, high CPU loads (e.g, when a user is watching videos) and keystroke input usually do not occur simultaneously, so the real-time performance of authentication will not be affected when multiple APPS are running at the same time. Therefore, the UIKI KDA system will not affect the responsiveness of mobile devices. It shows that UIKI has the advantages of low resource overhead and fast authentication speed, which is suitable for continuous authentication for mobile device users.

Finally, we analyzed the classification results of CAKI algorithm, the FARs and FRRs, and found there is no noticeable difference between the younger and older participants. It may be because the age of the participants is between 20 and 30 years old, and they have good eyesight and fast keystroke response speeds.

5. Conclusions

User authentication is widely used for protecting users' privacy data in mobile devices. The authentication method of keystroke dynamics has low costs and is stable on its features. However, the existing research is commonly based on word-independent methods, which has low recognition accuracy, and is unsuitable for free text authentication.

Aiming at the above problems, this paper proposes UIKI, which consists of the valid user modeling phase and the runtime user authentication phase. The data used in these two phases need to be cleaned to remove abnormal data. In the valid user modeling phase, UIKI uses the CAKI algorithm to iteratively cluster the training data to obtain optimized stable centroids, and then inputs the training data with fixed-length segments. After that, the system uses the FCA algorithm to obtain the centroids of the segmented data, and then uses the FRCP algorithm to calculate the offset distance between the stable centroids and each centroid in the centroid set. According to the specified confidence interval, the fluctuation range of valid user data centroids is obtained and set to the authentication feature of the valid user together with the stable centroids. In the run-time user authentication phase, after cleaning the continuous input data, UIKI performs a one-time clustering of the test data segments, and compares the clustering results with the stable centroids of the valid user and its fluctuation range to determine whether they are input data from the valid user. The experimental results show that UIKI has the advantages of high accuracy and low computational overhead, and it is suitable for non-interference and continuous authentication for mobile device users.

As for the future work, we plan to extend UIKI to the scenario of cross-device authentication in order to make it convenient for users, i.e., we use the training results on one device in the valid user modeling phase to the authentication on other devices in the runtime user authentication phase.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled “Identity authentication based on keystroke dynamics for mobile device users”.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants No. 61877015, No. 61902185, No. 61850410531, No. 61803338, No. 61572164, and No. 61473109, the Zhejiang Provincial Natural Science Foundation under Grants No. LY21F020028 and No. LY19F020016, the Jiangsu Provincial Natural Science Foundation under Grant No. BK20190448, and the Quality Project under Grant No. GK208802299013-103.

References

[1] S. Barra, A. Castiglione, F. Narducci, M. De Marsico, M. Nappi, Biometric data on the edge for secure, smart and user tailored access to cloud services, Future Gener. Comput. Syst. 101 (2019) 534–541, doi:10.1016/j.future.2019.06.019.

[2] J. Bonneau, The science of guessing: analyzing an anonymized corpus of 70 million passwords, in: Proceedings of the IEEE Symposium on Security and Privacy (S&P), 2012, pp. 538–552, doi:10.1109/SP.2012.49.

[3] M.M.H. Ali, V.H. Mahale, P. Yannawar, A.T. Gaikwad, Overview of fingerprint recognition system, in: Proceedings of the International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016, pp. 1334–1338, doi:10.1109/ICEEOT.2016.7754900.

[4] M.E. Fathy, V.M. Patel, R. Chellappa, Face-based active authentication on mobile devices, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 1687–1691, doi:10.1109/ICASSP.2015.7178258.

[5] D. Cho, K.R. Park, D.W. Rhee, Y. Kim, J. Yang, Pupil and iris localization for iris recognition in mobile phones, in: Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD), 2006, pp. 197–201, doi:10.1109/SNPD-SAWN.2006.58.

[6] X. Zhou, W. Liang, I. Kevin, K. Wang, H. Wang, L.T. Yang, Q. Jin, Deep learning enhanced human activity recognition for internet of healthcare things, IEEE Internet Things J. 7 (7) (2020) 6429–6438, doi:10.1109/JIOT.2020.2985082.

