Typing Signature Classification Model for User Identity Verification

Tapalina Bhattasali, Rituparna Chaki, Khalid Saeed and Nabendu Chaki

Abstract Typing pattern is a behavioral trait of user that is simple, less costly, and workable at any place having only computing device. In this paper, n-graph typing signature is built during user profiling based on keyboard usage pattern. The main aim of this paper is to increase inclusion of number of typing features (both temporal and global) during decision generation and to simplify the procedure of considering missing typing patterns (various monographs, digraphs, etc), which are not enrolled before. A modular classification model collection–storage–analysis (CSA) is designed to identify user. Typing signature becomes adaptive in nature through learning from environment. Module 1 is used for pattern acquisition and processing, and module 2 is used for storage, whereas module 3 is used for analysis. Final decision is generated on the basis of evaluated match score and enrolled global parameters. Proposed CSA model is capable to reduce space and time overhead in terms of dynamic pattern acquisition and storage without using any approximation method. A customized editor HCI is designed for physical key-based devices to build our own data set. Proposed CSA model can classify typing signature of valid and invalid user without incurring high overhead.

Keywords Classification model · Typing signature · n-graph
Wildcard character · Identity verification

T. Bhattasali (✉) · R. Chaki · N. Chaki
University of Calcutta, Kolkata, India
e-mail: tapalina@ieee.org

R. Chaki
e-mail: rchaki@ieee.org

N. Chaki
e-mail: nabendu@ieee.org

K. Saeed
Bialystok University of Technology, Bialystok, Poland
e-mail: k.saeed@pb.edu.pl

© Springer Nature Singapore Pte Ltd. 2018

R. Chaki et al. (eds.), *Advanced Computing and Systems for Security*,
Advances in Intelligent Systems and Computing 666,
https://doi.org/10.1007/978-981-10-8180-4_4

1 Introduction

In order to control resource access effectively, reliable identity verification mechanism needs to be used. User identity based on biometric features is more efficient than use of only password. Biometric techniques provide a solution to ensure that the desired services are accessed only by a legitimate user and no one else. The main reason behind high reliability of biometric features to represent user's identity compared to many other traditional mechanisms is that it cannot be stolen like password. Trustworthiness of low cost password-based authentication can be increased by analyzing typing pattern of users. It is a behavioral nature which can be captured by the way individual types on a keyboard.

Several studies [1, 2] show that individual's typing patterns are stored as template and can be effectively used for identification. Timing vectors are mainly used to classify patterns as valid or invalid.

Various features extracted from typing events can be divided into temporal features and global features. Temporal features represent timing data for typing of specific key. These features are calculated based on the time stamps when the key is pressed and released. Global features refer to typing pattern of the user, such as frequency of errors, frequency of using control keys such as caps lock, num lock, shift, alt (e.g., left or right), overall typing speed. It is cheaper to implement than other biometrics as no additional pattern acquisition device is required. Typing pattern can be classified as fixed [3] and free [4]. Fixed pattern is based on pre-defined content, which makes it more prone to forgeries. On the other hand, free pattern [5] is based on random typing on keyboard, which makes it more challenging for user profile creation.

This work is extended version of one of our previous works [6] on modular classification model to validate typing signature pattern [7]. This paper considers both fixed and free and long and short patterns. The major contribution of this work is to use the concept of wildcard character ('*') for n-graph (where n = 1, 2, 3,…, m) creation during feature processing at module 1. Average key latencies are extracted from frequently used n-graphs (grouped on the basis of wildcard character) to deal with the n-graphs which are not available during enrollment. Use of generalized wildcard character-based n-graph for used patterns (patterns which are frequently used) enhances flexibility. Wildcard characters also reduce the sparse entries of template to improve performance of a classification model. Here, several global features (typing length, frequency, and count of using different types of patterns) are also considered during decision generation as outer layer of pattern matching along with temporal features (key time, typing duration, time per key pressed, typing speed for both fixed and free patterns) to enhance accuracy of classification model. Here, typing patterns are collected through HCI interface.

The remaining part of this paper is structured as follows: Section 2 presents a survey of some well-known research works on this domain. Section 3 presents research gap analysis, Sect. 4 introduces proposed classification model to validate typing signature for user identity verification, and Sect. 5 presents the analysis part. Finally, Sect. 6 concludes the paper.

2 Literature Survey

In general, keystroke analysis [8–10] is based on the traditional statistical analysis or pattern recognition techniques. Drawbacks of both neural networks and statistical methods in terms of search times are identified. It is claimed that performance of keystroke-based authentication [11] is better than vein pattern recognition and is similar to fingerprint and voice recognition for Internet-based authentication. Several classifiers are used with the trade-off between computation and performance. In this paper, only standard keyboard is considered for user data inputs as in the works [10, 12, 13]. There are several works on keystroke authentication based on either fixed text or free text. Users can be identified using either one-time verification or multi-time verification (continuous or periodic mode). In [14], key press interval is taken as a signature identifier. Implementation is platform-independent and does not require excessive computational power. However, this logic deals with fixed text, and data are statistically analyzed to determine keystroke patterns.

Ahmed et al. [5] proposed a free-text keystroke dynamics-based authentication, where raw data (flight time for digraph, dwell time for monograph) are collected, processed, and converted to digraph and monograph formats by using approximation method to consider missing patterns during enrollment. To illustrate the concept, we have briefly presented the procedure of digraph approximation for missing digraph.

Average flight times are computed for the digraphs ending with given characters.

Digraph key orders are computed by sorting average flight times.

Missing digraphs are estimated using digraph key order and considered as input of digraph neural network.

Outliers are removed from monograph and digraph sets during enrollment by Pierce’s condition. After removing outlier, they are passed through sorting modules, which process data and calculate mapping tables for future use as a part of the signature. Missing digraphs need to be approximated by calculating average flight times for all the digraphs in the provided sample. User behavior for monographs is a 2-D relation between key order and its dwell time. Weights of the trained networks are also considered as a part of user’s signature.

Sometimes, nature of variations between multiple valid keystroke entries contains sufficient discriminatory information to improve keystroke authentication [9]. Variation in typing sequences is independent of typing proficiency unlike other parameters. Variations in the event sequences decrease significantly, if users are familiar with typing of a specific string. Collected raw data including pressed key, time stamps of the key events, IP address, browser type, date and time of submission are submitted to the back-end server. Any key can be typed for 12-character-long password having format of "SUUDLLLLDUUS", where S is any symbol, U stands for uppercase letters, L stands for lowercase letters, and D stands for numeric digit. As for example, user types string "HJealth", instead of "Health". One possible event sequence for this is {RSdHduRSu, LSd, J, LSu, BKdu, edu, adu, ldu, tdu, hdu}. Here, use of either right shift (RS) key or left shift (LS) key to type uppercase letters results into multiple events. Sliding window technique is considered on each subject pair to calculate the percentage of unique event sequences.

