Towards Generating a Unique Signature for Remote User by Keystrokes Dynamics

Puja Mukherjee, Rituparna Chaki

Towards Generating a Unique Signature for Remote User by Keystrokes Dynamics

Puja Mukherjee [1], Rituparna Chaki[2]

A.K Choudhury School of IT, Universiy of Culcutta, JD Block, Sector III, Salt Lake City, Kolkata, West Bengal 700098, India

pujamukherjee2014@gmail.com1, rituchaki@gmail.com2

Abstract. Keystrokes dynamics has been used for quite sometimes in authentication of users. The technique has immense possibilities due to ease of implementation and un-obtrusive nature. Researchers have been working for attaining improved accuracy rate of user identification. Such techniques are validated using standard data-set. As it turns out, the quality of input data is very much important for generating an accurate use pattern vector. In this paper, an application for data collection has been presented. The application, besides creating a user data-set, also generates a signature vector database.

Keywords: keystrokes, remote user, free-text, key hold time & key dwell time, signature vector, authentication.

1 Introduction

In this era enormous use of automated system together with the cloud based means gives a broader perspective to end user for storing as well as accessing data in an efficient manner. However it throws a big challenge to security and authentication domain. Prior to access the secured data, it is essential to verify the authenticity of the user. Determining the relevancy of the user with respect to the data is foremost agenda of authentication. Most of the advanced systems in different application working with distributed workstations (servers) deployed over different geographic region. The security of user and his/her data becomes more vulnerable in the wireless medium as there is no dedicated link or metod specified over there. We need a foolproof measure against unauthorized access to computer resources and data. The traditional authentication techniques were mostly depended on password based methods. The traditional techniques fail to provide enough protection to the user data. This has prompted the researchers to identify a new area of authentication known as Biometrics, which include finger prints, palm veins; face recognition, DNA, palm print, hand geometry, iris recognition, pattern of human

adfa, p. 1, 2011.

behavior, like- key typing rhythm, ETC. Keystroke dynamics [1, 6] or typing dynamics is a behavioral biometric, refers to the automated method of identifying or confirming the identity of an individual based on the manner and the rhythm of typing on a keyboard. The keystroke techniques are of two type - Keystroke Static authentication (KSA) and Keystroke dynamic authentication (KDA). In Static keystroke based technique, user authentication is done at a particular time instance. The continuous/ dynamic keystroke method is more effective than KSA and it requires the verification process to be continued during the entire session of user interaction. The raw measurements used for keystroke dynamics are dwell time and flight time.

The rest of the paper is organized as follows. In section 2, we review various approaches in keystroke biometrics briefly and analyze their error rates. In Section 3, our proposed approach is described. We give a full detail of the implementations of the approaches in Section 4 and provide experimental results in Section 5. Finally, Section 6 concludes the paper with suggestions for future work.

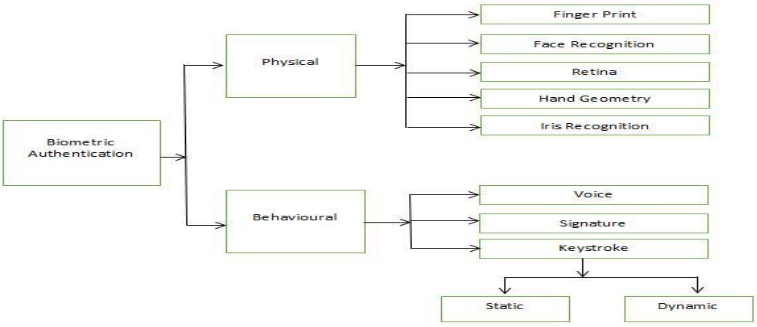

Fig. 1. Classification of Biometric Authentication

2 Literature Review

Most of the existing approaches focus on static verification, where a user types specific pre-enrolled string, e.g., a password during a login process, and then their keystroke features are analyzed for authentication purposes [1]. Pin et al. [2] proposed a solution with EER of 1.401% for strengthening existing password based authentication system by using two layer fusion approach. Using classification techniques based on template matching and Bayesian likelihood models Fabian Monrose [3] achieved accuracy level of 83.22-92.14%. Yu et al. [4] recommended nearest neighbor classifier with the new distance metric in order to identify a legitimate user with respect to a threshold value; this system achieved EER of 8.7%. Kenneth Revett et al. [5] achieved 95% of accuracy in user authentication by inventing software based module where combination of the typing speed and the first and last few characters of the login ID is enough to identify an authenticate user. Wang et al [8] introduced a new user authentication approach by using keystroke dynamic method. This method includes training and authentication. It showed better performance in term of FAR and FRR. Babaeizadeh [10] suggested a KDA based system for verifying a user while requesting for services via CSP in Mobile cloud computing (MCC) environment. The proposed ECC cryptographic algorithm along with keystroke duration attribute was proved to defend 97.33% of efforts for an imposter attack. The data quality, uniqueness and consistency of typing pattern can be improved by using artificial rhythms and tempo cues [11].

