International Journal of Soft Computing 7 (4): 169-180, 2012
ISSN: 1816-9503

Keystroke Dynamics: A Met Heuristic Approach

[1]R.P. Antokumar, [2]M. Akila and [3]M. Karnan
[1]Department of Computer Science and Engineering, Bharathiar University,
Coimbatore, Tamil Nadu, India
[2]Department of Information Technology, Vivekanandha College of Engineering for Women,
Elayampalayam, Namakkal, Tamil Nadu, India
[3]Department of Computer Science and Engineering, Tamilnadu College of Engineering,
Coimbatore, Tamil Nadu, India

Abstract: The need to secure sensitive data and computer systems from intruders while allowing ease of access for authenticates users is one of the main problems in computer security. Traditionally, passwords have been the usual method for controlling access to computer systems but this approach has many inherent flaws. Keystroke dynamics is a relatively new method of biometric identification and provides a comparatively inexpensive and low profile method of hardening the normal login and password process. This study presents the feature subset selection in Keystroke dynamics for identity verification and it reports the results of experimenting Ant Colony Optimization (ACO) technique on keystroke duration, latency and digraph for feature subset selection. Here, the Ant Colony Optimization is used to reduce the redundant feature values and minimize the search space. Optimum feature subset is obtained using keystroke duration values when compared with the other two feature values.

Key words: Feature extraction, feature subset selection, mean and standard deviation, Ant Colony Optimization Algorithm (ACO)

INTRODUCTION

Access to computer systems is usually controlled by user accounts with usernames and passwords. Such scheme has little security (Hu *et al*., 2008; Pavaday and Soyjaudah, 2007) if the information falls to wrong hands. Key cards or Biometric Systems (Kapczynski *et al*., 2006; Boechat *et al*., 2007; Jain *et al*., 2003; Blackburn *et al*., 2007) for example fingerprints (Hong and Jain, 1998) is being used now-a-days to improve the security.

Biometric Methods measure biological and physiological characteristics to uniquely identify individuals. The main drawback of most biometric methods is that they are expensive to implement because most of them require specialized hardware to strengthen security. But they require quite expensive additional hardware. On the other hand keystroke dynamics (Monrose and Rubin, 2000) consist of many advantages like:

- It can be used without any additional hardware
- Inexpensive
- Hardening the security

Keystroke dynamics has many applications in the computer security arena. One area where the use of a static approach to keystroke dynamics may be particularly interesting is in restricting source level access to the master server hosting a Kerberos (Azevedo *et al*., 2007) key database. Any user accessing the server is prompted to type a few words or a pass phrase in conjunction with his/her username and password. Access is granted if his/her typing pattern matches within a reasonable threshold of the claimed identity. This safeguard is effective as there is usually no remote access allowed to the server and the only entry point is via console login. Alternatively, dynamic or continuous monitoring of the interaction of users while accessing highly restricted documents or executing tasks in environments where the user must be alert at all times (for example air traffic control) is a ideal scenario for the application of a keystroke authentication system. Keystroke dynamics may be used to detect uncharacteristic typing rhythm (brought on by drowsiness, fatigue, etc.) in the user and notify third parties. Keystroke dynamics include several different measurements (Teh *et al*., 2007; Shepherd, 1995) such as:

Corresponding Author: M. Karnan, Department of Computer Science and Engineering, Tamilnadu College of Engineering, Coimbatore, Tamil Nadu, India

- Duration of a keystroke or key hold time
- Latency of keystrokes or inter-keystroke times
- Typing error
- Force keystrokes, etc.

Keystroke analysis (Leberknight *et al*., 2008) is of two kinds Static and Dynamic. Static keystroke analysis essentially means that the analysis is performed on typing samples produced using the same predetermined text for all the individuals under observation. Dynamic keystroke analysis implies a continuous or periodic monitoring of issued keystrokes and is intended to be performed during a log in session, after the authentication phase has passed. There are two phases namely Extraction phase and Verification phase as in Fig. 1. During the feature extraction phase (Boechat *et al*., 2007; Leberknight *et al*., 2008; Gaines *et al*., 1980; Young and Hammon, 1989) user keystroke features from one's name or password are captured, processed and stored in a reference file as prototypes for future use by system in subsequent authentication operations. During the verification phase (Bleha and Obaidat, 1991; Daw-Tung, 1997) user keystroke features are captured, processed in order to render an authentication decision based on the outcome of a classification process of the newly presented feature to the pre-stored prototypes (reference templates) (Hocquet *et al*., 2005; Yu and Cho, 2004). It would be necessary for the user to type his/her name or password a number of times in order for the system to be able to extract the relevant features that uniquely represent the user. However, the task of typing one's name or password over and over is both tiring and tedious in the feature extraction phase which could lead users to alter their normal typing pattern. Thus, most systems based on biometrics are required to researchers with a summarized set of information from which to extract knowledge. In order to reduce this problem, researchers could eliminate some features of the original dataset, selecting only the best ones in terms of class cohesion.

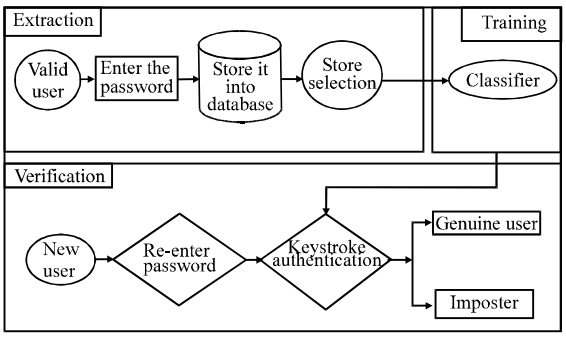

Fig. 1: Keystroke dynamics analysis framework

Feature subset selection (Yang and Honavar, 1998; John and Kohavi, 1994) is applied to high dimensional data prior to classification. Feature subset selection is essentially an optimization problem which involves searching the space of possible features to identify one that is optimum or near-optimal with respect to certain performance measures since, the aim is to obtain any subset that minimizes a particular measure (classification error for instance) (Shiv Subramaniam *et al*., 2007; Singhi and Liu, 2006). In order to reduce the complexity and to increase the performance of the classifier the redundant and irrelevant features are reduced from the original feature set. Many feature subset selection (Karnan *et al*., 2006) approaches is proposed in the previous studies.