Users' typing patterns are also continuously monitored for authentication [10]. To reduce dimensionality, a feature vector is extracted from each input stream of keystrokes (session) and digraphs are clustered based on the temporal features. Although digraphs and their corresponding interval times are analyzed, authors claimed that it can work well with any n-graph and their temporal features as well as with any classification algorithm. This clustering logic shows better performance than k-means algorithm. Optimal number of n-graphs is required for cluster formation to show accuracy.

It is observed from the above study that larger sample size gives rise to better accuracy in case of free-text authentication [15]. However, shorter enrollment period is better suited for the security perspective. Major requirement of keystroke accuracy is to include greater number of participants and collect multiple samples for a long period of time. It is found that if users are unable to log in, or accepted at low rate, their keystroke patterns are inconsistent in nature, which may increase false detection.

3 Research Gap Analysis

According to the literature survey [5, 8–10], a few generalized issues are identified to draw more attention of the researchers.

Most of the keystroke-based authentication procedures consider only the temporal parameters like dwell time and flight time from the users' typing patterns, which may vary with time and are not capable to produce accurate result. There are still few works that also consider global parameters such as typing sequences, count of errors during typing, habit of typing, stylometry to increase reliability of users' typing profile. Accurate authentication is guaranteed through use of larger sample size, which is not always available. Run-time verification of random typing patterns still remains an open research issue.

Although there are several works which can be used as a base of future research, we have found Ahmed et al.'s [5] work more interesting. It is considered as a base of our research work from the perspective of dynamic pattern collection and storage as they have provided a solution on:

How to match typing features of monographs and digraphs of a user during verification, which were missing during enrollment?

A few issues addressed by Ahmed et al. are considered here as background of our research work. In Ahmed et al. [5] work:

- Limitations of enrollment process are removed by approximation technique known as sorted time mapping (STM). Only flight time is considered for digraph approximation, and dwell time is considered for monograph approximation. Missing digraphs are approximated by calculating average flight times, and monographs are considered as a 2-D relation between key order and its dwell time.
- Separate mapping tables need to be generated and stored, and separate neural network classifier models need to be designed for each type of mapping table.
- Samples are passed through sorting modules before generation of mapping tables.
- Outliers are removed by Peirce's criterion based on statistical analysis of the Gaussian distribution without depending on collected samples.

Therefore, it can be said that Ahmed et al. proposed a flexible and adaptive logic, which may give rise to computation-intensive procedures that need to be simplified.

3.1 Problem Statement

On the basis of background of our work, we can state our research problem as follows:

How to analyze random typing patterns without maintaining separate templates and separate classification models for different graphs (digraphs, monographs, etc) along with minimizing computational overhead in terms of space and time and reducing the probability of false detection of typing signature without using approximation method?

3.2 Scope of Work

Scope of our research work can be summarized as follows:

- To consider generalized n-graph signature instead of considering monograph signature and digraph signature separately.
- To include more typing features (both temporal and global) of users.
- To reduce computational complexity in terms of space and time overhead for pattern acquisition and storage.
- To increase reliability of outlier detection based on collected pattern.
- To include simple normalization procedure.
- To design efficient classifier to identify valid or invalid user based on typing signature patterns.

4 Proposed Model

Our objective is to design a modular classification model, which can classify valid and invalid typing signature of users based on random typing patterns [6, 16] to ensure flexibility, reduce requirements of storage space and processing time. This model uses typing pattern as input and produces a decision based on pattern matching. Typing signatures are created using temporal parameters of typing patterns and weight of network model used by classifier. Classification model is activated during enrollment phase and verification phase. Enrollment phase includes data acquisition, feature extraction, template generation, storage, and learning. Verification phase is further divided into pattern matching and decision making. Terminologies used in this paper are presented in Table 1.

4.1 Typing Signature Classification Model

Typing signature classification model CSA includes module 1 (C) for collecting data, module 2 (S) for storage, and module 3(A) for analysis. CSA model is designed based on both functional (F) and non-functional (NF) requirements. F has higher priority compared to NF. F includes identity verification, whereas NF includes anomaly detection (classification between human and bot to avoid synthetic forgeries [17]). Identity verification includes two subclasses: distinction class (1: m verification, where n = m) and authentication class (1: 2 verification, where n = 2). Outputs of

Table 1 Terminologies

Terminologies	Meaning
Match score generator	Considers classifier output, average deviation, threshold, biasness to generate match score
Decision maker	Checks match score and relevant temporal and global parameters to check if claimed user is an authentic user or not
P_{in}	Number of processing unit in input layer is set to number of selected features of timing vectors
P_{out}	Number of processing unit in output layer is set to 1, because the output is known, i.e., either valid (1) or invalid (0)
P_{hi}	Number of processing unit in hidden layer is set to (Pin + Pout)/2
α	learning_factor
ŋ	adjust_factor
wt_{diff}	Difference in weight update
th_err	threshold_error
th_success	threshold_success
$node_{input}$	Number of input nodes

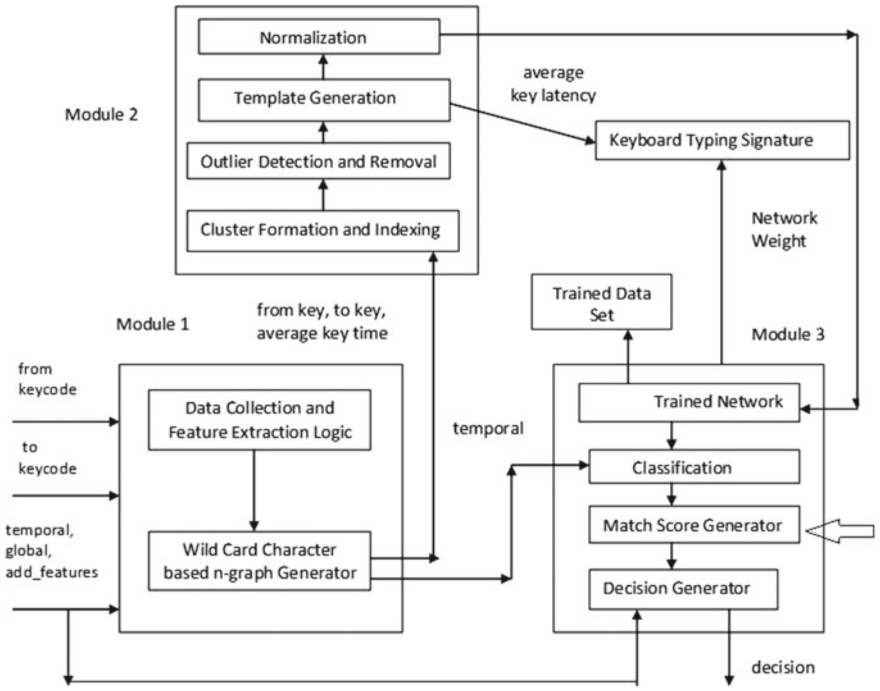

Module 2
Normalization
Template Generation
Outlier Detection and Removal
Cluster Formation and Indexing
average key latency
Keyboard Typing Signature
Network Weight
Trained Data Set
Module 3
Module 1
from key, to key, average key time
from keycode
to keycode
temporal, global, add_features
Data Collection and Feature Extraction Logic
Wild Card Character based n-graph Generator
temporal
Trained Network
Classification
Match Score Generator
Decision Generator
decision

Fig. 1 Three modules of CSA model

classification model are categorized into two classes: valid class and invalid class. Valid class is used to check whether user is claimed one or not (authentication needs to check only profile of claimed user) and whether user is enrolled before or not (distinction needs to check all the profiles enrolled before). Invalid class may include user not possessing claimed identity (having no malicious intension), imposters (having malicious intension), bots (valid user's profile is generated to behave as valid user artificially). Figure 1 represents three modules of CSA model.

