Computer User Authentication using Hidden Markov Model through Keystroke Dynamics

Sampath K. Vuyyuru, Vir V. Phoha, Shrijit S. Joshi
Louisiana Tech University
Ruston, LA 71272

Shashi Phoha, Asok Ray
Pennsylvania State University
University Park, PA 16802

Correspondence E-mail: phoha@latech.edu

ABSTRACT: We present a novel computer user authentication technique using hidden Markov model (HMM) through keystroke dynamics. We propose: (i) modified HMM parameters to reduce the order of computations involved in the forward (or backward) procedure by T^2 (reduction is from ${N_{sc}}^2T^3$ to ${N_{sc}}^2T$ where T and N_{sc} represents the length of the keystroke pattern and the number of states of an HMM per character respectively) and (ii) a strategy for estimating the number of states and the number of training iterations of an HMM. For each user, a distinct HMM is developed using modified Rabiner's re-estimation formulae of multiple observation sequences on six reference keystroke patterns. Authentication of a user is made in two stages: (i) the user identification stage, wherein we determine the user with the maximum probability score for the given keystroke pattern and (ii) the user verification stage, wherein we determine the probability score for the given keystroke pattern for a claimed user. Finally, a decision about the authenticity of a user is made using the results of both the stages and threshold criteria. Data for our experiments was collected from a group of 43 users; for training data, each user provided a set of nine reference keystroke patterns for the string "master of science in computer science," and for testing data, the number of keystroke patterns for each user varied from 0 to 102 with a total of 873 keystroke patterns. We obtained the best false accept rate of 0.74 % when the false reject rate was 8.06 % and the area under the receiver operating characteristics curve was 0.99603.

Keywords: Keystroke dynamics, Computer Security, User identification, User verification, User authentication, Hidden Markov Model, Receiver Operating Characteristics Curve, Area under Curve

This work was supported in part by the Army Research Office under Grant DAAD 19-01-1-0646

ACRONYMS

HMM	Hidden Markov Model
TP	True Positive
TN	True Negative
FAR	False Accept Rate
FRR	False Reject Rate
EER	Equal Error Rate
ROC	Receiver Operating Characteristics
AUC	Area under Curve

1 INTRODUCTION

In computer security, user authentication is the process by which a user attempts to confirm his/her claimed identity by providing relevant information such as a password. Different possible cases in a user authentication system are shown in Illustration 1.

Illustration 1: *In user authentication system, for a given attempt by a user, any one of the following four cases can happen where U_1 is the registered user (user known to the system) and U_2 is the unregistered user (user unknown to the system):*

Case 1: U_1 claims as U_1 and if it gets accepted then it is termed as a True Positive

Case 2: U_1 claims as U_1 and if it gets rejected then it is termed as a False Reject

Case 3: U_2 claims as U_1 and if it gets accepted then it is termed as a False Accept

Case 4: U_2 claims as U_1 and if it gets rejected then it is termed as a True Negative

We can see from Illustration 1 that an authentication system should have a low FAR (this is also refereed to as a Type II error [5]) and a low FRR (this is also refereed to as a Type I error [5]). In addition, the ROC [12] curve can be used to visualize the threshold independent performance of an authentication system. To compare the performance of two or more authentication systems, the AUC [12] value can be used, where the AUC value is calculated by finding the area under the ROC curve (the higher the AUC value the better is the authentication system).

1.1 Keystroke Dynamics for User Authentication

Keystroke dynamics [11] is a behavioral pattern exhibited by an individual while typing on a keyboard. User authentication through keystroke dynamics is appealing for many reasons such as: (i) it is not intrusive, and (ii) it is relatively inexpensive to implement, since the only hardware required is the computer [8]. Researchers have developed user authentication systems through keystroke dynamics using various pattern recognition techniques like neural networks [6, 19, 23, 24], statistical classification techniques [7, 14, 27, 29], decision trees [33], and others [9, 14]. Most of these systems cannot add or delete user(s) without retraining the entire system. But in this paper, we propose a method which can: (i) dynamically add or remove users without retraining the entire system, and (ii) adapt to the changing typing patterns of the user(s).

1.2 Motivation for using Hidden Markov Models

HMMs have proven to be useful in a variety of real world applications where considerations for uncertainty are crucial [15]. With an ability to handle the variability in speech signals, HMMs have proved to be an efficient model for statistically modeling speech signals which can be seen from the extensive application of HMMs for speech recognition such as methods proposed in [2-4, 17, 18, 25]. In the context of user authentication through keystroke dynamics, keystroke events have a non-deterministic nature; hence, modeling keystroke patterns with HMMs, which has the ability of handling stochastic process, can be used to recognize the keystroke patterns of a user.

1.3 A Brief Introduction to the Proposed Method

Initially, keystroke patterns are mapped to the speech signals and the parameters of the HMM are defined according to the mapping. But in Section 4.3, we will see that the resulting model has a very high number of computations. Therefore, in order to reduce the number of computations, we modified the two-dimensional state transition matrix into a three-dimensional matrix by eliminating the state transitions with zero probabilities. Other parameters of the HMM are also modified accordingly.

The proposed method has two phases: (1) the training phase and (2) the user authentication phase. In the training phase, a distinct HMM is modeled for each registered user with the modified HMM parameters, which is trained on six reference keystroke patterns using the modified Rabiner's re-estimation formulae of multiple observation sequences. While training, the number of states and the number of training iterations of an HMM is determined using an estimation strategy. The user authentication phase consists of two stages: (i) the user identification stage and (ii) the user verification stage. In the user identification stage, a one-to-many search is done to determine the user with the maximum probability score for the given keystroke pattern, and in the user verification stage, the probability score of the claimed user for the given keystroke pattern is determined. Finally, the decision about the authenticity of a user is made using the results of both the stages and threshold criteria.

Testing of our method is done on the same data set on which Sheng et al. [33] tested their method. The AUC [12] value obtained by our method is 0.99603 and by the Sheng et al. [33] method is 0.98873 (calculated from the error rates given in [33]). Furthermore, for evaluating the effectiveness of our method, we created three data sets by changing the number of registered/unregistered users. These three data sets consist of 43, 36, 25 registered users and 0, 7, 18 unregistered users respectively. We observed that the AUC value for all the sets is almost the same, which indicates that the average performance of the method did not change with a change in the number of registered/unregistered users.

1.4 Contributions of this Paper

Contributions of this paper are enumerated as follows:

(1) Developing a novel user authentication technique using the HMM

(2) Modifying the HMM parameters for reducing the order of computations (by T^2 where T represents the length of the keystroke pattern) involved in the forward-backward procedure of the HMM as compared to the method proposed by Rabiner in [28]

(3) Developing a strategy for estimating the number of states and the number of training iterations of an HMM

(4) Adding and removing of user(s) without retraining the entire system

(5) Updating the users' templates according to the changing typing patterns of the users

1.5 Organization of this Paper

The rest of the paper is organized as follows: Section 2 deals with the data collection for our method; Section 3 deals with a brief introduction to HMMs; Section 4 deals with the mapping of the HMM parameters to keystroke dynamics; Section 5 deals with the procedure for training an HMM for each user; Section 6 describes the procedure for user authentication;

experimental results are discussed in Section 7; finally, analysis of our method is given in Section 8 and conclusion and future work is discussed in Section 9.

2 DATA COLLECTION

The data set we used in our experiment is the same as that of Sheng et al. [33]. For collecting training and testing data sets, an experiment was conducted from November to December 2002. Forty-three users (labeled as user 1 to user 43) in the experiment provided a set of nine keystroke patterns (referred to as reference patterns) for the string, “master of science in computer science,” to set up an account. However, the number of test patterns for each user varied from 0 to 102 with a total of 873 patterns.