Related work: Yu and Cho (2003, 2004) propose a GA-SVM based wrapper approach for feature subset selection in which GA is employed to implement a randomized search and SVM, an excellent novelty detector with fast learning speed is employed as a base learner. The degree of diversity and quality are guaranteed and thus they gave result in an improved model performance and stability. Sung and Cho (2006) propose one step approach similar to that of Genetic Ensemble Feature Selection (GEFS) yet with a more direct diversity term in the fitness function and SVM as base classifier and similar to that of Yu and Cho (2003) yet with a diversity term and no more post processing step. In particular, so called uniqueness term is used in a fitness function, measuring how unique each classifier is from others in terms of the features used. To adapt SVM researchers use Gaussian kernel. GA was used to filter the data and to carry out a selection of characteristics. It reports an average FAR of 15.78% with minimum FAR of 5.3% and maximum FAR of 20.38% for raw data with noise. Azevedo *et al*. (2007) and Azevedo and Cavalcanti (2007) designed a hybrid system based on Support Vector Machines (SVM) and Stochastic Optimization Techniques. Standard Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) variation was used and produced a good result for the tasks of feature selection. Standard GA and PSO variation was used and produced a good result for the tasks of feature selection and personal identification with an FAR of 0.81% and IPR of 0.76% (Boechat *et al*., 2007) used weighted probability measure by selecting N features of the features vector with the minors of standard deviation, eliminating the features less significant. They obtained optimum result using 90% of the features with 3.83% FRR and 0% FAR.

FEATURE EXTRACTION

To capture a keystroke, it would be necessary for users to type their password number of times. The system

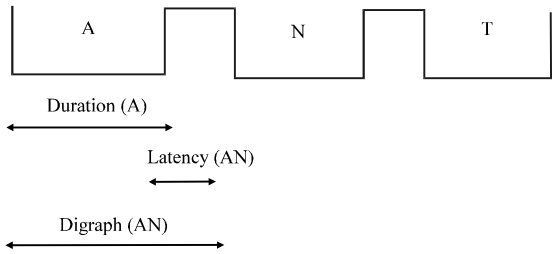

Fig. 2: Measurement of duration, latency and duration

would set about capturing these features using three methods regarding the time (msec) that a particular user maintains the key pressed (duration time) and the time elapsed between releasing one key and pressing the next (latency time) and the combination of the above is called digraph. The data was collected from 11 participants with different passwords. Each participant was asked to type his/her password 10 times. The mean and standard deviation values were measured as shown in Table 1.

The raw data file recorded by the application measures the duration, latency and digraph timing information in microseconds (μsec) for the entered password. Upon pressing submit, a raw-data text file is generated. During the creation of the raw data file, the mean (μ) and standard deviation (σ) (Monrose *et al*., 1999; Bergadan *et al*., 2002; Karnan *et al*., 2006) of each feature (i) of the pattern set (x) are calculated for N samples in agreement with the following equations:

$$\text{Mean } (\mu_i) = (1/N)\sum_{i=1}^{N} x\,[i] \quad (1)$$

$$\text{Standard deviation } (\sigma_i) = (1/N\text{-}1)\sum_{i=1}^{N} \left|x\,[i]\text{-}\mu\,[i]\right| \quad (2)$$

For instance, for the password ANT the timing information for duration is 205, 250, 235 msec. Figure 2 shows the measurement of duration, latency and digraph of keystrokes of the password ANT.

FEATURE SUBSET SELECTIONS

Several samples are typed by user for n number of times. During the verification phase, it takes more time to verify all the n number of features. Reducing this time complexity researchers are using feature subset selection methods. In feature subset selection (Yu and Cho, 2004, 2003) method, researchers extract the optimized features from the n number of features. It is essentially an optimization problem which involves searching the space of possible features to identify one that is optimum. Various ways to perform feature subset selection has been studied earlier. Here, researchers propose Ant Colony Optimization to select the feature subset.

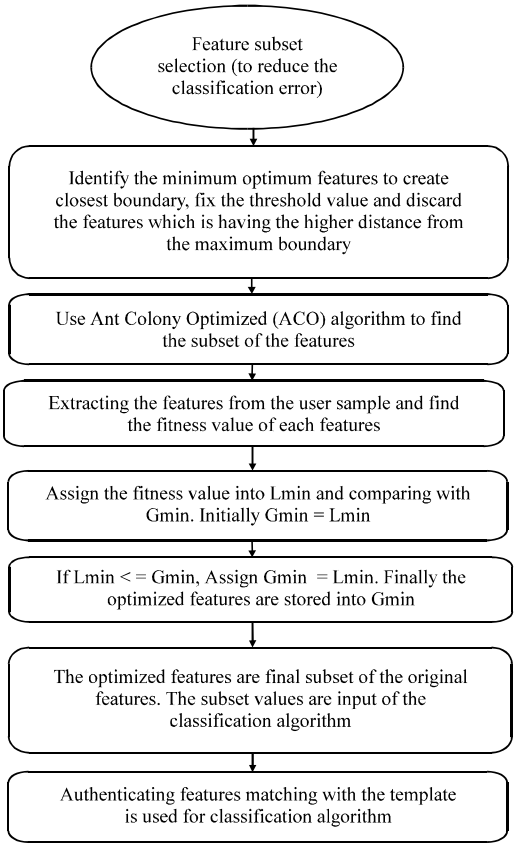

Fig. 3: Ant Colony Optimization for subset selection

Ant Colony Optimization (ACO): Ant algorithms (Dorigo *et al*., 1991; Dorigo and Gambardella, 1997) was first proposed by Dorigo and colleagues (Boechat *et al*., 2007) as a multi-agent approach to difficult combinatorial optimization problems such as the Traveling Salesman Problem (TSP) and the Quadratic Assignment Problem (QAP). There are currently various activities in the scientific community to extend and apply ant-based algorithms to many different discrete optimization problems. The ACO heuristic (Dorigo *et al*., 1991; Li and Xu, 2003) has been inspired by the observation on real ant colony's foraging behavior and on that ants can often find the shortest path between food source and their nest. Ant individuals transmitted information through the volatile chemical substances which ants left in his passing path and also known as the pheromone and then reach the purpose of finding the best way to search food sources. An ant encountering a previously laid trail can detect the dense of pheromone trail. It decides with high probability to follow a shortest path and reinforce that trail with its own pheromone. The large amount of pheromone is on the particular path, the large probability is that an ant selects that path and the paths pheromone trail will become denser (Fig. 3).