Typing signature classification model includes two tuples: {keyboard_typing_signature, add_feature}. Reference signature includes user-id, template vectors of frequently used typing patterns (used patterns), neural network weight. Typing patterns can be updated with time based on feedback path of classification model. Network weights are updated through learning phase. add_feature includes several temporal (which are not included for signature generation) and global features required during decision generation. Typing pattern classification is defined as a model having the following tuples: {fixed pattern, free pattern, long pattern, short pattern, temporal features, global features}. It is the example of multi-feature model. It is assumed that CSA model blocks any user for that time after two consecutive invalid attempts. Modules of our classification model are presented below.

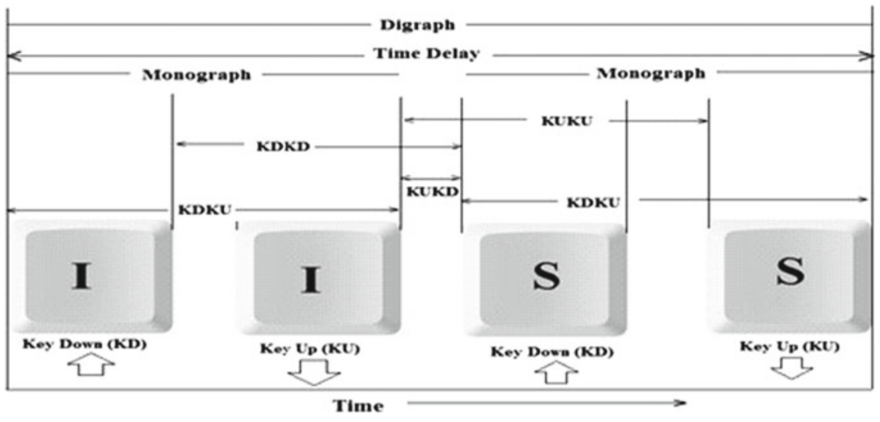


Fig. 2 Temporal data of keystroke for n-graph where n = 2

Module 1 is used for raw data collection, feature selection, and feature extraction. Any character of the keyboard can be typed. Application collects and submits the following information keys pressed and time stamps of the key events, date and time of submission, device-id, IP address. Raw data include key code, time stamp, time interval (KDKD (key-down, key-down), KUKU (key-up, key-up), KDKU (key-down, key-up), KUKD (key-up, key-down)). Time interval data of keystroke analysis (in milliseconds) are considered for n-graph. Figure 2 represents keystroke data for n-graph where n = 2.

- KDKU—Time gap between key press and key release (dwell time).
- KUKD—Time gap between release of previous key and press of next key (flight time).
- KDKD—Time gap between two successive key presses.
- KUKU—Time gap between two successive key releases.
- Key time—Time gap between key press of first key and key release of last key of n-graph.
- Average key time—Average of key times of n-graph.
- Average key latency—Average of all average key times of all same categories n-graph.

Typing patterns are classified into two categories: used patterns and rare patterns. Used pattern includes alphabet keys, whereas numeric and special characters are considered as rare patterns. Typing speed is calculated from the length of a string divided by key latency. Other parameters are count of pressing backspace, probability of using shift key or caps lock. It considers user's habit to type a sequence of string among all available possibilities; e.g., to type "Health", one typing sequence is

Table 2 Sample data during feature extraction

n-graphs (number of keys)	From key code	To key code	Key latency (ms)
and (3)	97	100	40
A (1)	15	15	20
is (2)	105	115	33

Table 3 Processed sample data

From key code	To key code		n-graph	Key time (ms)
97	100	a*d	and (3) around (1) acid (4)	• T1 • T2 • T3

Table 4 Reduced sample data

From key code	To key code	String (n-graph)	Average key time (ms)
97	100	a*d	(T1 + T2 + T3)/3

$<\text{shift}_{\text{down}},\ \text{h}_{\text{down}},\ \text{h}_{\text{up}},\ \text{shift}_{\text{up}},\ \text{e}_{\text{down}},\ \text{e}_{\text{up}},\ \text{a}_{\text{down}},\ \text{a}_{\text{up}},\ \text{l}_{\text{down}},\ \text{l}_{\text{up}},\ \text{t}_{\text{down}},\ \text{t}_{\text{up}},\ \text{h}_{\text{down}},\ \text{h}_{\text{up}}>$. There are various other combinations to type it. User's sample feature vector is presented in Table 2.

In the proposed logic, concept of wildcard character is used. n-graph-based logic considers "from key" and "to key" of any string. As for example, average key latency of A-A (monograph) and average key latency of a-d (digraph, n-graph) are considered. n-graph uses concept of wildcard character, i.e., where string is "a*d", '*' is wildcard character, starting character is 'a', and ending character is 'd', and the key time (kt) is calculated by considering time difference between first key and last key of n-graph, divided by n number of strings included within "a*d". Key latency (kl) sums up all available average key time (akt) of a specific wildcard character-based n-graph. Average key latency (akl) is used to reduce the size of the sample, which is a major requirement to transmit data in client–server environment. The concept of extracted feature is given in Tables 3 and 4.