2.1 Data Preprocessing

For each keystroke pattern, we collected key press and key release times for all the characters. From key press and key release times, the following four features can be extracted: (1) key hold time, (2) key press latency, (3) key release latency, and (4) key interval time as shown in Figure 1.

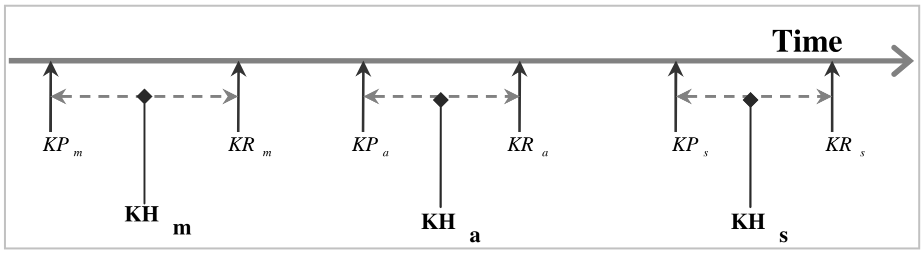


Figure 1: Determination of key hold times from the key press and key release times for the characters 'm', 'a', and 's' of the string "mas"

Illustration 2: *As shown in Figure 1, timing data captured for the string "mas" are: key press and key release times for the characters 'm', 'a', and 's'. Using key press and key release times: (1) key hold times are determined using the formula:* $KH_m = KR_m - KP_m$, *(2) key press latencies are determined using the formula:* $KPL_{ma} = KP_a - KP_m$, *(3) key release latencies are determined using the formula:* $KRL_{ma} = KR_a - KR_m$, *and (4) key interval times are determined using the formula:* $KI_{ma} = KP_a - KR_m$, *where* KH_m, KR_m, KP_m *represents the key hold, key release, and key press times for the character 'm' respectively, and* KP_a, KR_a *represents the key press and key release times for the character 'a' respectively, and*

KPL_{ma}, KRL_{ma}, and KI_{ma} represents key press latency, key release latency, and key interval time between the characters $'m'$ and $'a'$ respectively.

2.2 Feature Vector

Studies indicate that key hold times are more effective than key press latencies, key release latencies and key interval times for user authentication [16, 22, 26]. Accordingly, in our method, we used only key hold times for our feature vector. Key hold times for the space characters were not considered because while typing the space character, the user may pause for recollection of what has to be typed next. Thus, for each keystroke pattern, we have 32 key hold times which constitute our feature vector.

3 BACKGROUND[1]

3.1 A Brief Introduction to Hidden Markov Models

An HMM is a finite state machine where the system being modeled is assumed to be a Markov process with unknown parameters, where the unknown parameters are determined from the observable parameters of the system [28].

Definition 1: *An HMM λ is a five-tuple (S, V, A, B, π) where S represents the set of states, V represents the set of* observation *symbols, A represents the state transition probability matrix, B represents the observation symbol probability distribution and π represents the initial state matrix. $\lambda = \{A, B, \pi\}$ is the compact notation to indicate the complete parameter set of HMM.*

Notations of various terms related to Definition 1 are: (1) $S = \{S_1, S_2 ... S_N\}$ where N denotes the number of states and a state at time t is denoted as q_t, (2) $V = \{V_1, V_2, ... V_M\}$ where M denotes the number of distinct observation symbols, and an observation sequence is represented as $O = \{o_1, o_2, ... o_T\}$ where sub-script T denotes the number of observations in an observation sequence, and an observation symbol at time t is denoted as o_t, (3) $A = \{a_{ij}\}$ where a_{ij} represents the state transition probability from state i to state j, (4) $B = \{b_j(o_t)\}$ where $b_j(o_t)$ represents the probability of observing symbol o_t in state j, and (5) $\pi = \{\pi_i\}$ where π_i represents the initial state probability of state i $(\pi_i = P[S_i = q_1]$ where $1 \le i \le N)$.

[1]All the material given in this section is based on [28]

Illustration 3: *Figure 2 illustrates an HMM with 4 states labeled as S_1, S_2, S_3 and S_4; Figure 3 illustrates the state transitions and observation symbols in an HMM where the probability of transition from state S_i to state S_j is a_{ij}, and the probability of observing symbols V_m and V_n from the respective states S_i and S_j is given by $b_i(o_t)$ and $b_j(o_{t+1})$ respectively.*

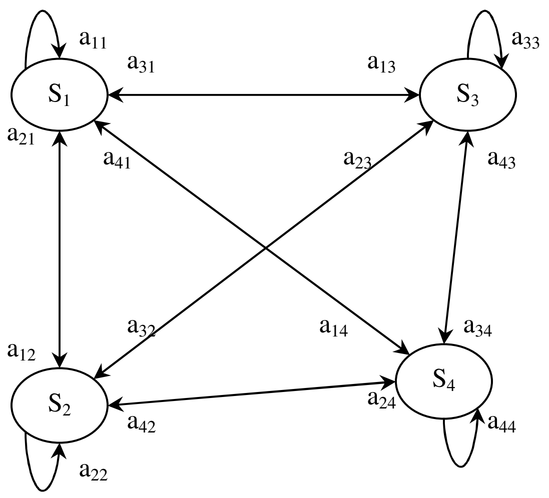


Figure 2: Illustration of an HMM with four states labeled as S_1, S_2, S_3 and S_4

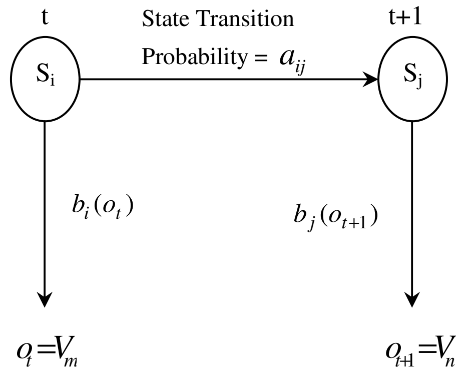


Figure 3: Illustration of State Transitions and Observation Symbols in an HMM

3.2 Three Basic Problems of HMMs

For HMMs to be useful in real world applications, the following three problems must be solved:

Problem 1: Given an observation sequence O and a model λ, how do we efficiently compute $P(O/\lambda)$

Problem 2: Given an observation sequence O and a model λ, how do we choose the corresponding state sequence $Q = q_1, q_2, ...q_T$ which best explains the observations

Problem 3: How do we adjust the model parameters $\lambda = \{A, B, \pi\}$ to maximize $P(O/\lambda)$

Problem 1 is an evaluation problem which aims at finding the likelihood of an observation sequence produced by a given model. The solution of this problem can be used in pattern recognition applications. For example, if we are to choose a model among several competing models, the solution of this problem gives the likelihood score and helps in finding the best matching model. Problem 2 tries to uncover the hidden part of the HMMs, i.e., to find the correct state sequence. But in our model, as we will see in the next section, the states do not have any physical significance. So, we do not address this problem. Problem 3 is a parameter optimization problem wherein, the likelihood of an observation sequence on a given model is maximized by adjusting the model parameters (A, B, π). The solution to this problem is used in training HMMs.

To solve the above-mentioned problems, forward and backward variables (defined below) are used.