At last, the ant colony collectively marks the shortest path which has the largest pheromone amount. Such simple indirect communication way among ants embodies actually a kind of collective leaning mechanism. The Ant Colony Optimization (Fig. 4) algorithm is as follows:

ACO calculation: Mean and standard deviation were measured for duration, latency and digraph for each sample. Ant Colony algorithm is used for selecting the optimum feature for each participant and the selected features are considered for classification. For instance, for the password ANT the calculated values using duration, latency and standard deviation as features and the result after applying ACO algorithm to it is displayed from Table 1-9. Considering the following Duration (D), Latency (L) and Digraph (Di) values from user's keystroke

Step 1: Get the feature values a[x] from duration/latency/digraph of keystrokes.
Step 2: Calculate the fitness function f[x] by the following equation for every a[x]

$$f[x] = 1/(1+a[x])$$

Step 3: Initialize the following

NI	=	5	(Number of iterations)
NA	=	2	(Number of Ants)
T_0	=	0.001	(Initial pheromone value for every a[x])
ρ	=	0.9	(Rate of pheromone evaporation parameter for every a[x])

Step 4: Store the fitness function values in S where S = {F [x], T_0, flag} where flag column mentions whether the feature isselected by the ant or not.
Step 5: The following is repeated for NI times
1: A random feature value g [x] in a [x] is selected for each ant with the criteria that the particular feature value should not have been selected previously.
2: Selected feature value`s, pheromone value is updated by the following:

$$T_{new} \leftarrow (1-\rho) * T_{old} + \rho * T_0 \text{ for g [x],}$$

where T_{new} and T_{old} are the new and old pheromone value of the feature value
3: Obtain Lmin min (g [x]) where Lmin is the local minimum
4: Check if Lmin Gmin then assign Gmin = Lmin
5: Else no change in Gmin value where Gmin is the Global minimum.
6: Select the best feature g [y] whose solution is equal to the Local minimum value at the end of the last iteration.
7: The selected g [y]'s pheromone value is globally updated by the following equation:

$$T_{new}7 \leftarrow (1- \alpha) * T_{old} + \alpha * \Delta T_{old}, \text{ for g[y]}$$

where α is a rate of pheromone evaporation parameter
8. Finally, the Gmin value is stored as optimum value

Fig. 4: The Ant Colony Optimization Algorithm

Table 1: Before feature subset selection using duration for mean and standard deviation

		Fitness value		Local minimum		Pheromone update		Global minimum	
Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
230	12.9	0.00432	0.76394	0.00414	0.04670	0.06100	0.06100	0.00414	0.046707
240	20.4	0.00414	0.04670	0.00414	0.03286	0.06759	0.06759	0.00414	0.032862
170	29.4	0.00584	0.03286	0.00523	0.03286	0.06759	0.06759	0.00414	0.032862
190	8.16	0.00523	0.10917	0.00438	0.03096	0.06759	0.06759	0.00414	0.030960
227	31.3	0.00438	0.03096	0.00369	0.03041	0.06759	0.06759	0.00369	0.030414
270	31.8	0.00369	0.03041	0.00369	0.02144	0.06759	0.06759	0.00369	0.021441
195	45.6	0.00510	0.02144	0.00398	0.02144	0.06759	0.06759	0.00369	0.021441
250	16.3	0.00398	0.05773	0.00398	0.05773	0.06759	0.06759	0.00369	0.021441
220	4.08	0.00452	0.19681	0.00452	0.12391	0.06759	0.06759	0.00369	0.021441
200	7.07	0.00497	0.12391	0.00432	0.07639	0.06759	0.06759	0.00369	0.021441

Table 2: After feature subset selection using duration-mean

		Fitness value		Local minimum		Pheromone update		Global minimum	
Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
270	31.8	0.00369	0.03041	0.00369	0.02144	0.06759	0.06759	0.00369	0.021441
250	16.3	0.00398	0.05773	0.00398	0.05773	0.06759	0.06759	0.00369	0.021441
240	20.4	0.00414	0.04670	0.00414	0.03286	0.06759	0.06759	0.00414	0.032862
230	12.9	0.00432	0.76394	0.00414	0.04670	0.06100	0.06100	0.00414	0.046707
220	4.08	0.00452	0.19681	0.00452	0.12391	0.06759	0.06759	0.00369	0.021441

Table 3: After feature subset selection using duration standard deviation

		Fitness value		Local minimum		Pheromone update		Global minimum	
Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
195	45.6	0.00510	0.02144	0.00398	0.02144	0.06759	0.06759	0.00369	0.021441
270	31.8	0.00369	0.03041	0.00369	0.02144	0.06759	0.06759	0.00369	0.021441
227	31.3	0.00438	0.03096	0.00369	0.03041	0.06759	0.06759	0.00369	0.030414
170	29.4	0.00584	0.03286	0.00523	0.03286	0.06759	0.06759	0.00414	0.032862
240	20.4	0.00414	0.04670	0.00414	0.03286	0.06759	0.06759	0.00414	0.032862

Table 4: Before Feature subset selection using Latency for mean and standard deviation

		Fitness value		Local minimum		Pheromone update		Global minimum	
Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
250	30.00	0.00398	0.03225	0.00398	0.03225	0.06100	0.06100	0.00398	0.03225
230	5.00	0.00433	0.16666	0.00432	0.09091	0.06759	0.06759	0.00398	0.03225
200	10.00	0.00497	0.09091	0.00362	0.01176	0.06759	0.06759	0.00369	0.01176
275	84.03	0.00362	0.01176	0.00362	0.01176	0.06759	0.06759	0.00362	0.01176
220	20.00	0.00452	0.04762	0.00452	0.04761	0.06759	0.06759	0.00362	0.01176
195	15.00	0.00510	0.06250	0.00473	0.06250	0.06759	0.06759	0.00362	0.01176
210	10.00	0.00473	0.09091	0.00452	0.09091	0.06759	0.06759	0.00362	0.01176
220	10.00	0.00452	0.09091	0.00369	0.02439	0.06759	0.06759	0.00362	0.01176
270	40.00	0.00369	0.02439	0.00369	0.02439	0.06759	0.06759	0.00362	0.01176

Table 5: Feature subset selection using latency-mean

		Fitness value		Local minimum		Pheromone update		Global minimum	
Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
275	84.03	0.00362	0.01176	0.00362	0.01176	0.06759	0.06759	0.00362	0.01176
270	40.00	0.00369	0.02439	0.00369	0.02439	0.06759	0.06759	0.00362	0.01176
250	30.00	0.00398	0.03225	0.00398	0.03225	0.06100	0.06100	0.00398	0.03225
230	5.00	0.00433	0.16666	0.00432	0.09091	0.06759	0.06759	0.00398	0.03225
220	10.00	0.00452	0.09091	0.00369	0.02439	0.06759	0.06759	0.00362	0.01176