Data Collection and Feature Extraction Logic

```
Input: U number of users, S number of samples per user
Output: matrix for each user containing from key code, to key code, average key
        latency in milliseconds
begin
for i=1 to U  for j=1 to S
  capture_event (key_press, key_release)
  if $(n+1)^{th}$ key code = 32  then,  // space as string separator
    ngraph = concat([1,2,...,n]$^{th}$ key)
      from_keycode = key code of 1st key
      to_keycode = key code of nth key
      write ngraph
      write 1st key $time_{keypress}$
      write nth key $time_{keyrelease}$
 if 1st key $time_{keypress}$ < nth key $time_{keyrelease}$ then,
    kt= |nth key $time_{keyrelease}$ -1st key $time_{keypress}$|   //key time
 else
     exit
 if 1st key = ‘x’ and nth key = ‘x’ then,
    $ngraph_{wildcard(x*)}$ = {1st key} ∪ {‘*’}
   // such as “a*”- start key ‘a’ and end key ‘a’
 if 1st key = ‘x’ and nth key = ‘y’ then,
  // such as “a*d”- start key ‘a’ and end key ‘d’’
 $ngraph_{wildcard(x*y)}$ = {1st key} ∪ {‘*’} ∪ {nth key}
 akt=(|nth key $time_{keyrelease}$ -1st key $time_{keypress}$|)/n  // average key time
 endfor
 p =count ($ngraph_{wildcard(x*)}$)
 q =count ($ngraph_{wildcard(x*y)}$)
 akl for $ngraph_{wildcard(x*)}$ = (∑akt)/p
 akl for $ngraph_{wildcard(x*y)}$= (∑akt)/q
 creatematrix (l×l)
 add_item(matrix_id, from_keycode, to_keycode, akl)
 endfor
 end
```

Module 2 is used for cluster formation from sample data, outlier detection (noisy data removal), normalization, template generation, and storage. This module is used for grouping filtered data to reduce sample size and to make searching faster. There are two major differences of proposed cluster formation with k-means algorithm. The first is that the number of clusters does not need to be specified in advance in this approach. The second is that indexing is used to reduce time complexity of k-means

[18] algorithm. Index database is used to search the location of a sample in a faster way. Clusterization of user's data is based on temporal information of used and rare patterns. Each cluster is partitioned into subclusters according to number of typable characters within a string. Intra-cluster difference (difference between a centroid and the sample data) and inter-cluster difference (difference between centroids of two clusters) are calculated to differentiate between valid and invalid data samples. Intra is used to measure the compactness of the clusters. Inter is used to measure the separation of the clusters. The logic of cluster formation is presented below.

Indexed Clustering Logic

```
Input: Dataset ds containing S data samples for U users
Output : p number of clusters  and p number of index
begin
   pick randomly k of n data points pt as cluster centre(cc)
for cc = 1 to k,
for pt = 1 to n,
 calculate d = distance(x1,y1,x2,y2) // distance between pt and cc
 calculate $d_{min}$
   add (cc, pt, $d_{min}$)
endfor
endfor
for each cc data point $m_i$ and for each non-cc data point $o_i$,
swap $m_i$ and $o_i$
endfor
 config_cost= compute(pt, d, cc)
for each cluster c,
   closeness= compute ($x_i$, $x_{mean}$,pt, n)
calculate $config\_cost_{min}$
   set cc = newcc($config\_cost_{min}$)
    calculate intra                    //  Intra-Cluster distance
   calculate inter                   // Inter-Cluster distance
  if no change in cc then,
    continue
    generate index for each cluster
validity=  intra/inter
endfor
end
```

Outliers of the extracted features are detected to remove noisy data during cluster formation at module 2 by distant outlier detection logic presented below.

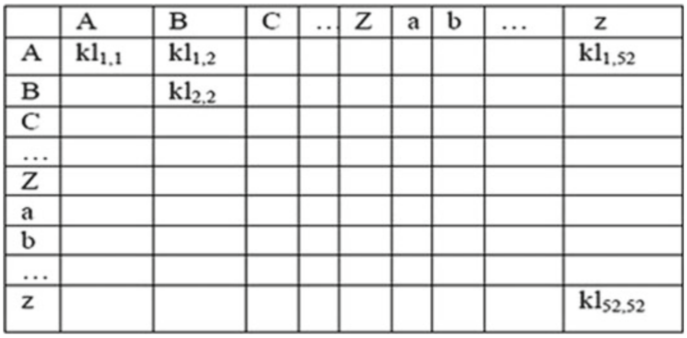
Distant Outlier Detection Logic

```
    Input:Dataset ds containing S data samples, p number of clusters
    Output: Set of Outliers OUT
  begin
     for each cluster $C_i$
       set $radius_i$ = radius of cluster $C_i$
     for each point $o_j$ in $ds_i \in C_j$
       if d>= $radius_i$ then,
          OUT = OUT $\cup$ $o_j$
      else if $o_j$belongs to both $C_i$ and $C_j$
          OUT = OUT $\cup$ $o_j$
    endfor
    endfor
    delete outlier set OUT
  end
```

Outlier set includes any pt, which is not included within any cluster, which is included in more than one cluster, which is on the boundary of the cluster. Outlier removal technique proposed in [8] is simplified here by using standard deviation method to remove noisy data during cluster formation.

Template is generated from the clustered data. There are several clusters formed to record user's typing patterns. Template stores the keystroke signatures of a user. Temporal data of used patterns of keystroke (alphabets) are normally used for template generation. Single template for each user that enables proposed logic to distinguish between legitimate user and impostor with minimum rate of error is considered. A 52×52 matrix is formed to represent time delay to press a key or time delay between two keys. Both upper case and lower case characters are considered. Diagonal elements represent monographs, and non-diagonal elements represent digraphs. Sparse entries are discarded to reduce template size, and frequently used n-graphs (monographs and digraphs) are considered during data normalization. $kl_{i,j}$ represents here average key latency for monographs, digraphs as well as n-graphs. Sample template in matrix format is provided in Fig. 3.

Each cell of the template matrix contains key latency. In matrix, the common cell between A and M includes average of all time latency of n-graphs whose starting character is 'A' and ending character is 'M'. Template is generated in such a way so that size of the template is reduced. Data are normalized and changed into a form recognizable by module 3 classifier. Data normalization logic is presented below using MinMax [19] logic.



	A	B	C	..	Z	a	b	...	z
A	$kl_{1,1}$	$kl_{1,2}$							$kl_{1,52}$
B		$kl_{2,2}$							
C									
...									
Z									
a									
b									
...									
z									$kl_{52,52}$

Fig. 3 Template format

Data Normalization Logic

```
Input: Timing Vectors of Sample Data (array of average key latency) $T_{i,j}$ for User
       $U_k$
Output: Normalized mean value of ($T_{i,j(Uk)}$)
begin
  sort $T_{i,j}$ in ascending order, $T_{i,j(Uk)} < T_{i,j+1(Uk)}$ determine
  minval of $T_i$ of $U_k$ // minimum value for i=1 to n
                                        number of timing vector of $U_k$
  determine maxval of $T_i$ of $U_k$ // maximum value for i=1 to n
                                        number of timing vector of $U_k$
  calculate mean value of $T_{i(Uk)}$
  calculate $minval_{new}$ and $maxval_{new}$ of $T_{i,j(Uk)}$ // new range for normalization
  caculate normalized value of $meanT_{i(Uk)}$
  calculate norm_mean_$T_{i(Uk)}$ within range [ $minval_{new}$= 0.0 and $maxval_{new}$=1.0]
end
```

To reduce complexity, mean value of timing vectors is considered here within the range of [0.0, 1.0]. For this reason, $minval_{new}$ is set to 0.0 and $maxval_{new}$ is set to 1.0.