Definition 2: *Forward variable $\alpha_t(j)$ represents the probability of the partial observation sequence, '$o_1, o_2, ...o_t$' and state j at time t given a model λ. It is estimated using the forward procedure as shown below:*

Forward Procedure

1) Initialization:	$\alpha_1(i) = \pi_i b_i(o_1),$	$1 \le i \le N$
2) Induction:	$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}),$	$1 \le j \le N$, $1 \le t \le T-1$
3) Termination:	$P(O/\lambda) = \sum_{i=1}^{N} \alpha_T(i)$	

Definition 3: *Backward variable $\beta_t(j)$ represents the probability of the partial observation sequence, '$o_{t+1}, o_{t+2}, ...o_T$' and state j at time t given a model λ. It is estimated using the backward procedure as shown below:*

Backward Procedure:

1) Initialization:	$\beta_T(i) = 1,$	$1 \le i \le N$
2) Induction:	$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j),$	$1 \le i \le N$, $t = T-1, T-2...1$

In addition to the forward and backward variables, variable γ is also used in solving the re-estimation problem i.e., Problem 3.

Definition 4: *The variable $\gamma_t(j)$ represents the probability of being in state j at time t given a model λ and an observation sequence O. It is estimated using forward and backward variables as shown below:*

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}, \quad \begin{array}{l} 1 \le t \le T \\ 1 \le i \le N \end{array}$$

4 MAPPING HMM PARAMETERS TO KEYSTROKE DYNAMICS

Many real world processes that produce observable symbols can be modeled as signals, and these signals can be characterized using various signal modeling techniques [28]. HMM is one such technique, which has been extensively applied in speech recognition systems and various other applications [2-4, 17, 18, 25, 31].

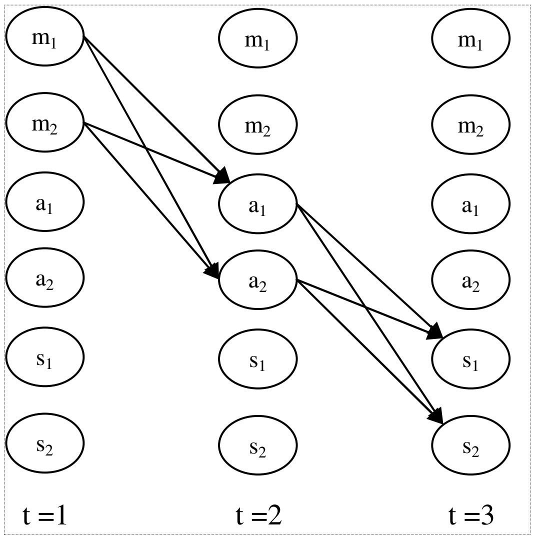
If the key hold times for the characters of a particular string (of a user) are considered as energy levels (amplitude) of a signal (signal wave), then the user can be thought of as a signal source whose signal can be in different energy levels. These energy levels can be interpreted as the states of the signal. For such a signal, the number of states will be at least the number of characters in the string. Furthermore, each character can be in one or more number of sub-states which are unknown. Moreover, the individual key hold times constitute the observable symbols of the signal. It follows from the above discussion that the pattern of key hold times can be modeled using an HMM and can be used to authenticate users based on their typing patterns.

4.1 Mapping the States of the HMM to Keystroke Dynamics

The states of our model are the characters present in the string typed by the user. Moreover, for each character, there are a certain number of sub-states (N_{sc}, which are also referred to as states hereafter). We assumed an equal number of sub-states for all the characters for a particular model and the actual number of sub-states for the model is estimated while training the model. Hence, the total number of states N in a model with T characters is given by $N = N_{sc}T$.

Illustration 4: *Consider the first three characters of the reference string typed by a user whose N_{sc} is 2. The value of T is 3 (as there are three characters), so we have*

$N = 6$ (as $N = N_{sc}T$). As shown in Figure 4, at time $t = 1$, state transitions are only from the sub-states of $'m'$ (m_1 or m_2) to the sub-states of $'a'$ (a_1 or a_2) and at time $t = 2$, state transitions are only from the sub-states of $'a'$ (a_1 or a_2) to the sub-states of $'s'$ (s_1 or s_2). Figure 4 illustrates the state transition matrix and initial state distribution of user 4 for the first three characters ("mas") of the reference string where N_{sc} is 2.



$$A = \begin{array}{c|cccccc} & m_1 & m_2 & a_1 & a_2 & s_1 & s_2 \\ \hline m_1 & 0 & 0 & 0.66 & 0.34 & 0 & 0 \\ m_2 & 0 & 0 & 0.80 & 0.20 & 0 & 0 \\ a_1 & 0 & 0 & 0 & 0 & 0.70 & 0.30 \\ a_2 & 0 & 0 & 0 & 0 & 0.58 & 0.42 \\ s_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_2 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$\pi = \begin{array}{c|c} m_1 & 0.58 \\ m_2 & 0.42 \\ a_1 & 0 \\ a_2 & 0 \\ s_1 & 0 \\ s_2 & 0 \end{array}$$

Figure 4: State transitions in the HMM for the string "$\mathbf{mas}$" with two states for each character where $\{\mathbf{m}_1, \mathbf{m}_2\}$, $\{\mathbf{a}_1, \mathbf{a}_2\}$, and $\{\mathbf{s}_1, \mathbf{s}_2\}$ represent the sub-states for the characters $'\mathbf{m}'$, $'\mathbf{a}'$, and $'\mathbf{s}'$ respectively

We can see from Illustration 4 that the state transitions in the HMM of our model are only from the states of t^{th} character to the states of $(t+1)^{th}$ character at time instant t. This implies that for a_{ij} to have a non-zero probability value, state i must belong to one of the sub-states of the t^{th} character (i.e. $((t-1)N_{sc}+1) \le \; i \; \le (tN_{sc})$) and state j must belong to one of the sub-states of the $(t+1)^{th}$ character (i.e. $(tN_{sc}+1) \le \; j \; \le ((t+1)N_{sc})$). This can be represented as follows:

$$a_{ij} = \begin{cases} P[S_{t+1}=j|S_t=i], & \text{if } ((t-1)N_{sc}+1) \le i \le (tN_{sc}),\; (tN_{sc}+1) \le j \le ((t+1)N_{sc}) \\ 0, & \text{otherwise} \end{cases}$$

For example, in order for the state transition probability value (a_{ij}) of Illustration 4 to have a non-zero value, at time $t = 1$, i must be either 1 or 2 (m_1 or m_2) and j must be either

3 or 4 (a_1 or a_2) i.e., $1 \le i \le 2$ and $3 \le j \le 4$, which can be obtained by substituting $t = 1$ in the above equation for non-zero a_{ij} value.

We can see from Figure 4 that only the states of the first character (m_1, m_2) can have non-zero initial state probabilities. This implies that in order for π_i to have a non-zero probability value, i must be one of the states of the first character $(1 \le i \le N_{sc})$. This can be represented as follows:

$$\pi_i = \begin{cases} P[q_1 = S_i], & \textit{if } 1 \le i \le N_{sc} \\ 0, & \textit{otherwise} \end{cases}$$

4.2 Mapping the Observation Symbols of the HMM to Keystroke Dynamics

The individual key hold times constitute the observation symbols of the model. Most of the previous researchers have either explicitly or implicitly assumed that the keystroke features follow the Gaussian distribution [7, 10, 13, 30, 33]. In addition, we can see from Figures 5(a) and 5(b) that key hold times follow a Gaussian like distribution. Hence, we assumed that the probability of the key hold times follow the Gaussian distribution. Thus, the probability of observing a symbol (key hold time) o_t in state j with mean μ_j and standard deviation σ_j is given by: $b_j(o_t) = N(o_t, \mu_j, \sigma_j)$.