Table 6: Feature subset selection using latency-standard deviation

		Fitness value		Local minimum		Pheromone update		Global minimum	
Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
275	84.03	0.00362	0.01176	0.00362	0.01176	0.06759	0.06759	0.00362	0.01176
270	40.00	0.00369	0.02439	0.00369	0.02439	0.06759	0.06759	0.00362	0.01176
250	30.00	0.00398	0.03225	0.00398	0.03225	0.06100	0.06100	0.00398	0.03225
220	20.00	0.00452	0.04762	0.00452	0.04761	0.06759	0.06759	0.00362	0.01176
220	10.00	0.00452	0.09091	0.00369	0.02439	0.06759	0.06759	0.00362	0.01176

Table 7: Before feature subset selection using digraph for mean and standard deviation

		Fitness value		Local minimum		Pheromone update		Global minimum	
Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
240	12.200	0.00831	0.07552	0.00831	0.07552	0.06100	0.06100	0.00831	0.07550
235	5.470	0.00847	0.15456	0.00847	0.04708	0.06090	0.06759	0.00831	0.04708
185	20.200	0.01082	0.04708	0.00885	0.04708	0.06090	0.06759	0.00831	0.04708
232.5	17.700	0.00885	0.05336	0.00885	0.04739	0.06090	0.06759	0.00831	0.04708
223.5	20.100	0.00891	0.04739	0.00879	0.04739	0.06090	0.06759	0.00831	0.04708
232.5	12.600	0.00879	0.07331	0.00879	0.07331	0.06090	0.06759	0.00831	0.04708
202.5	7.740	0.00984	0.11441	0.00860	0.07127	0.06090	0.06759	0.00831	0.04708
235	13.000	0.00850	0.07127	0.00821	0.75458	0.06090	0.06759	0.00821	0.04708
235	17.300	0.00821	0.05458	0.00497	0.75458	0.06090	0.06759	0.00497	0.04708
200	10.488	0.00497	0.08704	0.00497	0.07552	0.06090	0.06759	0.00497	0.04708

Table 8: Feature subset selection using digraph-mean

		Fitness value		Local minimum		Pheromone update		Global minimum	
Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
200	10.488	0.00497	0.08704	0.00497	0.07552	0.06090	0.06759	0.00497	0.04708
235	17.300	0.00821	0.05458	0.00497	0.75458	0.06090	0.06759	0.00497	0.04708
240	12.200	0.00831	0.07552	0.00831	0.07552	0.06100	0.06100	0.00831	0.07550
235	5.470	0.00847	0.15456	0.00847	0.04708	0.06090	0.06759	0.00831	0.04708
235	13.000	0.00850	0.07127	0.00821	0.75458	0.06090	0.06759	0.00821	0.04708

Table 9: Feature subset selection using digraph-standard deviation

		Fitness value		Local minimum		Pheromone update		Global minimum	
Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
185	20.2	0.01082	0.04708	0.00885	0.04708	0.06090	0.06759	0.00831	0.04708
240	12.2	0.00831	0.07552	0.00831	0.07552	0.06100	0.06100	0.00831	0.07550
223.5	20.1	0.00891	0.04739	0.00879	0.04739	0.06090	0.06759	0.00831	0.04708
235	17.3	0.00821	0.05458	0.00497	0.75458	0.06090	0.06759	0.00497	0.04708
232.5	17.7	0.00885	0.05336	0.00885	0.04739	0.06090	0.06759	0.00831	0.04708

profile: D ((A, 240), (N, 215), (T, 235)), L ((AN, 275), (NT, 235)) and Di ((A, 515), (N, 470), (T, 235). The mean of duration, latency and digraph are computed as shown in Table 1-9. Computation of duration for each letter of the password ANT and calculation of mean, fitness value, local minimum, pheromone update and global minimum is as follows: for instance if the duration = {A-240, N-215, T-235}.

Calculation of fitness value for duration:

$$\text{Mean } (\mu_i) = (1/N) \sum_{i=1}^{N} x\,[i]$$
$$= 240 + 215 + 235/3$$

Therefore:

$$\mu_i = 230 = a\,[x]$$
$$\text{Fitness value } f\,[x] = 1/1 + a\,[x]$$
$$f\,[x] = 1/1 + 230$$
$$= 0.00432$$
$$\text{Standard Deviation } (\sigma_i) = (1/N\text{-}1) \sum_{i=1}^{N} |x\,[i] - \mu\,[i]|$$
$$= \sqrt{(230\text{-}240)^2 + (230\text{-}215)^2 + (230\text{-}235)^2/2}$$
$$(\sigma_i) = 12.9 = a\,[x]$$
$$\text{Fitness value } f\,[x] = 1/1 + 12.9$$
$$= 0.76394$$

Calculation of local minimum for duration: Initially the fitness value a [x] is directly assigned to Local minimum (Lmin) for the first value (i.e., a [1]). Then, the next fitness value a [x] (i.e., a [2]) is compared with the previous value already calculated. The minimum is found in them and is replaced with the Local minimum value. For example:

$$\text{For mean value } a\,[1] = 0.00432$$
$$\text{Assign } a\,[1] = \text{Lmin}$$
$$\text{Lmin} = 0.00432$$
$$\text{After finding the next value } a\,[2] = 0.00414$$

Check whether a [1] less then or equal to the value a [2]. If the condition is true, assign Lmin = a [1]. Otherwise, Lmin = a [2]; here, in this example if (0.00432<= 0.00414). Lmin = 0.00414. For standard deviation, similarly:

$$a\,[1] = 0.76394$$
$$\text{Assign } a\,[1] = \text{Lmin}$$
$$\text{Lmin} = 0.76394$$
$$\text{The second value } a\,[2] = 0.04670$$

Check the condition, a [1] less then or equal to the value a [2]. Here, in this example, if (0.76394<0.04670); Lmin = 0.04670.

Calculation of global minimum for duration: Global minimum (Gmin) is assigned Lmin value initially (i.e., Lmin = Gmin). Next the value in the Gmin is compared with Lmin, to find the minimum amongst them. For example, for mean, Lmin = 0.00414; initially, Gmin = Lmin, i.e., Gmin = 0.00414; for next feature value, condition should be satisfied for Gmin, i.e. (Gmin <= Lmin).

Next, Lmin consist of next minimum value. According to this example next minimum value is same value. This value is compared with Gmin if the value is less then or equal to the Lmin then, Gmin = 0.00414. For Standard Deviation, Lmin = 0.04670; initially, Gmin = Lmin; Gmin = 0.04670; Next Lmin = 0.03286; compare with Gmin values, i.e. (Gmin <= Lmin). Finally, Gmin = 0.03286.