3 The Proposed System

Biometric authentication systems usually have two phases for verification purpose- Enrolment Phase and Authentication Phase. In enrolment phase user data is gathered, processed and stored in a database. This becomes a template for future authentication phase. In authentication phase, the user data is acquired and processed. A matching process is there to check the authenticity of the user based on his pre-stored reference templates.

Our fundamental objective is to generate a unique signature for each individual way by analyzing his/her typing behavior. The proposed system will capture user data on a continuous basis and it use the concept of free-text (i.e. no dedicated text to be provided by the user in order to create individual's profile). In brief, the characteristics of the proposed model are:

1. Keystrokes based continuous authentication.
2. Dynamic (all text editor based data collection.
3. . Unique signature vector for each user

The proposed logic has three sub-phases for identifying a user's unique behavior, these are: data collection, Preprocessing of stored data and signature vector generation.

Our proposed system depicted in Figure2 focused on generating a unique typing behavior of each individual.

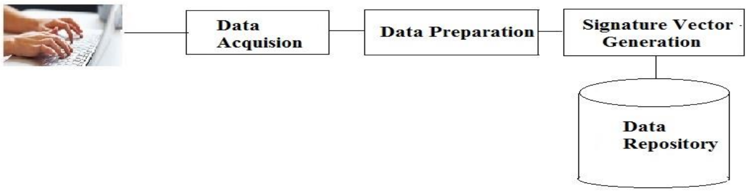

Fig. 2. Block Diagram of the proposed system.

Here is a brief description of each sub-phase:

- **Data Acquisition:** Here raw keystroke data of individuals are collected via various input devices. These may consist of normal computer keyboard, customized pressure sensitive keyboard, virtual keyboard etc. The output of this phase is a text file of an individual's typing behavior with key dwell time and key hold time.

- **Data Preparation: P**re-processing procedures such as feature selection, dimension reduction, and outlier detection [] are to be applied to the collected samples prior to feature extraction to ensure or to increase the quality of feature data. A substantial number of data samples are collected for each individual.
- **Signature Vector Generator**: The output of phase II is used as input in this phase. This file is used to generate a unique signature for each individual by applying some rules on the identified features and store them in database for future classification.

3.1 Data Acquisition

For the purpose of the work we have designed a routine to collect user data (key typing behavior). This routine aims to collect events generated by individuals (operators of computer systems) while using a keyboard. At present, the system works on the MS-Windows platform and does not require any additional libraries. The proposed logic works continuously in background and records a user's activity associated with a keyboard. The events are captured on the fly and saved in text files user character [user_id, vi] in a database. A sample of collected input data is presented below.

Input data collection is carried out for each user user_id separately. We can represent each key event as a vector with 5 tuples. On ith Session the key pressed event represented with vector vi is as follows,

$$v_i = \{Session\_ID, \text{key\_name}_i, \text{hold\_time}_i, \text{dwell\_time}_i, \text{sys\_time}_i\}$$

where, key_namei is the name of the ith key pressed event, naming convention is according to standard QWERTY keyboard interface on the session with Session_ID; hold_timei is the timestamp difference between key pressed and key released; dwell_timei is the timestamp difference between (i-1) th key release and ith key pressed; sys_timei is the system generated time in hour and minute when the event occur.

V is the composite vector $\{v_1, v_2 \ldots v_n\}$; n depends on the overall key press occur on each session on a single day. In practice we restrict the number of sample data collected from the user hence our database is a collection of $SV = \{V_1, V_2 \ldots, V_m\}$ where m is number of sample data collected for each uid.