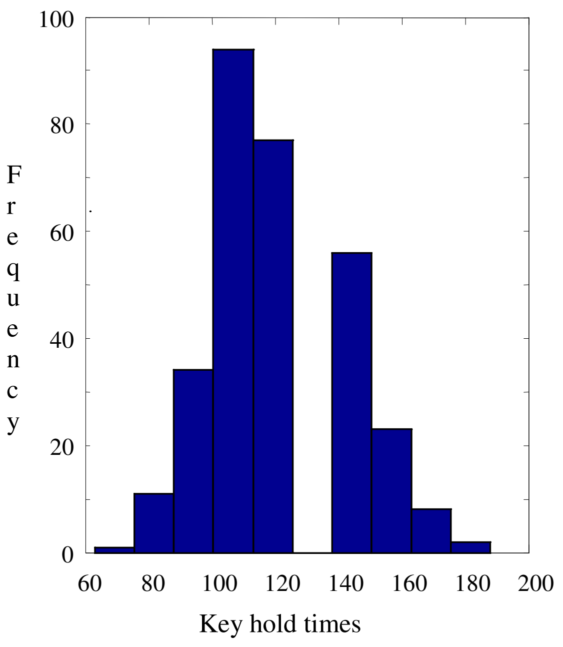


Figure 5(a): Frequency histogram of key hold times for character ‘e’ of User 2

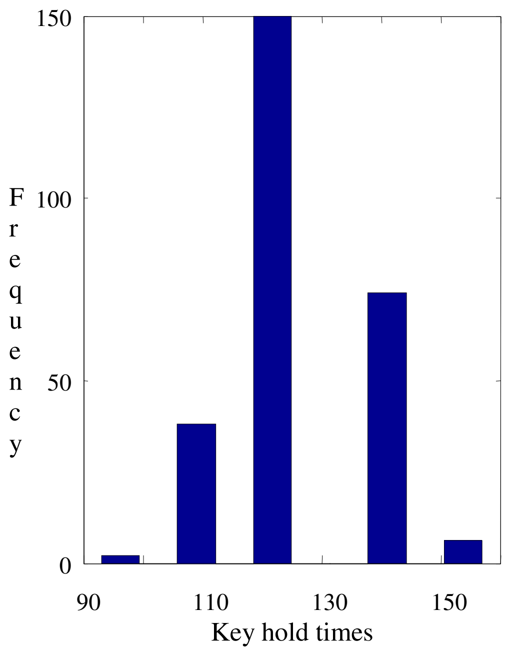


Figure 5(b): Frequency histogram of key hold times for character ‘e’ of User 21

4.3 Modifying the HMM Parameters

The computations involved in the forward (or backward) procedure is in the order of N^2T [32]. Thus, for example, if $N_{sc} = 3$ and $T = 32$, we need about 300,000 computations.

In the above example, as $N = 96$ ($N_{sc}T$) the size of the state transition matrix is 96×96. The reason for such a high number of computations can be attributed to the size of the state transition matrix. But in our model, as we have already seen, at any given instant of time only N_{sc} states can have non-zero probability values for state transitions. Hence, in each column of the state transition matrix, there are at most N_{sc} non-zero values (this can be seen from Figure 4). But, as all the values take part in the computations, even though most of them result in zeros, the total number of computations is quite high. Therefore, in order to reduce the total number of computations, the structure of the HMM parameters was modified (The modified parameters are represented with a bar over the parameters in order to distinguish from the original parameters).

4.3.1 Modifying the Initial State and the State Transition Probabilities of the HMM

The state transition matrix is chosen to be a three-dimensional matrix ($\overline{A} = \{\overline{a}_{trs}\}$) where the first dimension corresponds to the number of characters in the observation sequence and the other two dimensions corresponds to the non-zero $N_{sc} \times N_{sc}$ sub-matrix of each character in the original state transition matrix. For example, if the character at time instant t is considered, then the other two dimensions correspond to the state transition probabilities from the states of the t^{th} character to the states of the $(t+1)^{th}$ character. The modified state transition matrix in terms of the original state transition matrix is given below:

$$\overline{a}_{trs} = a[r + (t-1)N_{sc}][s + tN_{sc}], \qquad 1 \le r, s \le N_{sc}, \quad 1 \le t \le T$$

We can see from the above equation that, at any particular instant of time, the size of the state transition matrix considered (while estimating forward or backward variables) is $N_{sc} \times N_{sc}$.

Only the states of the first character can have non-zero initial state probability values, so we changed the size of the initial state distribution (π) from N to N_{sc}.

4.3.2 Modifying the Mean and the Standard Deviation Vectors of the HMM

Similarly, the mean and standard deviation vectors were also chosen to be two-dimensional vectors (as opposed to one-dimensional vector) where the first dimension corresponds to the current character and the second dimension corresponds to the sub-states for that character. Thus, the probability of observing a symbol o_t in the s^{th} sub-states of the t^{th} character in terms of the modified mean and standard deviation vectors is $N(o_t, \mu_{ts}, \sigma_{ts})$; this in terms of the original observation symbol probability distribution can be represented as:

$$\overline{b}_s(t, o_t) = b_{s+(t-1)N_{sc}}(o_t), \qquad \begin{matrix} 1 \le s \le N_{sc} \\ 1 \le t \le T \end{matrix}$$

4.4 Modifying the Forward and Backward Procedures

The modified HMM parameters are used in the forward and backward procedures in order to reduce the number of computations required for estimating the forward and backward variables. The forward and backward procedures in terms of the modified HMM parameters are given below (where the modified parameters are substituted in place of the original HMM parameters in the forward and backward procedures and for the derivation of the forward and backward procedures refer to [28]):

Modified Forward Procedure

1) Initialization: $\overline{\alpha}_1(r) = \overline{\pi}_r \overline{b}_r(1, o_1), \qquad 1 \le r \le N_{sc}$

2) Induction: $\overline{\alpha}_{t+1}(s) = \left[\sum_{r=1}^{N_{sc}} \overline{\alpha}_t(r) \overline{a}_{trs} \right] \overline{b}_s(t+1, o_{t+1}), \qquad \begin{matrix} 1 \le s \le N_{sc} \\ 1 \le t \le T-1 \end{matrix}$

3) Termination: $P(O / \lambda) = \sum_{r=1}^{N_{sc}} \overline{\alpha}_T(r)$

Modified Backward Procedure

1) Initialization: $\overline{\beta}_T(s) = 1, \qquad 1 \le s \le N_{sc}$

2) Induction: $\overline{\beta}_t(r) = \sum_{s=1}^{N_{sc}} \overline{a}_{trs} \overline{b}_s(t+1, o_{t+1}) \overline{\beta}_{t+1}(s), \qquad \begin{matrix} 1 \le r \le N_{sc} \\ 1 \le t \le T-1 \end{matrix}$

Now, the order of computations involved in the forward procedure is $N_{sc}^2 T$ where the number of multiplications required is $N_{sc}(N_{sc}+1)(T-1)+N_{sc}$ and the number of additions required is $N_{sc}(N_{sc}-1)(T-1)$ (see Appendix for details). For the above example with $N_{sc}=3$ and $T=32$, we need about 300 computations as opposed to 300,000 computations with the original HMM parameters.

5 TRAINING PHASE

For each registered user an HMM is trained using six reference patterns. The number of states and the number of training iterations of an HMM are determined using an estimation strategy. The detailed description about the training procedure is given in the following sub-sections.

5.1 Initial Estimation

Model parameters $\overline{A}$ and $\overline{\pi}$ were initialized randomly, using uniform distribution over the interval $(0,1)$. Initial estimation of σ for the states of each character (which is chosen to be same for all the states of a particular character) is calculated using six reference patterns. Initial estimation of μ for the states of each character is chosen such that the difference between the mean values of two successive states is the standard deviation (of the six reference patterns) and the average of μ for all the states is the mean of the six reference patterns.