Calculation pheromone update for duration:

$$T_{new} \leftarrow (1\text{-}\rho) \times T_{old} + \rho \times T_0 + c$$

Where:

T_{new} = New pheromone rate
T_{old} = Old pheromone rate
T_0 = Initial pheromone value and
c = Constant value

Initially, $T_{old} = 0.001$, $T_0 = 0.001$, c = 0.06. For example for mean, first local pheromone is updated as:

$$T_{new} = (1\text{-}0.9) \times 0.001 + 0.9 \times 0.001 + 0.06$$
$$= 0.1 \times 0.001 + 0.0009 + 0.06$$
$$= 0.06100$$

T_{old} value change due to the previous T_{new} value, i.e., T_{old} = 0.06100. For standard deviation, local pheromone updated as:

$$T_{new} = (1\text{-}0.9) \times 0.001 + 0.9 \times 0.001 + 0.06$$
$$= 0.1 \times 0.001 + 0.0009 + 0.06$$
$$= 0.06100$$

Similarly, the values for the latency and digraph are calculated as earlier. Mean and standard deviation is used to extract the features from the Keystroke duration, latency and digraph. From the extracted features the optimized features are selected using the Ant Colony Optimization Algorithm to reduce the searching space.

CLASSIFICATION USING BACKPROPAGATION NEURAL NETWORK

Neural networks are simplified models of the biological nervous system which is a computing, performed like a human brain. A Neural network has a

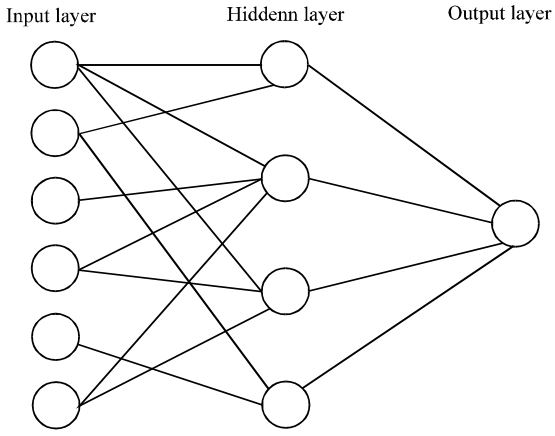

Fig. 5: Neural network architecture

parallel distributed architecture with a large number of nodes and connections as shown in Fig. 5. Each connection points from one node to another and associated with weights. The backpropagation neural network is a network of simple processing elements working together to produce a complex output. The back propagation paradigm (Obaidat and Macchairolo, 1991) has been tested in various applications such as bond rating, mortgage application evaluation, protein structure determination, signal processing and handwritten digit recognition (Rumelhart *et al*., 1987; Hecht-Nielsen, 1990; Obaidat and Macchairolo, 1991). It can learn difficult patterns such as those found in typing style and can recognize these patterns even if they are variations of the ones it initially learned.

The backpropagation neural network uses a training set composed of input vectors and a desired output (the desired output is usually a vector instead of a single value). The backpropagation neural network has many processing element. These elements or nodes are arranged into layers: input, middle and output. The output from a backpropagation neural network is computed using a procedure known as the forward pass:

- The input layers propagate a particular input value component to each node in the hidden layer
- Hidden layers compute output values which become inputs to the output layer
- The output layers compute the network output for the particular input values

The forward pass produces an output vector for a given input vector based on the current state of the network weights. Since, the network weights are initialized to random values, it is unlikely that reasonable outputs will result before training. The weights are adjusted to reduce the error by propagating the output error backwards through the network. This process is where the backpropagation neural network gets its name and is known as the backward pass:

- Compute error values from the output layer. This can be computed because the desired output is known
- Compute the error for the middle layer nodes. This is done by attributing a portion of the error at each output layer node to the middle layer nodes which feed that output node. The amount of error due to each middle layer node depends on the size of the weight assigned to the connection between the two nodes
- Adjust the weight values to improve network performance
- Compute overall error to test network performance

The training set is repeatedly presented to the network and the weight values are adjusted until the overall error is below a predetermined tolerance. The Back Propagation Algorithm can be implemented in two different modes: online mode and batch mode. In the online mode the error function is calculated after the presentation of each input timing vector and the error signal is propagated back through the network, modifying the weights before the presentation of the next timing vector. This error function is usually the Mean Square Error (MSE) of the difference between the desired and the actual responses of the network over all the output units. Then, the new weights remain fixed and a new timing vector is presented to the network and this process continues until all the timing vectors have been presented to the network. The presentation of all the timing vectors is usually called one epoch or a single iteration. In practice many epochs are needed before the error becomes acceptably small. In the batch mode the error signal is calculated for each input timing vector and the weights are modified every time the input timing vector is been presented. Then, the error function is calculated as the sum of the individual MSE for each timing vector and the weights are accordingly modified (all in a single step for all the timing vectors) before the next iteration. In the forward pass outputs are computed and in the backward pass weights are updated or corrected based on the errors. The development of the Back Propagation Algorithm is a landmark in neural networks in that it provides a computationally efficient method for the training of multi-layer perceptrons.

Backpropagation neural network learning rule: Initially, the inputs and outputs of the feature subset selection

algorithms are normalized with respect to their maximum values as shown in Fig. 2. In neural networks (Haider *et al*., 2000; Brown and Rogers, 1993) it lies between zeros to one. These normalized values are assigned to the input neurons. The number of hidden neurons is greater then or equal to the number of input neurons. And there is only one output neuron. Initial weights are assigned randomly. The output from the each hidden neuron is calculated using the sigmoid function:

$$S_1 = 1/(1 + e^{-\lambda x})$$

Where:

λ = 1

x = $\Sigma_i w_{ih} I_i$, where w_{ih} is the weight assigned between input and hidden layer and I is the input valu

The output from the output layer is calculated using the sigmoid function:

$$S_2 = 1/(1 + e^{-\lambda x})$$

Where:

λ = 1

x = $\Sigma_i w_{ho} O_h$, where w_{ho} is the weight assigned between hidden and output layer and O_h is the output value from hidden neurons

S_2 = Subtracted from the desired output

Using this error (e) value, the updation of weight is performed as:

$$Delta = e \times S_2 \times (1-S_2)$$

The weights assigned between input and hidden layer and hidden and output layer are updated as:

$$W_{ho} = W_{ho} + (n \times delta \times S_1)$$

$$W_{ih} = W_{ih} + (n \times delta \times I_i)$$

Where:

n = The learning rate

I = The input values

Again calculate the output from hidden and output neurons. Then, check the error (e) value and update the weights. After several iterations when the difference between the calculated output and the desired output is less then the threshold value, the iteration is stopped (Fig. 6).