Additionally we store the total number of BACKSPACE key-press during each session the user interact with his/her machine. The sample collected for each session for the BACKSPACE key can be described with a vector $TB = \{TB_1, TB_2 \ldots TB_l\}$; where L= number of sessions on a single day, andTB_j = {session_j, backspace_count_j};

where backspace_count_j is the total number of times BACKSPACE key is pressed in sessionj. Then we compute the average number of BACKSPACE key-press on a single day and store them into the database with day_id . The average number of the BACKSPACE key-press (AB) on kth day is calculated as follows;

$$AB_k = \frac{1}{L}\sum_{j=1}^{L} TBj,$$ where L is the total number of sessions on kth day.

All AB_k will consititute a vector $A_i = \{uid, day_i, AB_i\}$ describe the average number of BACKSPACE key-press on ith day by the user u_{id}.

Table 1. Key_event_recoder

```
Input: The key pressed (K) from any text editor, like: Word Pad, Note pad, Facebook, etc.
Output: a) A text file key_detail{Session_ID, key pressed, system time of key press, k_hold, k_dwell} b) N=Total number of session on a particular day.
1. Initialize count = 0, k_hold = 0, k_dwell = 0, session_counter = 0;
2. Assume a threshold $TH_D$ in milliseconds for dwell time.
3. For $\forall k \in K$
   a. K_pressed =Time duration (in millisecond) of key pressed.
   b. K_released = Time duration (in millisecond) of key released
   c. Compute k_hold = K_pressed- K_released.
   d. count=count+1; //count total number of key pressed
   e. Compute k_dwell = K_released[i] - K_pressed[i+1];
      i. If k_dwell> $TH_D$ then Session_counter=Session_counter+1;
   f. Continue till closing of all editors.
4. Compute the total number of sessions on each day (N) of interaction
   with the dedicated machine for each individual.
5. Stop.
```

3.2 Data Preparation

In this phase we select unique features for generating individual signature. For this, key_hold_time and key_dwell_time are selected for analysis. We aim to generate a specific range for each key event for these two features.

Our database stored the collected sample in the form of vector $SV = \{V_1, V_2 ..., V_m\}$ where m is total number of session for each u_{id} on a particular day. The preprocessing done on Vi, where Vi = {v1, v2 ... vn}; n=number of key pressed on ith session.

We sort the key pressed event in a session and measure the maximum and minimum holding time of the key event (k). Store the range of key holding time and check for update on next sessions. Finally we get a list for each key_event (k) for Day (d) with specified range for user u_id and store them into database in the form of vector K_H {day_id, key_event_k, $max\_hold\_time_k$, $min\_hold\_time_k$}. $max\_hold\_time_k$ and $min\_hold\_time_k$ defines the range for key holding time for kth key_event on day day_id.

For key_dwell_time feature, we make a pairing between adjacent keys (k, k+1) and store the pair-wise dwell time. In each session, we select the same pairs and list all the dwell_time values. This way, a range for all possible key-pairs is obtained for a day, and stored as vector K_D {day_id, key_pair $_j$, max_dwell_time$_j$, min_dwell_time$_j$, } per user.

Table 2. Table 3. *Key_Hold_Time*

```
Input: The text file containing the key press event with hold time for each user U for a particular day D.
Output: The sample file containing the range for key hold time feature for all possible key on day D.
    1. For each s ∈ Session_Id
        a. Sort all the keys pressed and group the similar keys into $K_{group}$.
        b. For each k ∈ $K_{group}$,
               Find the $k\_MAX_{hold\_time}(s)$ and $k\_MIN_{hold\_time}(s)$ for k.
    2. Check for the re-occurrence of the same key(k) in all sessions of day D.
    3. If $k\_MAX_{hold\_time}(s) >= k\_MAX_{hold\_time}(s+1)$ replace $k\_MAX_{hold\_time}(s)$ with $k\_MAX_{hold\_time}(s+1)$.
       If the $k\_MIN_{hold\_time}(s) <= k\_MIN_{hold\_time}(s+1)$ replace the $k\_MIN_{hold\_time}(s)$ with $k\_MIN_{hold\_time}(s+1)$.
    4. Stop.
```

Table 4. Key_dwell_Time

```
Input: The text file containing the key press event (K) with dwell time ($k_D$) for each user (u_id) for a particular day D.
Output: The sample file containing the time range of key dwelling period for all possible pairs of keys on D day.
    1. Initialize an array K_D [], where K_D[i] = dwell time for a pair of keys.
    2. For ∀ s ∈ Session_Id
        a. For each k ∈ K
            i.  Group $k^{th}$ & $(k+1)^{th}$ into a pair $P_{(k, k+1)}$ and store the respective dwell time ($k_D$) into K_D.
            ii. Search for the re-occurrence of $P_{(k, k+1)}$. Store all the $k_D$ values in K_D.
    3. Combine all the sessions on day D and check for the re-occurrence of the similar key pair.
    4. Find the maximum and minimum $k_D$ value in the K_D array and define it as the range for all pairs of keys.
    5. Stop.
```

In order to generate a template for individual uid we constructed a unique signature vector for each individual. Our feature space has 3 attributes (features); key_hold_time (kh), key_dwell_time (kd) and Backspace_key_count (bkc). For template creation we consider first two features from the feature space.