Illustration 5: *If there are five states for each character and μ_t, σ_t are the mean and standard deviation of t^{th} character of the six reference patterns, then the initial estimation for the mean and standard deviation of the observation symbols for the five states of t^{th} character are: $(\mu_t - 2\sigma_t, \sigma_t)$, $(\mu_t - \sigma_t, \sigma_t)$, (μ_t, σ_t), $(\mu_t + \sigma_t, \sigma_t)$, $(\mu_t + 2\sigma_t, \sigma_t)$ respectively. If there are only four states, then the initial estimation will be $(\mu_t - 3\sigma_t, \sigma_t)$, $(\mu_t - \sigma_t, \sigma_t)$, $(\mu_t + \sigma_t, \sigma_t)$, $(\mu_t + 3\sigma_t, \sigma_t)$.*

5.2 Parameter Re-estimation

As mentioned above, the model for each registered user is trained using six reference patterns. The forward variable and the backward variable for each reference pattern is estimated using the modified forward and backward procedures (as explained in Section 4.4).

Using these variables and the model parameters of all the reference patterns, the parameters of the final model are re-estimated using the modified Rabiner's re-estimation formulae of multiple observation sequences [28] as shown below:

Modified Rabiner's Re-estimation Formulae for Multiple Observation Sequences (in terms of the Modified HMM Parameters)

$$\overline{\pi}_r = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \gamma_1^k(r)}{\sum_{k=1}^{K} \frac{1}{P_k}} \qquad \overline{a}_{trs} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \left(\overline{\alpha}_t^k(r) \overline{a}_{trs} \overline{b}_s \left(t+1, o_{t+1}^k\right) \overline{\beta}_{t+1}^k(s) \right)}{\sum_{k=1}^{K} \frac{1}{P_k} \left(\overline{\alpha}_t^k(r) \overline{\beta}_t^k(r) \right)}$$

$$\overline{\mu}_{ts} = \frac{\sum_{k=1}^{K} \gamma_t^k(s) o_t^{(k)}}{\sum_{k=1}^{K} \gamma_t^k(s)} \qquad \overline{\sigma}_{ts} = \frac{\sum_{k=1}^{K} \gamma_t^k(s) \left(o_t^{(k)} - \overline{\mu}_{ts} \right)^2}{\sum_{k=1}^{K} \gamma_t^k(s)}$$

where, P_k represents the likelihood of the model on k^{th} reference pattern and all the HMM variables belonging to k^{th} reference pattern are superscripted with k.

5.3 Estimation Strategy for the Number of States

A strategy is proposed for estimating the number of states for each character (N_{sc}) and the number of training iterations of the model. The proposed strategy is illustrated in Figure 6. The number of states of each character (N_{sc}) is varied from one to six (number of training patterns). For each value of N_{sc}, the model parameters are re-estimated and validated on nine reference patterns (in which three patterns are unseen). In validation, three patterns with the least likelihood are selected and the average of the three likelihoods is returned as a result. The model that gives the best result on validation is selected as the final model.

In Figure 6, λ_{temp} denotes a temporary HMM which is created for each changed value of (N_{sc}), and it is compared with the best classifier yet (λ); if λ_{temp} is better than λ (based on the validation result) then λ is replaced with λ_{temp}. Here, $MAXN_{sc}$ represents the maximum number of states per observation (the number of reference patterns) and $MAXIteration$ represents the maximum number iterations of the training algorithm.

```
Input:    Training Data, Validation Data
          $T \leftarrow 32$
          $MAXN_{sc} \leftarrow 6, MAXIteration \leftarrow 15$

Step 1:   For $N_{sc} \leftarrow 1$ to $MAXN_{sc}$
              $\lambda_{temp} \leftarrow$ Initial Estimation of $A, \pi, \mu, \sigma$

Step 2:       For $Iteration \leftarrow 1$ to $MAXIteration$
                  Train $\lambda_{temp}$ with 1 iteration
                  Validate $\lambda_{temp}$
                  If ($\lambda_{temp}$ is the best classifier yet)
                      $\lambda \leftarrow \lambda_{temp}$
        End of Step 2
End of Step 1
```

Figure 6: Pseudo-code representing the Strategy for Estimating the Number of States and the Number of Training Iterations of the HMM

5.4 Adaptability

The training phase results in a set of HMMs $\lambda = \{\lambda_1, \lambda_2 ... \lambda_U\}$ where λ_i represents an HMM for the i^{th} registered user and the sub-script U represents the number of registered users. As each user has a distinct HMM: (i) adding/removing user(s) can be done without retraining the entire system, and (ii) when the typing pattern of a user changes over time, the model for that user can be updated by collecting new training patterns.

As shown in Figure 7, if there are U registered users in the system then the model of the $(U+1)^{th}$ user can be added (registered) to the system without affecting the models of the

U registered users. Similarly, for updating the model of the i^{th} registered user, the previous model of the i^{th} user is replaced with the updated model.

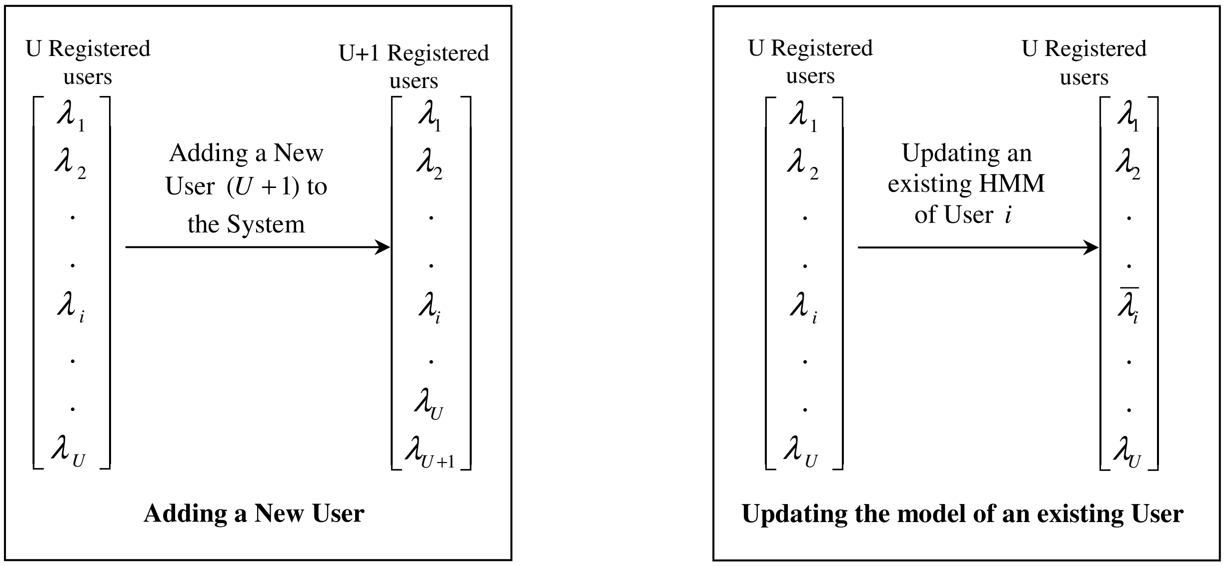

U Registered users
U+1 Registered users
Adding a New User $(U+1)$ to the System
λ_1 λ_2 λ_i λ_U λ_{U+1}
Adding a New User
U Registered users
U Registered users
Updating an existing HMM of User i
$\bar{\lambda}_i$
Updating the model of an existing User

Figure 7: Illustration of adding a new user and updating the model of an existing user in our user authentication system

6 USER AUTHENTICATION

In our method, authentication of a user is made in two stages: (i) in the user identification stage, a user is identified by searching through all the registered users, and (ii) in the user verification stage, a user is verified using the claimed identity. Thus, we can see that for user authentication our method has a one-to-many search instead of a one-to-one search. From Illustration 6 we can see that the user authentication system with a one-to-many search can have higher overall performance as compared to the system with a one-to-one search.