Module 3 is used for training and testing of data sets. User's reference signature results from timing data of n-graph and the weights of the trained neural network with backpropagation (N2BP). Training configuration includes number of layers (input, hidden, output), activation function (sigmoid function), initial weight, termination condition. N2BP is trained with normalized data generated from module 2. During validation, collected input patterns are transmitted to pre-learned N2BP. Training logic is presented below.

Training Logic of N2BP

```
    Input:  data sample S, target values of each S, number of epoch, N2BP model
    Output: Trained N2BP model
begin
    initialize all weights and biases in network
    set weight to first component of input timing vector
    // for faster convergence
    initialize $\alpha = 0.1$
    initialize $^{\eta} = 0.9$
    for each epoch,
     do while($wt_{diff}$ <= th_err)
    for each training sample $TS_i$,
    for each input layer unit j,
         set output $O_j = I_j$endfor
         for each hidden layer unit j,
           set input $I_j$
           set output $O_j$
          endfor
    for each j in output layer,
             calculate $Error_j = O_j \times (1- O_j)(T_j - O_j)$    // Tj target value
    endfor
    for each j in hidden layer,
       calculate $Error_j = O_j \times (1- O_j) \times (T_j - O_j)$
     endfor
    for each weight wi,j,
        calculate $\Delta w_{i,j}$
        update weight $w_{i,j} = w_{i,j} + \Delta w_{i,j}$
    endfor
    for each bias bj,
      calculate $\Delta bj$
      update bias bj, $bj = bj + \Delta bj$
    endfor
   endfor
  endfor
end
```

Error data at the output layer are backpropagated to the earlier ones, allowing incoming weights to be updated until all training data have been used. Timing vector is taken as an input, comparing the current output with the target output and adjusting the weight values according to the backpropagation algorithm. Once the learning process completes, testing phase is activated to check whether the input pattern matches with previously stored template data of claimed user. N2BP model updates repeatedly until error is reduced to negligible amount. The verification logic is presented below.

Verification Logic of Sample by Trained N2BP **Input:**

```
  Input: Timing Vector $TV_i$ of claimed user $U_i$
  Output: Classification Decision either valid (1) or invalid (0)
begin
  initialize sc to 0
  for each component in input vector $TV_i$ of claimed User $U_i$
     calculate Signature $Sig_{(Ui)}=TV_i \times W_{i,j}$
                         // $W_{i,j}$ is weight of the network
     calculate abs_dif= $[(TV_i-W_{i,j})+ (RS(U_i)- Sig(U_i))]$
                         // RS(Ui) represents stored reference signature
  endfor
  calculate tolerance_limit=adjust_factor × variability
  if abs_dif<= tolerance_limit then,
       increment sc                    //success counter
  if sc / $node_{input}$>= th_success and targetval=outputval
                 or   targetval-outputval<0.0001 then,
       validity = 1
       update weight
  else
       validity = 0
  end
```

N2BP model may not generate exactly 0 or 1 for using sigmoid function. If allowable tolerance level of N2BP model <= 0.1, it is treated as 0, and if it is >=to 0.9, it is equal to 1. If testing of data for N2BP model results within 0 and 1, then testing data is treated as valid (1); otherwise, it is treated as invalid (0). Match logic evaluates score only on the basis of classification accuracy, average key latency, average deviation, weight, and biasness of the network.

Average deviation [6, 16] for n-graph is calculated as follows:

$$\Delta_{\text{ngraph}} = \text{difference}\left(\text{RSKL}_i, \text{kl}_i, \text{N}, 100\right)$$

where N is the total number of data set, kl_i is the key latency for the *i*th sample in user’s session, $RSKL_i$ is resulting from claimed user’s n-graph neural network model. Average deviation represents similarity in user’s behavior in a session to the valid user’s behavior. Lower number represents high similarity to ensure that session belongs to the same user. Match score is calculated as follows:

$$\begin{aligned} \text{match\_score} &= 1, \quad \text{if m is} <= \text{th}, \\ &\quad \text{where m} = \Delta \times \text{b, and bis biasness, this threshold} \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Table 5 Fixed pattern: .tie5Roanl (10-graph)

uid	Session	From key code	To key code	n-graph	Typing gap (in second)	Character per second (cps)
u1	s1	48	108	10	(Absolute value of KD of '.'-KU of 'l')/10-3	Typing gap/number of characters

Decision logic finally takes the decision based on match score and relevant features including parameters such as rare data pattern, user nature of typing to verify claimed identity.

5 Performance Analysis

Performance of proposed work is analyzed in MATLAB R2012b. At present, analysis is done in laboratory environment [20–22]. QWERTY keyboard is considered for data collection through Windows API. All the participants use same HP laptop (2.4 GHz Pentium 4 processor, with 2 GB of RAM, and running Windows 7 (64 bits)).

In order to analyze the proposed method, data are collected locally through user-friendly interface. Typing events are collected from QWERTY keyboard through Windows API. Besides considering popularly used parameters like true acceptance rate (TAR), true rejection rate (TRR), false acceptance rate (FAR), false rejection rate (FRR), equal error rate (EER), and accuracy, we have considered here parameters like failure to capture (FTC), average false acceptance, average false rejection.

FTC = (erroneous capture / total number of typing stroke)

FTC rate (%) = FTC × 100

Average False Acceptance = ∑Number of Imposters accepted as Genuine user for a specific threshold / count of false acceptance for that threshold

Average False Rejection = ∑Number of Genuine user rejected for a specific threshold / count of false rejection for that threshold

5.1 Analysis of the Collected Data

Fixed texts are treated as a password. Two passwords can be used alternatively: (i) .tie5Roanl (10-graph) and (ii) try4-mbs (8-graph). Another fixed pattern is used as optional pass code: —a quick brown fox jumps over the lazy dog: —containing all 26 alphabets (used pattern). Free texts are collected from users as they type sample text within 100–150 words. It can vary from user to user. Common texts are considered to differentiate between two users (inter-variance), whereas uncommon texts are considered to analyze typing pattern of the same user (intra-variance) (Tables 5 and 6).

Table 6 Fixed pattern: try4-mbs (8-graph)

uid	Session	From key code	To key code	n-graph	Typing gap (in second)	Character per second (cps)
u1	s2	116	115	8	Absolute value of KD of 't' -KU of 's')/10^{-3}	typing gap/number of characters

Table 7 Temporal data of free text of uid AB for session starts at 1:24:31 AM

Key	Key code	Key event time (ms)
I	73	22746326.54
space	32	22746326.70
a	97	22746326.94
m	109	22746327.16
space	32	22746327.31
a	97	22746327.45
space	32	22746327.63
d	100	22746327.78
o	111	22746327.92
c	99	22746330.23
t	116	22746330.55
o	111	22746330.85
r	114	22746331.14
	46	22746331.42

Table 8 Data of different users for fixed text and free text

uid	Fixed text typing duration	Time per key pressed for fixed text	Length of free text	Free-text typing duration	Time per key pressed for free text	Free-text typing speed
AB	14	1.4	69	93	1.34782608695652	0.741935483870968
PB	8	1	67	74	1.1044776119403	0.905405405405405
AD	5	0.625	85	69	0.811764705882353	1.23188405797101
MP	14	1.75	71	146	2.05633802816901	0.486301369863014
MB	16	1.6	68	192	2.82352941176471	0.354166666666667

Tables 7, 8, and 9 represent raw data collected from different users at module 1.