After the preprocessing of the input data stored in the form of K_H and K_D vector in our repository we proceed to generate a signature vector S_V for each user.

U_V = {u_id, Avg_hold_time, Avg_dwell_time}

Avg_hold_time derived from max_hold_time, min_hold_timek ∈ K_H vector for∀$k \in Key$, Key comprises of all key event possessed by the user for the entire sample collection period. Similarly, max_dwell_time$_{j,}$ min_dwell_time$_{j}$ ∈ K_D used for obtaining Avg_dwell_time for ∀ $k_p \in Key\_Pair$.

Table 5. Signature_vector_generator

```
Input: Hold time range for all key $K_{(Hold)}$ and pair-wise dwell time range for all possible key pair ∈ $K_{(Pair)}$ day *D.*
Output: A signature vector U_V = {*u_id*, (*Avg_hold_time ,key*), (*Avg_dwell_time, key_pair)*} for each user (*u_id).*
  1. Initialize maximum_hold = 0, minimum_hold = 0, max_dwell = 0, min_dwell = 0;
  2. For ∀d∈D, (D= No. of days of collection)
     a. For ∀ k ∈ $K_{(Hold)}$,
        i. Set maximum = $k\_MAX_{hold\_time}$ (d) and minimum= $k\_MIN_{hold\_time}$ (d).
        ii. Update maximum and minimum if $k\_MAX_{hold\_time}$ (d+1) > maximum and $k\_MIN_{hold\_time}$ (d) < minimum respectively.
     b. For ∀ $(l, l+1)$ ∈ $K_{(Pair)}$,
        i. Set max_dwell = $l\_MAX_{dwell\_time}$ (d) and min_dwell= $l\_MIN_{dwell\_time}$ (d).
        ii. Update max_dwell and min_dwell if $l\_MAX_{dwell\_time}$ (d+1) > max_dwell and $l\_MIN_{dwell\_time}$ (d+1) < min_dwell respectively.
  3. Compute the Avg_hold_time from max_dwell and min_dwell value and store in U_V  with corresponding key set pairs ($K_{Pair}$).
  4. Stop.
```

4 Experimental Results

We collected the data-sets from 10 participants for 10 days. The sample data-set collected for each individual shown in Table 4.1 based on session on a day. The users were asked to run our proposed application in background during the entire period of interaction with their dedicated machine. The sample data collected from different machine having different configuration.

The users were not bound to press any dedicated text string and there is no additional interface for capturing data. All the active windows accessed by the users were taken into consideration for generating sample data-set.

The collected samples for each user on a particular day then sorted in alphabetic order of key events. The processed samples depicted in Table 4.2 and Table 4.3 for hold time and dwell time features respectively.

Table 6. Sample collected for an individual on a session.

Session_Id	Key_Name	Hold_Time (sec)	Dwell_Time (sec)	System_Time
1	E	0.140	0.281	23:37
1	X	0.109	0.407	23:37
1	T	0.094	0.125	23:37
1	Space	0.094	0.062	23:37
1	F	0.109	0.266	23:37
1	I	0.062	0.141	23:37
1	M	0.047	0.188	23:37
1	Backspace	0.094	0.312	23:37
1	L	0.016	0.281	23:37

Table 7. Processed data for Key hold time feature per user

Day_ID	Key	Min_Hold_Time	Max_ Hold_Time
1	A	0.031	0.266
1	B	0.079	0.122
1	Backspace	0.031	0.344
1	C	0.078	0.188
1	Comma	0.125	0.125
1	D	0.078	0.156
1	Delete	0.031	0.110
1	Down	0.062	0.125

Table 8. Processed data-set for Key Dwell time feature per user

Day_ID	Key-Pair	Min_Dwell_time	Max_ Dwell_time
1	Space-Backspace	0.0203	0.844
1	D-Space	0.016	0.141
1	E-S	0.156	0.203
1	E-Space	0.031	0.110
1	G-Space	0.032	0.047
1	I-M	0.172	0.188
1	M-E	0.063	0.172
1	N-D	0.094	0.125

1	O-N	0.218	0.297

Table 9. Signature vector set for a particular user in the enrolled data-set.