Illustration 6: *Consider a case where a registered user* (RU_1) *is trying to act as some other registered user* (RU_2) *where* $(RU_1 \neq RU_2)$*. If the search is one-to-one then only the template of the claimed user* (RU_2) *is used to verify the authenticity of the given keystroke pattern. But if the search is one-to-many then the template of the actual user* (RU_1) *is also considered (as in one-to-many search templates of all the users are considered), and the chances are that the actual user* (RU_1) *will be identified (as the given keystroke pattern is similar to the stored template of* RU_1 *rather than that of* RU_2*), which will result in a TN* *rather than an FAR. Hence, a one-to-many search helps in reducing the insider impostor attack and thereby improving the overall performance of an authentication system.*

We tested the performance of our method with the one-to-one search (user verification) and with the one-to-many search (user authentication) on 873 test patterns. Figure 8 illustrates the threshold independent average performance of both the methods using the ROC curve. We can see from Figure 8 that the user authentication with the one-to-many search has higher average performance than that of user verification with the one-to-one search.

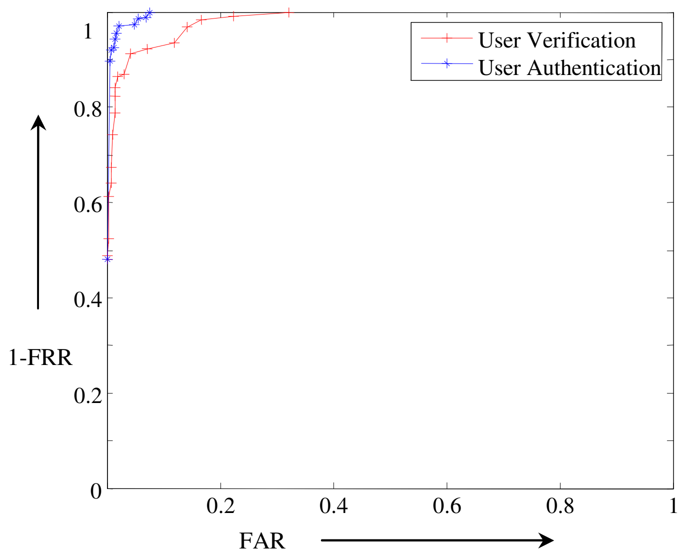


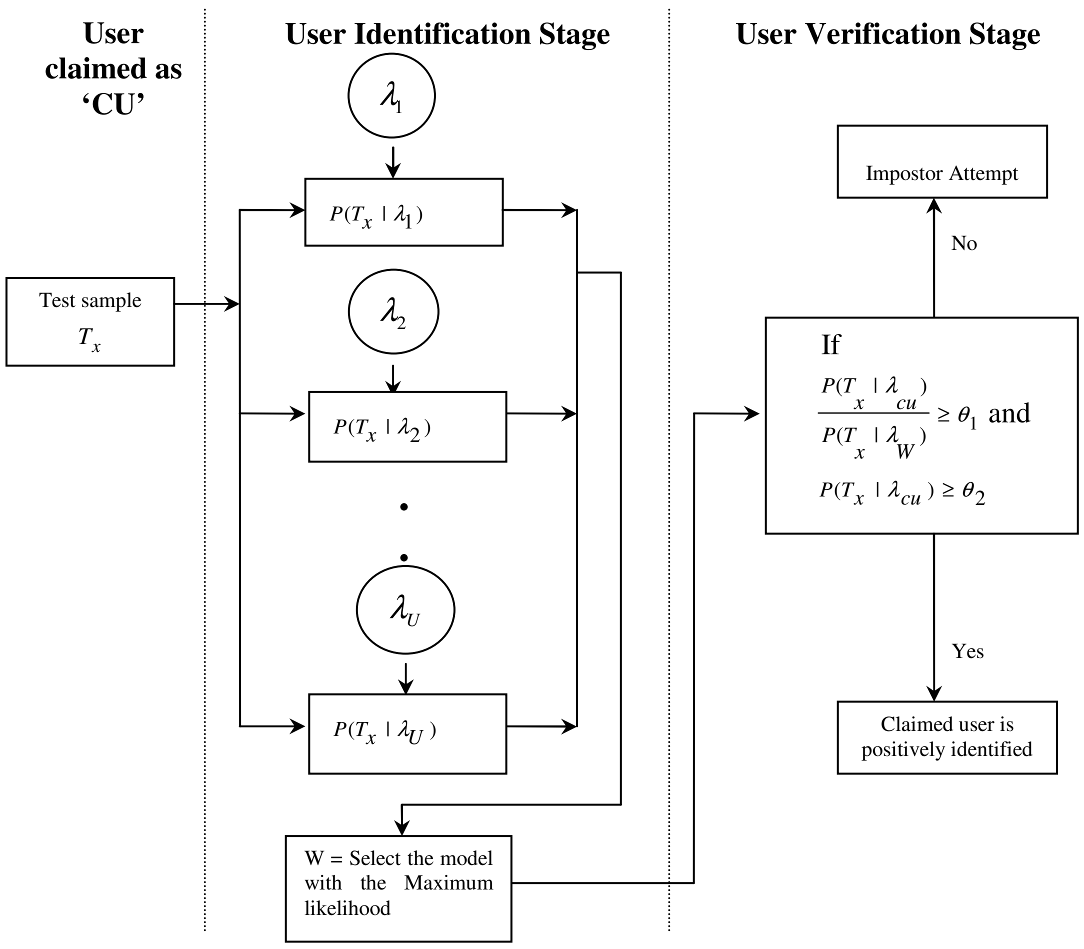
Figure 8: Comparison of User Authentication and User Verification techniques

6.1 The User Identification Stage

The procedure for user identification is illustrated in Figure 9. For the test sample T_x the likelihood of all the registered users ($P(T_x \mid \lambda_i)$ where $1 \le i \le U$) is calculated using the modified forward procedure (which is described in Section 4.4). The user with the highest likelihood is determined and is termed as the winner (represented with W) for the sample T_x .

6.2 The User Verification Stage

The procedure for user verification is illustrated in Figure 9. Test sample T_x is classified as a genuine attempt if the ratio between the likelihood scores of the claim, CU, ($P(T_x \mid \lambda_{cu})$) and the winner ($P(T_x \mid \lambda_W)$) is greater than a threshold θ_1 and the likelihood score of the claim is greater than a threshold θ_2; otherwise, it is classified as an impostor attempt where θ_1 and θ_2 can be adjusted for allowing trade-offs between FAR and FRR values.



User claimed as 'CU'
User Identification Stage
User Verification Stage
λ_1
$P(T_x \mid \lambda_1)$
Test sample T_x
λ_2
$P(T_x \mid \lambda_2)$
λ_U
$P(T_x \mid \lambda_U)$
W = Select the model with the Maximum likelihood
Impostor Attempt
No
If $\frac{P(T_x \mid \lambda_{cu})}{P(T_x \mid \lambda_W)} \geq \theta_1$ and $P(T_x \mid \lambda_{cu}) \geq \theta_2$
Yes
Claimed user is positively identified

Figure 9: Block Diagram of Computer User Authentication using the HMM

7 EXPERIMENTAL RESULTS

Training data was collected from a group of 43 users and each user provided a set of nine reference keystroke patterns for the reference string to set up an account. Our method is tested with 873 test patterns wherein the number of test patterns for each user varied from 0 to 102. Hence, in order to calculate the true error rates, we averaged the error rates over the number of users as opposed to the number of attempts [1].