Experimental results: Assign the subset selection values as input to the back propagation neural network. For, e.g., if the subset values are 270, 250, 240, 230, 220 msec. Let I_i be input of input, O_i be output of input, I_h be input of hidden, O_h be output of hidden, I_o be Input of output, O_i be output of output. After applying the BPNN learning the following calculations are done.

Forward pass

Input of input: Input of input and output of input layers is f (x); Input f (x) = (0.00369, 0.00398, 0.00414, 0.00432, 0.00452).

Input of hidden: Assign weights randomly between input to hidden layers as:

W_{ih} = (-0.7696789, -0.547986, -0.07547, 0.427683, 0.907837...)

Assume five weights for each single node. Multiply each output of input into weight that assigned randomly. Such as:

Step 1: ACO feature subset selection algorithm values are considered as input.

Step 2: These feature values are normalized between 0 and 1 and assigned to input neurons

Step 3: Represents the weights to the link of input nodes to hidden nodes connection, hidden nodes to output nodes. Initial weights are assigned randomly between -0.5 to 0.5.

Step 4: Input to hidden neuron (I_i) multiply with weight w_{ih}, initially assigned.

Step 5: The output from each hidden neuron (O_h) is calculated using sigmoid function

$$S_1 = 1/(1 + e^{-\lambda x}) \text{ where } \lambda = 1 \text{ and } x = \Sigma_i w_{ih} I_i$$

where w_{ih} is the weight assigned between input and hidden layer and k_i is the input value.

Step 6: The input to the output layer (I_o) is find using multiply by weight w_{ho} with output of hidden O_h

Step 7: The output from the output layer (O_o) is calculated using the sigmoid function:

$$S_2 = 1/(1 + e^{-\lambda x}) \text{ where } \lambda = 1 \text{ and } x = \Sigma w_{ho} O_h$$

where w_{ho} is the weight assigned i between hidden and output layer and O_h is the output value from hidden neurons

Step 8: S_2 is subtracted from the desired output. Using this error (e) value, the weight change is calculated as: delta = $e \times S_2 \times (1 - S_2)$

Step 9: Update the weights using this delta value.

Step 10: $W_{ho} = W_{ho} + (n \times delta \times S_1)$

Step 11: $W_{ih} = W_{ih} + (n \times delta \times I_i)$ where n is the learning rate, I is the input values.

Step 12: Perform Steps 5-10 with the updated weights till the target output is equal to the desired output.

Fig. 6: BPNN learning algorithm

$$I_h = O_i \times W_{ih}$$

$$I_{ih\,1} = 0.00369 \times -0.7696789 = -0.002840115141$$

Output of hidden: Compute sigmoid function as:

$$S_l = 1/(1 + e^{-\lambda x})$$

Where:
$\lambda = 1$
$x = \Sigma_i w_{ih} O_i$

$$S_l = 1/(1+e^{-(-0.002840115141+-0.00202206834+2.784843e-4+0.0015781502+0.00334991853)})$$
$$S_l = 0.5000860924038992$$

Weight between hidden to output: Assign the weights randomly between hidden to output layer as:

$$W_{ho} = (-0.982075, 0.6894, -0.04521, -0.0879, -0.09341)$$

Input of output layer: Multiply the weight between hidden and output layer (W_{ho}) and the output of hidden layer:

$$I_o = S_l \times W_{ho}$$
$$I_o = 0.5000860924038992 \times (-0.982075)$$
$$I_o = -0.4912688993669402$$

Output of output layer: Sigmoid function of output layer is calculated as follows:

$$O_o = 1/(1 + e^{-\lambda x})$$

Where:
$\lambda = 1$
$x = \Sigma_i w_{ho} O_h$

$$O_o = 1/(1+ e^{-(-0.4912688+0.344282+-0.0226156+-0.044055+-0.0465606)})$$
$$O_o = 0.435310$$

Error signal: Compute the error signal using the formula as Error = $(T_o\text{-}O_o)^2$ where T_o is target output and is assigned -0.1 and O_o is output of output and is assigned -0.435310:

$$\text{Error} = (T_o - O_o)^2$$
$$\text{Error} = (0.1 - 0.435310)^2$$
$$\text{Error} = 0.1124328894755997$$

Backward pass: Weights are adjusted to achieve the target output and reduce the error value:

$$D = (T_o\text{-}O_{o1})(O_{o1})(1\text{-}O_{o1})$$
$$D = (0.1\text{-}0.435310)\ (0.435310)\ (1\text{-}0.435310)$$
$$D = -0.08242429601970899$$

Output to hidden weight:

$$Y = S1 \times D$$
$$Y = [0.50008\ \ 0.49939 \times -0.082424296$$
$$0.50023\ \ 0.50119\ \ 0.49841]$$
$$Y = [-0.041218741953536\ \ -0.04116219888646$$
$$-0.04123151770956\ \ -0.04131089230660$$
$$-0.04108480246268]$$
$$[\Delta w]^1 = [\Delta w]^0 + \eta[y][\text{Assume } \eta = 0.6]$$
$$[\Delta w]^0 = 0$$
$$[\Delta w]^1 = 0.6\ [-0.041218741953\ -0.04116219888646$$
$$-0.04123151770956\ -0.0413108923066$$
$$-0.0410848024626]$$
$$[\Delta w]^1 = [-0.024731546\ -0.024697349$$
$$-0.024738941\ -0.024786546$$
$$-0.02465088]$$

Hidden to input weight: The adjusted weight between hidden to input:

$$[e] = [w][D]$$
$$= (-0.9820)\ (-0.08242429601970899)$$
$$= 0.08094065869135422$$