U_ID	Avg_hold_time	Key	Avg_dwell_time	Key-pair
U1	0.1485	A	0.4321	Space-Backspace
U1	0.1005	B	0.0785	D-Space
U1	0.1875	Backspace	0.1795	E-S
U1	0.133	C	0.0705	E-Space
U1	0.125	Comma	0.0395	G-Space
U1	0.117	D	0.18	I-M
U1	0.0705	Delete	0.1175	M-E
U1	0.0935	Down	0.1095	N-D

We differentiate the user behavior based on the two unique feature discussed so far, i.e., hold time and dwell time. [Figure 4.2, 4.3] illustrate the comparative analysis of two user *USER1 and USER2* depending on key Hold time and key dwell time feature.

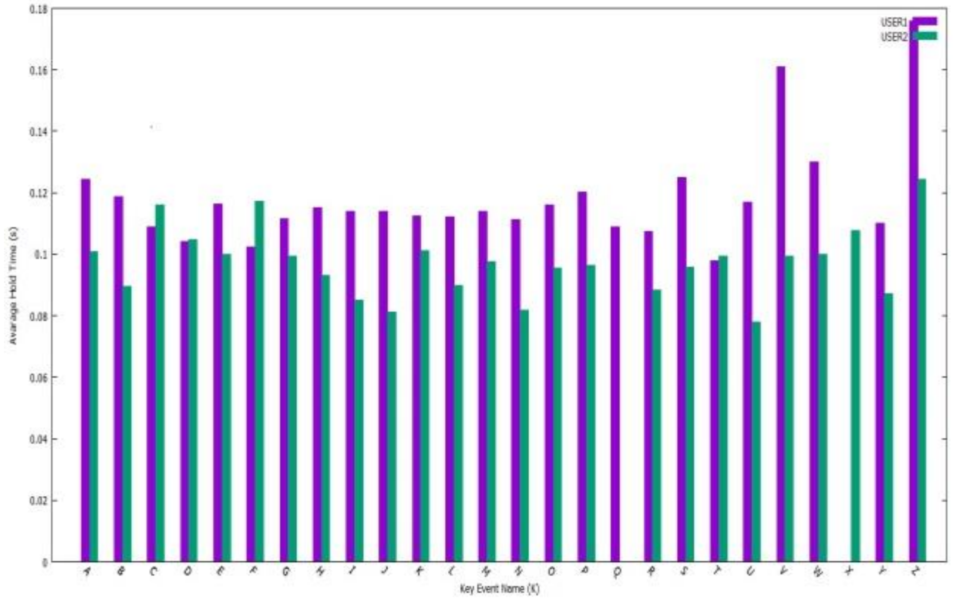

Fig. 3. Average key holding time for all possible keys

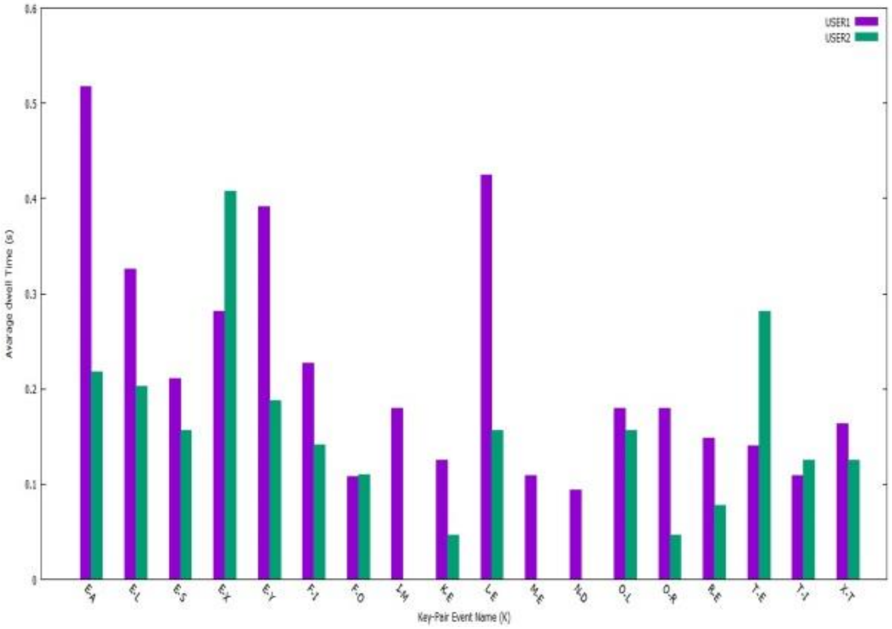

Fig. 4. Average Dwell time for a key pair

5 Conclusion

We have observed that the prevalent biometrics based techniques for identification of a legitimate user often suffered from high FAR and FPR rates, which had a negative effect on the respective accuracy rate. The study reveals a fact that most of the developed applications consider a dedicated text (mainly passwords of specific format) to be typed by the user. However, the fixed text examples failed to capture significant variations in individual typing due to limited characters used. In this paper, we have used free-text concept to solve this issue. The software for collecting user data is designed to be machine independent, and samples are collected from a varying set of computers. Our proposed signature vectors deal with all possible key events so that the aggregated behavior of the end user is stored in to the repository. Our future work will concentrate on the classification verification part of the individual based on these store templates.

6 References

1. D. Umpires and G. Williams, "Identity Verification through Keyboard Characteristics", *Int'l J. Man-Machine Studies, vol. 23, No. 3, 263–273. (1985)*
2. *Pin Sheen The, Andrew BEng Jin Theo, Connie Tee, Thin Song On "Keystroke dynamics in password authentication enhancement", Expert Systems with Applications: An International Journal, Volume 37, 8618-8627 (2010)*
3. *Fabien Monroe, Ariel D. Rubin, Keystroke dynamics as a biometric for authentication, in Elsevier- Future Generation Computer Systems, volume 16, no. 4, 351–359 (2000)*
4. *Yu Hong Yuban Deng Anil K. Jai, "Keystroke Dynamics for User Authentication", in Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Computer Society Conference, 117 – 123 (2012)*