[7] A. Casanova, L. Cascone, A Castiglione, M. Nappi, C. Pero, Eye-movement and touch dynamics: a proposed approach for activity recognition of a web user, in: Proceedings of the 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), 2019, pp. 719–724, doi:10.1109/SITIS.2019.00117.

[8] A. Castiglione, K.K.R. Choo, M. Nappi, S. Ricciardi, Context aware ubiquitous biometrics in edge of military things, IEEE Cloud Comput. 4 (6) (2017) 16–20, doi:10.1109/MCC.2018.1081072.

[9] C. Shen, Y. Zhang, X. Guan, R.A. Maxion, Performance analysis of touch-interaction behavior for active smartphone authentication, IEEE Trans. Inf. Forensics Secur. 11 (3) (2015) 498–513, doi:10.1109/TIFS.2015.2503258.

[10] C. Giuffrida, K. Majdanik, M. Conti, H. Bos, I sensed it was you: authenticating mobile users with sensor-enhanced keystroke dynamics, in: Proceedings of the 11th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), 2014, pp. 92–111, doi:10.1007/978-3-319-08509-8_6.

[11] C. Shen, T. Yu, S. Yuan, Y. Li, X. Guan, Performance analysis of motion-sensor behavior for user authentication on smartphones, Sensors 16 (3) (2016) 345, doi:10.3390/s16030345.

[12] F. Monrose, A.D. Rubin, Keystroke dynamics as a biometric for authentication, Future Gener. Comput. Syst. 16 (4) (2000) 351–359, doi:10.1016/S0167-739X(99)00059-X.

[13] D. Gunetti, C. Picardi, Keystroke analysis of free text, ACM Trans. Inf. Syst. Secur. 8 (3) (2005) 312–347, doi:10.1145/1085126.1085129.

[14] A. Messerman, T. Mustafić, S.A. Camtepe, S. Albayrak, Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics, in: Proceedings of the International Joint Conference on Biometrics (IJCB), 2011, pp. 1–8, doi:10.1109/IJCB.2011.6117552.

[15] A. De Luca, A. Hang, F. Brudy, C. Lindner, H. Hussmann, Touch me once and i know it's you!: implicit authentication based on touch screen patterns, in: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI), 2012, pp. 987–996, doi:10.1145/2207676.2208544.

[16] R. Janakiraman, T. Sim, Keystroke dynamics in a general setting, in: Proceedings of the 2007 International Conference on Biometrics (ICB), 2007, pp. 584–593, doi:10.1007/978-3-540-74549-5_62.

[17] E. Al Solami, C. Boyd, A. Clark, I. Ahmed, User-representative feature selection for keystroke dynamics, in: Proceedings of the 5th International Conference on Network and System Security (NSS), 2011, pp. 229–233, doi:10.1109/ICNSS.2011.6060005.

[18] C.J. Tsai, K.J. Shih, Mining a new biometrics to improve the accuracy of keystroke dynamics-based authentication system on free-text, Appl. Soft Comput. 80 (2019) 125–137, doi:10.1016/j.asoc.2019.03.033.

[19] X. Zhou, B. Wu, Q. Jin, User role identification based on social behavior and networking analysis for information dissemination, Future Gener. Comput. Syst. 96 (2019) 639–648, doi:10.1016/j.future.2017.04.043.

[20] P.S. Teh, S. Yue, A.B.J. Teoh, Improving keystroke dynamics authentication system via multiple feature fusion scheme, in: Proceedings of the International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012, pp. 277–282, doi:10.1109/CyberSec.2012.6246096.

[21] D.R. Gentner, Keystroke timing in transcription typing, in: W.E. Cooper (Ed.), Cognitive Aspects of Skilled Typewriting, Springer, New York, NY, 1983, pp. 95–120, doi:10.1007/978-1-4612-5470-6_5.

[22] A.K. Belman, V.V. Phoha, Discriminative power of typing features on desktops, tablets, and phones for user identification, ACM Trans. Priv. Secur. 23 (1) (2020) 1–36, doi:10.1145/3377404.