Similarly the remaining four weights are multiplied by the error Difference value (D):

$$[D^*] = [e][O_H][1\text{-}O_H]$$
$$= [0.08094065869135422]\ [0.50008]\ [1\text{-}0.50008]$$
$$= 0.02023516415481834$$
$$[x] = [S_l][D^*]$$
$$= [0.5000\ \ 0.49939 \times 0.02023516415481834$$
$$0.50023\ \ 0.50119\ \ 0.4984]$$
$$[\Delta v]^1 = \alpha[\Delta v]^0 + \eta[x]$$
$$[x] = [0.01011758207740917$$
$$0.01010523862727473$$
$$0.01012223616516478$$
$$0.0101416619227534$$
$$0.01008520581476146]$$
$$[\Delta v]^1 = 0.6 \times [0.01011758207740917$$
$$0.01010523862727473$$
$$0.01012223616516478$$
$$0.0101416619227534$$
$$0.01008520581476146]$$
$$[\Delta v]^1 = [\ 7.467458159e\text{-}5\ \ -5.2419426329e\text{-}5$$
$$3.4376054e\text{-}6\ \ -1.57123260e\text{-}4$$
$$7.102493535e\text{-}6]$$

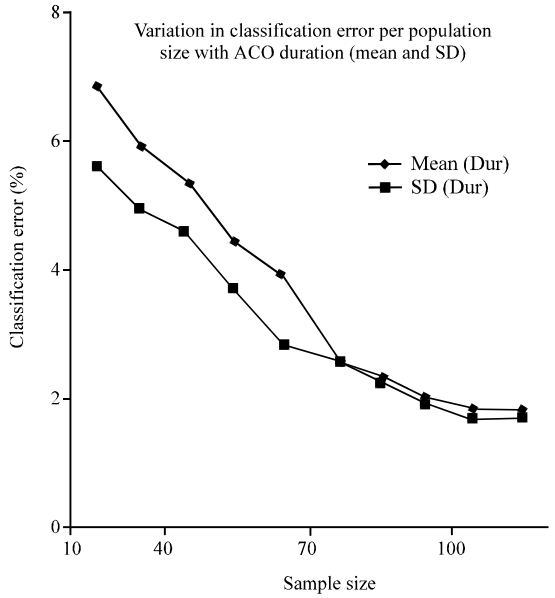

Fig. 7: Classification error1

Similarly, twenty five weights are calculated and old weights of input to hidden layer are replaced. After training the user typing pattern, fix the threshold values for each trained user. Again the users are asked to verify by giving the user name and password. After the verification of user name and password, the typing pattern is verified through the comparison of desired output with fixed threshold value. If the error value is <0.001 then the user is considered as valid user otherwise invalid user. FAR and FRR rate is calculated using the following equations:

$$FAR = FA/N \times \text{No. of user's}$$

Where:

FAR = False Acceptance Rate
FR = Number of incidence for False Acceptance
N = Total number of samples

$$FRR = FR/N \times \text{No. of user's}$$

Where:

FRR = False Rejection Rate
FR = Number of incidence for False Rejection
N = Total number of samples

Receiver Operating Characteristics (ROC): ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution. ROC analysis is related in a direct and natural way to cost/benefit analysis of diagnostic decision making.

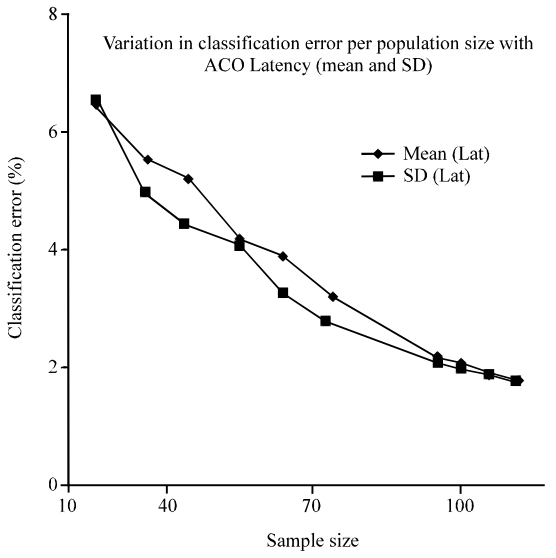

Fig. 8: Classification error2

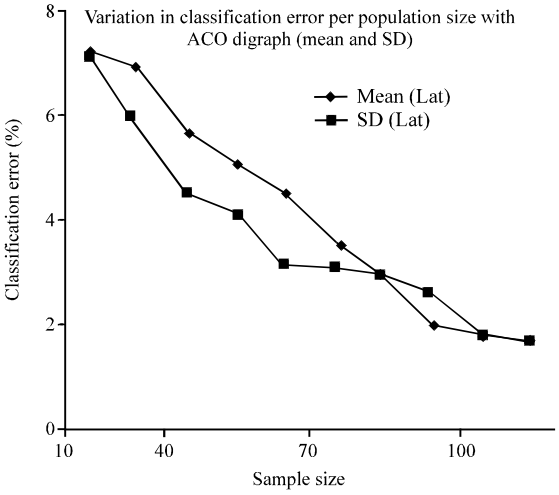

Fig. 9: Classification error3

Figure 6- 9 shows the ROC curves for comparison of mean and standard deviation (duration, latency and digraph) of classification performance. The error rate reduced ones the population size increased. The following figure shows the error rates of various users for the different number of sample collections.

CONCLUSION

Features are extracted from 10 users and each user wants to type 50 samples. Using that samples mean and standard deviation is calculated for duration, latency and digraph. subsets of features are selected Ant Colony Optimization Algorithm. The features are classified

using Backpropagation Algorithm using Jack-Knife Method. ACO using duration mean provide the best performance comparing with latency and digraph. The classification error was 0.167%.

REFERENCES

Azevedo, G.L.F. and G.D.C. Cavalcanti, 2007. An approach to feature extraction for keystroke dynamics systems based on pso and feature weighting. Proceedings of the IEEE Congress on Evolutionary Computation, September 25-28, 2007, Singapore, pp: 3577-3584.

Azevedo, G.L.F., D.G.C. Cavalcanti and E.C.B. Carvalho Filho, 2007. Hybrid solution for the feature selection in personal identification problems through keystroke dynamics. Proceedings of the International Joint Conference on Neural Networks, August 12-17, 2007, Orlando, FL, pp: 1947-1952.

Bergadan, F., D. Gunetti and C. Picardi, 2002. User authentication through keystroke dynamics. ACM Trans. Info. Syst. Security, 5: 367-397.

Blackburn, D., C. Miles, B. Wing and K. Shepard, 2007. Biometrics overview report. National Science and Technology Council Sub-Committee on Biometrics.

Bleha, S.A. and M.S. Obaidat, 1991. Dimensionality reduction and feature-extraction applications in identifying computer users. IEEE Transact. Syst. Man Cybernetics, 21: 452-456.

Boechat, G.C., C.J. Ferreira and C.B Edson and C. Filho, 2007. Authentication personal. Proceedings of the International Conference on Intelligent and Advanced Systems, November 25-28, 2007, Kuala Lumpur, Pp: 254-256.