FAR and FRR are the error rates used to evaluate the performance of the biometric classifiers [1]. The FAR and 1-FRR values obtained on 873 test patterns (with 43 registered users and 0 unregistered users) at different thresholds are shown in Table 1. Error rates at different thresholds are plotted in Figure 10. It shows that the FAR and FRR intersect at 3.04 % which is called as the equal error rate (EER). We can see from Table 1 and Figure 10 that when the FAR is less than 0.5 %, even with a slight change in threshold, there is a significant change in the FRR. Later when the FAR is greater than 0.5 %, there is no significant difference between the rates of increase in the FAR and decrease in the FRR (with a change in threshold).

FAR	0%	0.61%	0.75%	1.23%	1.51%	2.22%	4.87%	6.84%	7.69%
1-FRR	48.06%	89.6%	91.94%	92.5%	94.32%	97.05%	97.31%	98.96%	100%

Table 1: The FAR and (1-FRR) values when the method is tested on 873 keystroke patterns of 43 registered users

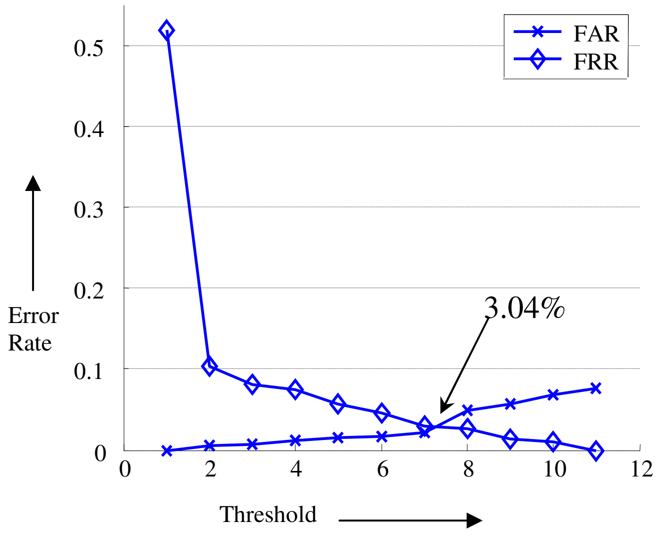


Figure 10: The FAR and FRR plotted on the same graph to determine the EER when the method is tested on 873 login attempts of 43 registered users

7.1 The Evaluation on Three Sets

For evaluating the effectiveness of our method, we created three data sets, by varying the number of registered/unregistered users, from the test data of 873 login attempts. Table 2 gives the detailed description of the three data sets. In Table 2, the test set of Set # 1 is the test data of 873 login attempts of 43 registered users. The test sets of Set # 2 and Set # 3 are the sub-sets of the test set of Set # 1; wherein, (1) the test set of Set # 2 has 807 login attempts of 36 registered and 7 unregistered users, and (2) the test set of Set # 3 has 561 login attempts of 25 registered and 18 unregistered users. In Table 2, valid attempt (VA) represents a login attempt made by a registered user to match his/her stored template and invalid attempt (IA) represents an attempt made by any (either registered or unregistered) user acting as some other registered user. The training set consist of six keystroke patterns for each registered user and the validation set consist of nine keystroke patterns for each registered user.

Set #	# RU	Training Set (6 Patterns per user)	Validation Set (9 Patterns per user)	Testing Set		
				# UU	# VA	# IA
1.	43	258	387	0	216	657
2.	36	216	324	7	186	621
3.	25	150	225	18	117	444

# RU – Number of Registered users
# IA – Number of Invalid Attempts
UU – Number of Unregistered users
VA – Number of Valid Attempts

Table 2: Description of the three sets which were created from the training set of 387 keystroke patterns and the test set of 873 keystroke patterns of 43 users where the number of patterns per user in training is six and in validation is nine

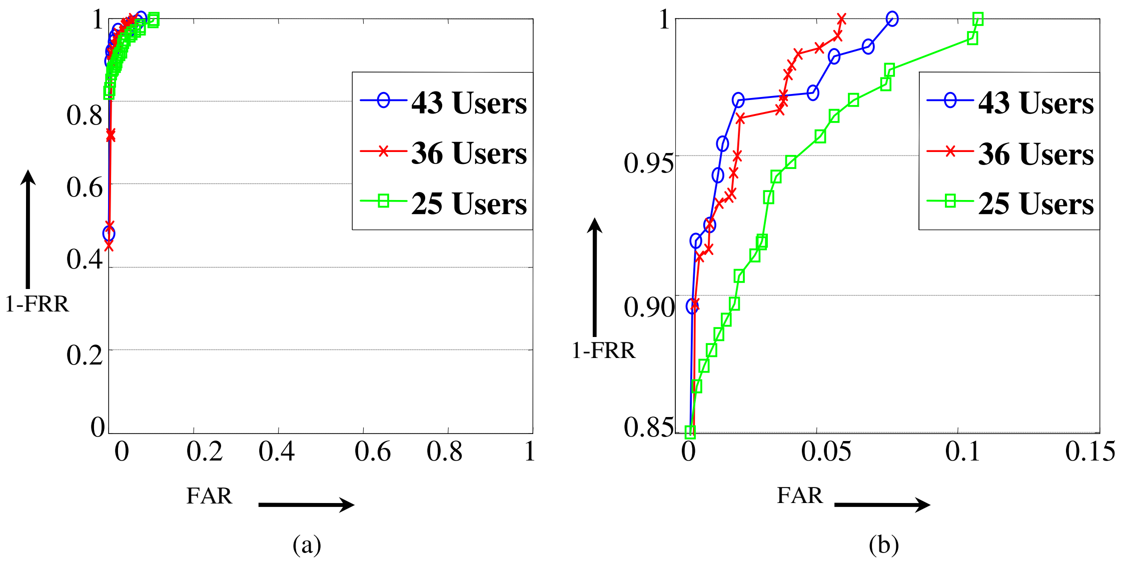


Figure 11: Evaluation of our method on the three test data sets (by varying the number of registered users in each set) using the ROC Curves

As ROC graphs are used for organizing classifiers and visualizing their performance [12], we have plotted our results for the three sets on the same ROC graph as shown in Figure 11 where Figure 11(b) represents a closer view of Figure 11(a). We can see that all three curves are very close to each other.

In addition to the ROC graph, Table 3 evaluates our method on the three tests sets where: (1) the first column represents the set number, (2) the second column represents the maximum accuracy obtained on each test set, (3) the third column represents the values of the intersection point (EER) of the FAR and the FRR for the three sets, (4) the fourth column represents the values of the FRR when the FAR is zero, (5) the fifth column represents the values of the FAR when the FRR is zero, and (6) the sixth column represents the AUC value [12] (which measures the threshold independent average performance of a classifier) obtained on each set. It can be seen from Table 3 that, (1) the maximum accuracy is almost the same for all the sets, (2) the EER value is slightly increased with the increase in the number of unregistered users, (3) the FRR at 'Zero FAR' is almost the same for the first two sets but it dropped significantly for the third set, and (4) there is not much difference between the FAR at 'Zero FRR' for the first two sets whereas for the third set it is slightly increased. In spite of these minor differences, the AUC value for all three sets is almost the same, which indicates that the performance of our method is not affected much with the change in the number of registered or unregistered users.