In typing signature analysis, there is a probability of failures during data acquisition. These failures are due to (i) users who want to type faster than their ability, (ii) users who are disturbed by the environment, (iii) users who know that their typing times are being saved, (iv) users who may not be familiar with using the keyboard. Failure to capture (FTC) can be calculated from erroneous captures and a total num-

Table 9 Typing patterns of different users (global parameters)

uid	Used patterns	Alphabets	Shift	Space	Rare patterns
AB	53	53	4	13	3
PB	154	144	12	34	12
AD	147	139	13	33	16
MP	49	49	0	12	2
MB	199	181	19	45	20

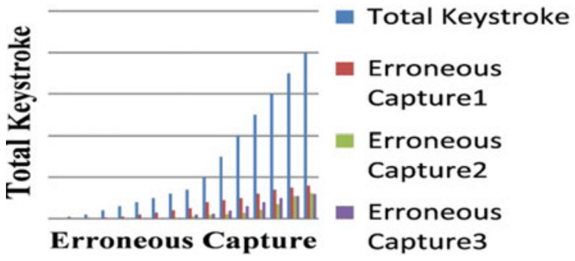


Fig. 4 Total keystroke versus erroneous capture of keystroke

Table 10 Free-text sample entry

n-graph	Common string	From key code	To key code	Average key latency (ms)	
				User 1	User 2
3	AND	65	100	12.5	20.1

ber of keystrokes. FTC in the proposed work is low, as there is less requirement of typing fixed text. It reduces the probability of mistakes. User can type text freely and can use any key of the keyboard during entire session. Figure 4 represents plot of total keystroke and erroneous capture. FTC for user 1 (erroneous capture1) is found to be 0.2, whereas FTC for user 2 (erroneous capture2) and user 3 (erroneous capture3) is 0.155 and 0.15, respectively. FTC of user 1 is analyzed for general fixed text-based authentication, whereas FTC of user 2 and user 3 is analyzed according to the proposed work. It shows that FTC rate is around 20% in general, whereas it is around 15% for our proposed work. It proves that FTC rate is improved in proposed work.

Sample size includes 35 users. Verification is done 10 times at each threshold level. Table 10 represents free-text sample entry.

In Fig. 5, average key latency is plotted against five different users for typing the same strings. Timing data of typing same n-graph vary for five different users.

Average false acceptance and average false rejection are calculated from Tables 11 and 12.

In Fig. 6, trade-off between average false acceptance and average false rejection is plotted as average false detection against different threshold levels. It shows that

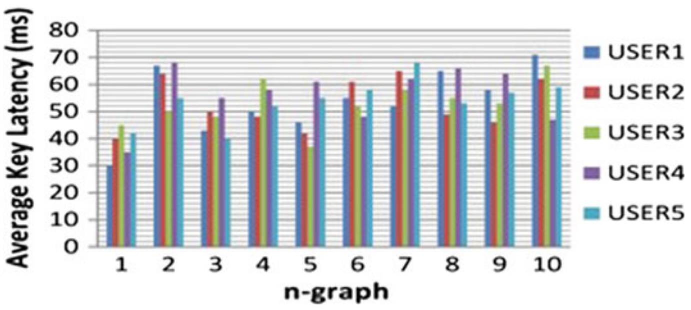


Fig. 5 Plot of average key latency for same n-graphs

Table 11 Average false acceptance and false rejection

Threshold	Number of invalid users accepted out of total sample size at different sessions	Number of valid users rejected out of total sample size at different sessions
5	<12,11,12,11,11,11,11,12,11,12>	<0,0,0,0,0,0,0,0,0,0>
10	<10,11,10,9,11,11,10,11,11,11>	<0,0,0,1,0,0,1,0,0,0>
15	<9,10,9,8,9,9,8,10,9,9>	<0,0,2,1,0,2,0,1,0,0,>
20	<7,8,7,8,9,7,7,8,7,7>	<0,0,2,1,2,1,0,0,2,0>
25	<7,7,6,7,6,7,7,6,7,8>	<1,3,2,2,0,0,0,0,0,0>
30	<6,6,7,6,6,6,6,6,6,6>	<3,0,0,2,3,0,0,0,0,0>
35	<5,5,6,7,5,6,4,5,5,5>	<3,0,0,0,0,3,0,3,0,0>
40	<3,4,3,4,4,5,4,3,4,4>	<1,3,2,1,0,0,0,2,0,0>
45	<2,3,3,4,2,3,3,3,4,3>	<2,1,0,0,1,0,2,3,0,0>
50	<2,3,2,1,3,2,3,2,3,1>	<2,0,0,2,0,4,0,0,1,0>
55	<1,2,1,1,2,3,0,1,2,2>	<4,0,4,0,0,0,0,1,0,0>
60	<2,1,1,0,0,2,2,1,2,2>	<3,2,1,0,0,0,3,0,0,0>
65	<1,0,1,1,0,1,1,2,0,1>	<4,0,2,1,0,2,0,0,0,2>
70	<1,0,0,0,1,0,1,1,0,1>	<1,3,0,0,4,0,0,2,1,0>
75	<1,0,0,1,1,0,0,0,0,0>	<3,0,4,1,0,0,0,0,2,2>
80	<0,0,0,1,0,0,0,1,0,0>	<4,2,0,3,0,0,0,3,2,1>
85	<0,0,0,0,0,0,0,0,0,0>	<5,4,6,5,4,4,5,4,4,4>
90	<0,0,0,0,0,0,0,0,0,0>	<8,9,7,7,8,7,7,7,7,8>
95	<0,0,0,0,0,0,0,0,0,0>	<12,12,11,10,12,11,11,12,11,11>

FAR is high and FRR is low when the threshold level is low. However, reverse is true when the threshold is high (Table 13).