Brown, M. and S.J. Rogers, 1993. User identification via keystroke characteristics of typed names using neural networks. Int. J. Man Mach. Stud., 39: 999-1014.

Daw-Tung, L., 1997. Computer-access authentication with neural network based. Proceedings of the International Conference on Neural Networks, June 9-12, 1997, Houston, TX, pp: 174-178.

Dorigo, M. and L.M. Gambardella, 1997. Ant colonies for the traveling salesman problem. Bio Systems, 43: 73-81.

Dorigo, M., V. Maniezzo and A. Colorni, 1991. Positive feed back as a search strategy. Technical Report Politecnico di Milano, 58426 Italy.

Gaines, R., W. Lisowski and N. Shpiro, 1980. Authentication by keystroke timing: Some preliminary results. Rand Report R-256-NSF. Rand Corporation, Grand No. MCS.

Haider, S., A. Abbas and A.K. Zaidi, 2000. A multi-technique approach for user identification through keystroke dynamics. Proceedings of the International Conference on Systems, Man and Cybernetics, 2000 IEEE, October 8-11, 2000, Nashville, TN, pp: 1336-1341.

Hecht-Nielsen, R., 1990. Neurocomputing. Addison-Wesley Publishing Co., Reading, MA., USA.

Hocquet, S., J.Y.V. Ramel and H. Cardot, 2005. Fusion of methods for keystroke dynamics authentication. Proceedings of the 4th Workshop on Automatic Identification Advanced Technologies, October 17-18, 2005, Buffalo, New York, USA., pp: 224-229.

Hong, L. and A.K. Jain, 1998. Integrating faces and fingerprints for personal identification. IEEE Trans. Patt. Anal. Mach. Intell., 20: 1295-1307.

Hu. J., D. Gingrich and A. Sentosa, 2008. A k-Nearest neighbor approach for user authentication through biometric keystroke dynamics. Proceedings of the IEEE International Conference on Communications, May 19-23, 2008, Beijing, pp: 1556-1560.

Jain, A., L. Hong and S. Pankanti, 2003. Biometrics identification system. Signal Processing, 83: 2539-2557.

John, G. and R. Kohavi, 1994. Irrelevant features and the subset selection problem. 11th International Conference on Machine Learning, 1994, Morgan Kaufmann Publishers, pp: 121-129.

Kapczynski, A., P. Kasprowki and P. Kuzniacki, 2006. Modern access control based on eye movement analysis and keystroke dynamics. Proceedings of the International Multiconference on Computer Science and Information Technology, April 5-7, 2006, Madison WI., USA., pp: 477-483.

Karnan, M., K. Thangavel, R. Sivakumar and K. Geetha, 2006. Ant colony optimization for feature selection and classification of microcalcifications in digital mammograms. Proceedings of the International Conference on Advanced Computing and Communications, December 20-23, 2006, Surathkal, pp: 298-303.

Leberknight, C.S., G.R. Widmeyer and M.L. Recce, 2008. An investigation into the efficacy of keystroke analysis for perimeter defense and facility access. Proceedings of the IEEE Conference on Technologies for Homeland Security, May 12-13, 2008, Waltham, MA, pp: 345-350.

Li, Y. and Z. Xu, 2003. An ant colony optimization heuristic for solving maximum independent set problems. Proceedings 5th International Conference on Computational Intelligence and Multimedia Applications, September 27-30, 2003, China, pp: 206-211.

Monrose, F. and A.D. Rubin, 2000. Keystroke dynamics as a biometric for authentication. Future Gen. Comput. Syst., 16: 351-359.

Monrose, F., M.K. Reiter and S. Wetzel, 1999. Password hardening based on keystroke dynamics. Proceedings of the 6th ACM conference on Computer and Communications Security, November 1-4, 1999, Berlin, Heidelberg, pp: 73-82.

Obaidat, M.S. and D.T. Macchairolo, 1991. A multilayer neural network system for computer access security. Ind. Electron., 40: 235-242.

Pavaday, N. and K.M.S. Soyjaudah, 2007. Investigating performance of neural networks in authentication using keystroke dynamics. Proceedings of the the 8th IEEE Africon Conference, September 26-28, 2007, Windhoek, Namibia, pp: 1-8.

Rumelhart, D.E., G.E. Hinton and R.J. Williams, 1987. Learning Internal Representations by Error Propagation. In: Parallel Distributing Processing, Rumelhart, D.E., G.E. Hinton and R.J. Williams (Eds.). MIT Press, Cambridge, MA, pp: 318-362.

Shepherd, S.J., 1995. Continuous authentication by analysis of keystroke typing characteristics. Proceedings of the European convention on security and detection, May 16-18, 1995, Brighton, pp: 111-114.

Shiv Subramaniam, K.N., S.R. Bharath and S. Ravinder, 2007. Improved authentication mechanism using keystroke analysis. Proceedings of the International Conference on Information and Communication Technology ICICT 07, March 7-9, 2007, Dhaka, pp: 258-261.

Singhi, S.K. and H. Liu, 2006. Feature subset selection bias for classification learning. Proceedings of the 23rd International Conference on Machine Learning, June 14-18, 2009, Montreal, Canada, pp: 849-856.

Sung, K.S. and S. Cho, 2006. GA SVM wrapper ensemble for keystroke dynamics authentication. Proceedings of the International Conference on Biometrics, January 5-7, 2006, China, Hong Kong, pp: 654-660.

Teh, P.S., A.B.J. Teoh, T.S. Ong and H.F. Neo, 2007. Statistical fusion approach on keystroke dynamics. Proceedings of the 3rd International Conference on Signal-Image Technologies and Internet-Based System, December 16-18, 2007, Shanghai, China, pp: 918-923.

Yang, J. and V. Honavar, 1998. Feature subset selection using a genetic algorithm. IEEE Intell. Syst., 13: 44-49.

Young, J.R. and R.W. Hammon, 1989. Method and apparatus for verifying an individual's identity. US Patent 6862610, US Patent Issued on March 1, 2005, Estimated Patent Expiration.

Yu, E. and S. Cho, 2003. GA-SVM wrapper approach for feature subset selection in keystroke dynamics identity verification. Proceedings of the International Joint Conference on Neural Networks, July 20-24, 2003, Beach, Portland, Oregon, pp: 2253-2257.

Yu, E. and S. Cho, 2004. Keystroke dynamics identity verification-its problems and practical solutions. Comput. Security, 23: 428-440.