5. *Kenneth Revert, Sergio Terrier de Magadha's and Henrique Santos, "Data Mining a Keystroke Dynamics Based Biometrics Database Using Rough Sets", University of Technology of Compiegne (2005).*
6. *M. Karnana, M. Akilab, N. Krishnarajc, "Biometric personal authentication using keystroke dynamics: A review", Applied Soft Computing, Volume 11, no. 2, 1565–1573. (2011)*
7. *Xu. Wang, G. Fangxia, and M. Jian-feng, "User authentication via keystroke dynamics based on difference subspace and slope correlation degree", Digital Signal Processing, Volume 22, no. 5, pp. 707-712. (2012)*
8. *Mahnoush Babaeizadeh, Majid Bakhtiari, and Mohd Aizaini Maarof, "Authentication Method through Keystrokes Measurement of Mobile users in Cloud Environment", Vol. 6, No. 3. (2014)*
9. *Seong-seob Hwang, Hyoung-joo Lee, Sungzoon Cho, "Improving authentication accuracy using artificial rhythms and cues for keystroke dynamics-based authentication" in Expert Systems with Applications in ELSEVIER, Volume 36, Issue 7, Pages 10649–10656. (2009)*
10. *K. Kotani and K. Horii, "Evaluation on a keystroke authentication system by keying force incorporated with temporal characteristics of keystroke dynamics", Behaviour and Information Technology, vol. 24, no. 4, pages 289–302. (2005)*
11. *S. S. Hwang, S. Cho, and S. Park, "Keystroke dynamics-based authentication for mobile devices", Computers and Security, vol. 28, no. 1-2, pages. 85–93. (2009)*

12. *M. Nauman, T. Ali, and A. Rauf, "Using trusted computing for privacy preserving keystroke-based authentication in smartphones", Telecommunication Systems, vol. 52, no. 4, pages. 2149–2161. (2011)*
13. *Y. Kaneko, Y. Kinpara, and Y. Shiomi, "A hamming distance-like filtering in keystroke dynamics," in Proceedings of the 9th Annual International Conference on Privacy, Security and Trust (PST '11), pages. 93–95, (2011).*
14. *W. Lee, S.-S. Choi, and B.-R. Moon, "An evolutionary keystroke authentication based on ellipsoidal hypothesis space," in Proceedings of the 9th annual conference on Genetic and evolutionary computation, Pages 2090-2097. (2007)*
15. *D. Hosseinzadeh and S. Krishnan, "Gaussian mixture modelling of keystroke patterns for biometric applications," in Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions, Vol.:38, pages 816-826. (2011)*