Figure 7 represents comparative analysis of FAR and FRR of the proposed work at different threshold levels along with FAR and FRR of Ahmed et al. work with digraph approximation and without approximation method. According to analysis result, it is seen that proposed model works far better than the work without using any approximation method and is almost equivalent to approximation method with

Table 12 Calculation of FAR and FRR

Threshold (%)	Average false acceptance	FAR (%)	Average false rejection	FRR (%)
5	11.26	75.1	0	0.0
10	10.5	70.0	0.15	1.0
15	9.0	60.0	0.6	4.0
20	7.5	50.0	0.75	5.0
25	6.78	45.2	0.752	5.0
30	6.06	40.4	0.768	5.1
35	5.28	35.2	0.918	6.1
40	3.78	25.2	0.927	6.2
45	3.01	20.1	0.93	6.2
50	2.25	15.0	0.966	6.4
55	1.51	10.1	0.976	6.5
60	1.26	8.4	0.982	6.5
65	0.768	5.1	1.07	7.1
70	0.468	3.1	1.08	7.2
75	0.33	2.2	1.22	8.1
80	0.163	1.1	1.5	10.0
85	0	0.0	4.5	30.0
90	0	0.0	7.5	50.0
95	0	0.0	11.26	75.1

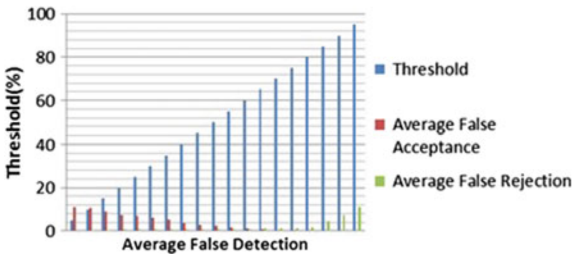


Fig. 6 Average false acceptance and average false rejection trade-off with threshold level

less overhead. Sometimes, proposed model even works better than the work using approximation method.

5.2 Discussion

The proposed work is based on generalized concept of n-graph-based classification model. Our classification enhances the flexibility compared to existing works [5, 8–10]. There is no requirement of maintaining mapping tables for monographs and digraphs or modeling the neural network separately [5]. It reduces time and space

Table 13 Comparative analysis of FAR and FRR

Threshold (%)	Ahmed et al. (with digraph approximation)		Ahmed et al. (without digraph approximation)		Proposed work (without digraph approximation)	
	FAR (%)	FRR (%)	FAR (%)	FRR (%)	FAR(%) (approx.)	FRR(%) (approx.)
5	71.671	0	98	0	75.1	0.0
10	68.443	0	96	0	70.0	1.0
15	56.184	0	84	5	60.0	4.0
20	52.201	0	80	6	50.0	5.0
25	48.291	0	78	8	45.2	5.0
30	44.093	0	65	10	40.4	5.1
35	38.149	0	60	10	35.2	6.1
40	29.021	0	45	10	25.2	6.2
45	19.428	0	35	15	20.1	6.2
50	16.09	0	25	15	15.0	6.4
55	12.162	0	15	30	10.1	6.5
60	7.308	0.082	10	40	8.4	6.5
65	3.851	1.711	9	50	5.1	7.1
70	0.92	1.711	8	50	3.1	7.2
75	0.92	4.82	5	60	2.2	8.1
80	0.0152	6.1	5	60	1.1	10.0
85	0.0152	12.82	3	70	0	30.0
90	0	48.6	2	80	0	50.0
95	0	91	2	90	0	75.1

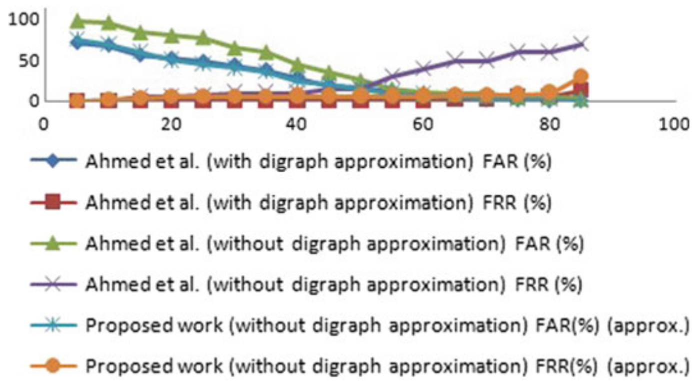


Fig. 7 Comparative analysis of FAR and FRR at different threshold levels

overhead. As the proposed work is based on the adaptive model of user's nature, it updates dynamically according to iterative learning process and enhances the flexibility. Feature extraction calculates average latency of n-graph in a very efficient way that reduces the feature set size. Simple clustering technique is used to group similar data samples in a cluster, where indexing technique is used for each cluster to reduce search time without enhancing the cost. Sparse entries in template are discarded to avoid operational complexity of classifiers. It may be discarded when it has no use; again, it can be added, according to the requirement, which consumes less resource. Odd data are removed in the simplest way compared to the other existing methods [5]. Data are normalized using a very simple logic. A reference typing signature of the user in the proposed model is based on temporal feature vector of user and the weight of network model, which can efficiently differentiate between two users. In summary, analysis of proposed model reveals that

- No mapping tables or separate neural network models for monographs and digraphs.
- Simple clustering technique groups similar data samples, where indexing technique is used for each cluster to reduce search time.
- Odd data are discarded based on intra-cluster and inter-cluster distance.
- Data are normalized using simple min-max logic.
- Both temporal and global features are considered for decision making.

6 Conclusions

The primary goal of the proposed model is to classify users accurately in a less complex manner. This does not introduce any additional costs and needs no additional hardware except the keyboard that every computer is equipped with. Dynamic pattern analysis is still a challenging issue. The proposed work focuses on this direction. Here, user's typing nature is captured besides considering temporal data of typing patterns. In module 1, typing patterns of users are recorded as they type short fixed patterns as well as long free patterns. Thus, the variations in the typing styles of individual users can be obtained easily. Sparse entries of user templates are reduced by using the wildcard character, and user's adaptability is enhanced by feedback path. This reduces the complexity faced by the classifiers. In module 2, clusters of data samples are formed and indexing is considered for each cluster for faster searching. In module 3, classifier decision is merged with the match score generator and decision generator, which enhance the accuracy level.

At a glance, the features of proposed model are as follows:

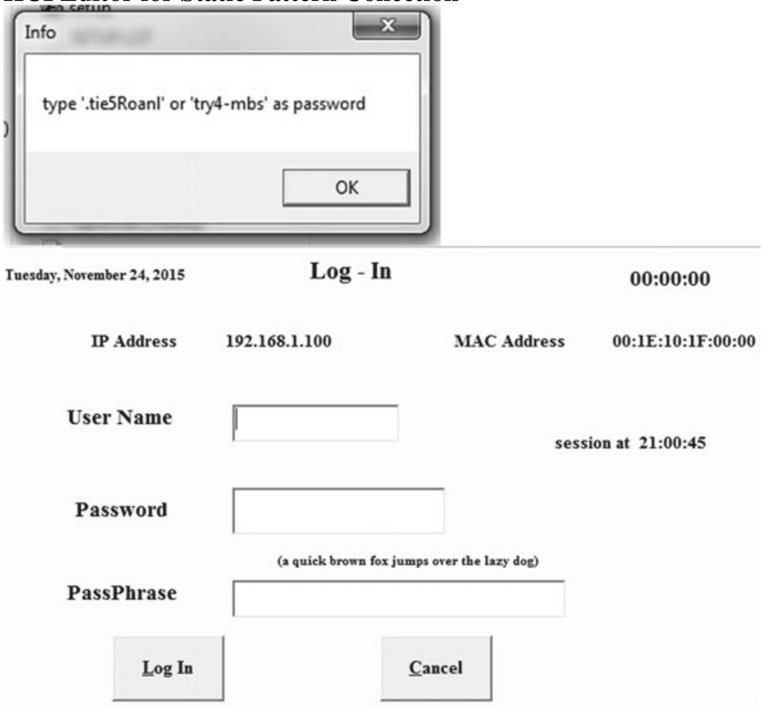
- Classification model considers temporal as well as global features of both static and dynamic patterns.
- Wildcard character ('*') is used to reduce the problem of matching patterns, which are not collected before.
- Size of template is reduced, and sparse entries are discarded to avoid processing overhead of classification model.
- Computation overhead is reduced along with maintaining accuracy for a limited set of samples in laboratory environment.

The authors believe that the proposed work could give better performance even in remote applications such as remote health care. The work is on to study various challenges in this domain and adjust the solution accordingly, so that it can be considered for further detailed processing and analysis.

Appendix

HCI User Interface

HCI Editor for Static Pattern Collection



Info

type '.tie5Roanl' or 'try4-mbs' as password

OK

Tuesday, November 24, 2015

Log - In

00:00:00

IP Address 192.168.1.100

MAC Address 00:1E:10:1F:00:00

User Name

session at 21:00:45

Password

(a quick brown fox jumps over the lazy dog)

PassPhrase

Log In

Cancel

HCI Editor for Dynamic Pattern Collection

References

1. Wang, Y., Du, G-Y., Sun, F.-X.: A model for user authentication based on manner of keystroke and principal component analysis. In: Proceedings of International Conference on Machine Learning and Cybernetics, pp. 2788–2792 (2006)
2. Balagani, K.S., Phoha, V.V., Ray, A., Phoha, S.: On the discriminability of keystroke feature vectors used in fixed text keystroke authentication **32**(7), 1070–1080 (2011)
3. Gunetti, D., Picardi, C., Karnan, M., Akila, M., Krishnaraj, N.: Biometric personal authentication using keystroke dynamics: a review. J. Appl. Soft Comput. **11**(2), 1515–1573 (2011). (Elsevier)
4. Gunetti, D., Picardi, C.: Keystroke analysis of free text. ACM Trans. Inf. Syst. Secur. **8**(3), 312–347 (2005)
5. Ahmed, A., Traore, I.: Biometric recognition based on free-text keystroke dynamics. IEEE Trans. Cybern. **44**(4), 458–472 (2014)
6. Bhattasali, T., Panasiuk, P., Saeed, K., Chaki, N., Chaki, R.: Modular logic of authentication using dynamic keystroke pattern analysis. ICNAAM, AIP Publ. Am. Inst. Phys. **1738**, 180012 (2016)
7. Bhattasali, T., Saeed, K.: Two factor remote authentication in healthcare. In Proceedings of IEEE International Conference on Advances in Computing, Comunications and Informatics, pp. 380–381 (2014)
8. Giroux, R.S., Wachowiak, M.P.: Keystroke based authentication by key press intervals as a complementary behavioral biometric systems. In: Proceedings of IEEE International Conference on Man and Cybernetics, pp. 80–85 (2009)
9. Syed, Z., Banerjee, S., Leveraging, B.C.: Variations in event sequences in keystroke dynamics authentication systems. In: Proceedings of IEEE International Symposium on High-Assurance Systems Engineering, pp. 9–11 (2014)
10. Shimshon, T., Moskovitch, R., Rokach, L., Elovici, Y.: Clustering Di-graphs for continuously verifying users according to their typing patterns. In: Proceedings of IEEE Convention of Electrical and Electronics Engineers in Israel, pp. 445–449 (2010)
11. Karnan, M., Krishnaraj, N.: Bio password—keystroke dynamic approach to secure mobile devices. In: Proceedings of IEEE International Conference on Computational Intelligence and Computing Research, pp. 1–4 (2010)

12. Hu, J., Gingrich, D., Sentosa, A.: A K-Nearest neighbor approach for user authentication through biometric keystroke dynamics. In: Proceedings of IEEE International Conference on Communications, pp. 1551–1510 (2008)
13. Killourhy, K.S., Maxion, R.A.: Comparing anomaly-detection algorithms for keystroke dynamics. In: Proceedings of IEEE/IFIP International Conference Dependable Systems and Networks, pp. 125–134 (2009)
14. Araujo, L.C.F., Sucupira, L.H.R., Lizarraga, M.G., Ling, L.L., YabuUti, J.B.T.: User authentication through typing biometrics features. IEEE Trans. Signal Process. **53**(2), 851–855 (2005)
15. Killourhy, K.S., Kevin, S., Maxion, R.A., Roy, A.: Free versus transcribed text for keystroke dynamics evaluations. In: Proceedings of Workshop: Learning from Authoritative Security Experiment Results, pp. 1–8 (2012)
16. Bhattasali, T., Saeed, K., Chaki, N., Chaki, R.: Bio-authentication for layered remote health monitor framework. J. Med. Inf. Technol. **23**(2014), 131–140 (2014)
17. Jain, K., Ross, A., Pankanti, S.: Biometrics: a tool for information security. IEEE Trans. Inf. Forensics Secur. **1**(2), 125–143 (2001)
18. Kao, B., Lee, S.D., Lee, P.K.F., Cheung, D.W., Ho, W.S.: Clustering uncertain data using voronoi diagrams and r-tree index. IEEE Trans. Knowl. Data Eng. **22**(9), 1219–1233 (2010)
19. Xie, Q.Y., Cheng, Y.: K-Centers min-max clustering algorithm over heterogeneous wireless sensor networks. In: Proceedings of IEEE Wireless Telecommunications Symposium, pp. 1–6 (2013)
20. Giot, R., El-Abed, M., Hemery, B., Rosenberger, C.: Unconstrained keystroke dynamics authentication with shared secret. Elsevier Comput. Secur. **30**(1–7), 427–445 (2011)
21. Upmanyu, M., Namboodiri, A.M., Srinathan, K., Jawahar, C.V.: Blind authentication: a secure crypto-biometric verification protocol. IEEE Trans. Inf. Forensics Secur. **5**(2), 255–218 (2010)
22. Montalvao, J., Freirem, E.O., Bezerra, M.A., Garcia, R.: Empirical keystroke analysis in passwords. In: Proceedings of ISSNIP/IEEE Biosignals and Biorobotics Conference: Bio signals and Robotics for Better and Safer Living (BRC), pp. 1–6 (2014)