Set #	Maximum Accuracy	EER	FRR (at FAR=0)	FAR (at FRR=0)	AUC
1	97.69 %	3.04 %	51.94 %	7.69 %	0.99603
2	97.45 %	3.38 %	54.96 %	5.881 %	0.99458
3	96.84 %	4.67 %	17.93 %	10.72 %	0.99342

Table 3: Results of our method on the three test sets in terms of Maximum Accuracy, EER, Zero FAR, Zero FRR and AUC

7.2 Comparison with Existing User Authentication Methods

Most of the previous researchers developed user authentication systems using either user identification or user verification techniques. Our method combines both the techniques for authenticating users in order to improve the overall performance of the system. We can see from Illustration 6 and Figure 8 that the overall performance of the authentication system can be increased by combining both the identification and the verification techniques.

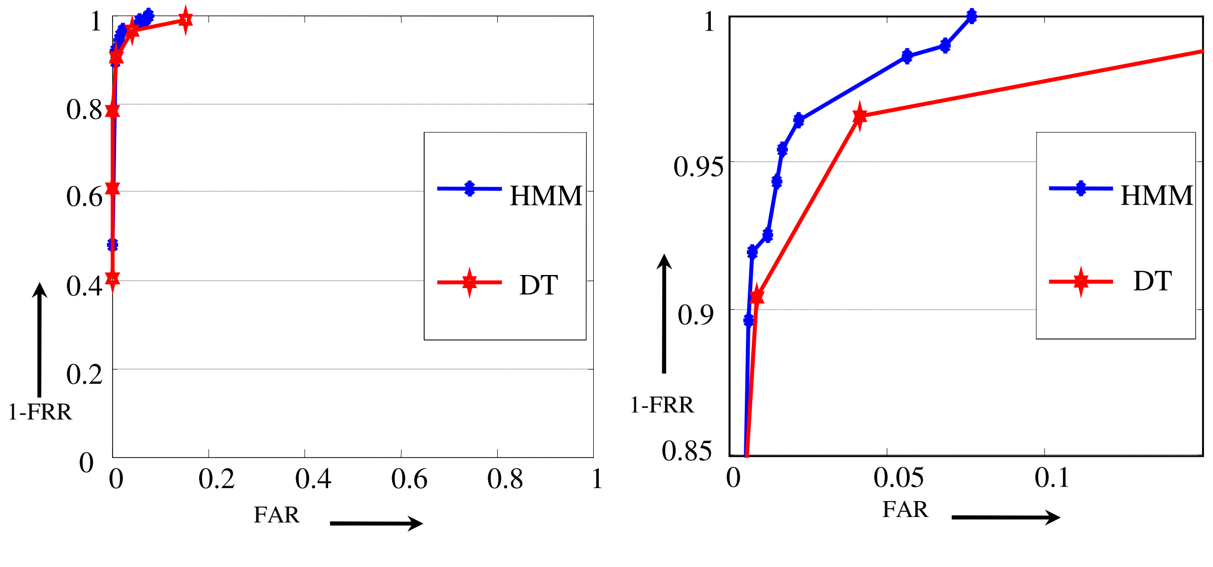
Most of the previous methods for user authentication cannot dynamically add or remove users without retraining the entire system such as neural network and clustering techniques [19-21, 23, 24]. But in our method, users can be dynamically added or deleted without retraining the entire system.

Joyce and Gupta [29] proposed a statistical method which is relatively simple and effective but Sheng et al. [33] tried the approach on our data set and obtained an FAR of 7.72 % when the FRR was 37.0 %, which are too large to be accepted. Sheng et al. [33] tested their method on the same data set (on which we tested our method) using decision trees (DT) and obtained the best results with wavelet transform. We have reproduced the results obtained by Sheng et al. [33] on 43 registered users in Table 4.

FAR	15.26	4.15	0.88	0.19	0.03	0.00	0.00	0.00
1-FRR	98.85	96.56	90.38	78.25	60.71	40.55	23.60	8.82

Table 4: The FAR and (1-FRR) values obtained by Sheng et al. [33] using wavelet transform

Results of our method (shown in Table 1) and that of the Sheng et al. [33] method (shown in Table 4) are plotted on the same ROC graph as shown in Figure 12. We can see from Figure 12 that the FAR is almost the same for both methods when the FRR is greater than 0.1; whereas, when the FRR is less than 0.1, increase in the FAR for the Sheng et al. [33] method is more than that of our method. The AUC value of the method proposed by Sheng et al. [33] is 0.98873 and the AUC value of our method is 0.99603.



(a) **(b)**

Figure 12: Comparison of our method (HMM) with the method proposed by Sheng et al. [33] (DT) using the ROC Curves

8 ANALYSIS

The computations involved in the forward and backward procedures are in the order of N^2T, where N is the number of states in an HMM (in our case it is $N_{sc}T$) and T represents the number of observation symbols in the observation sequence. As a result, the order of computations in terms of N_{sc} is in the order of $N_{sc}^2T^3$. But the order of computations involved in the modified forward procedure (or backward procedure) is in the order of N_{sc}^2T. Thus, the order of computations is reduced by T^2.

Each user has his/her specific model that assists in adding/deleting the user without retraining the entire system. Having a separate model for each user makes it convenient to update the system when the typing pattern of the users changes over time.

9 CONCLUSION AND FUTURE WORK

In this paper, we have proposed a method for computer user authentication using the HMM through keystroke dynamics by mapping the patterns of key hold times to speech signals. The HMM parameters are modified for reducing the order of computations (by T^2 where T represents the length of the keystroke pattern) involved in the forward and backward procedures of the HMM. The advantage of our method is the adaptability to the changing typing pattern of user(s) and addition/deletion of user(s) without retraining the entire system.

We tested our method on 873 test patterns and obtained the best FAR of 0.74 % when the FRR was 8.06 % and the AUC value was 0.99603. Furthermore, when we tested our method on three data sets by varying the number of registered/unregistered users, we observed that the average performance of the method did not change with a change in the number of registered/unregistered users. Our future work could include extending this work for continuous monitoring system.

APPENDIX

In this appendix, we determine the order of computations involved in the modified forward Procedure.

Modified Forward Procedure:

Initialization: $\overline{\alpha}_1(s) = \overline{\pi}_s \overline{b}_s(1, o_1)$, where $1 \le s \le N_{sc}$

Induction: $\overline{\alpha}_{t+1}(s) = \left[\sum_{r=1}^{N_{sc}} \overline{\alpha}_t(r)\overline{a}_{trs}\right]\overline{b}_s(t+1, o_{t+1})$, where $1 \le s \le N_{sc}, 1 \le t \le T-1$

Initialization step: For each value of s, there is a multiplication and zero additions. As s varies from 1 to N_{sc} there are N_{sc} multiplications and zero additions in the initialization step.

Induction step: In Induction step, for each s,t pair, there are $N_{sc}+1$ Multiplications involved in which N_{sc} multiplications are involved in $\left[\sum_{r=1}^{N_{sc}} \overline{\alpha}_t(r)\overline{a}_{trs}\right]$ and 1 multiplication with $\overline{b}_s(t+1, o_{t+1})$. As $1 \le s \le N_{sc}$, $1 \le t \le T-1$ the total number of multiplications in Induction step is $N_{sc}(N_{sc}+1)(T-1)$.

Similarly, for each s,t pair, there are $N_{sc}-1$ additions in $\left[\sum_{r=1}^{N_{sc}} \overline{\alpha}_t(r)\overline{a}_{trs}\right]$. Hence, the total number of additions in Induction step is $N_{sc}(N_{sc}-1)(T-1)$.

Thus, the total number of multiplications involved in the modified forward procedure is $N_{sc}(N_{sc}+1)(T-1)+N_{sc}$ and the total number of additions involved is $N_{sc}(N_{sc}-1)(T-1)$.

From this, we conclude that the order of computations involved in the forward procedure with the modified HMM parameters $N_{sc}^2 T$ …………...(1)

The order of computations involved in the forward procedure with the original HMM parameters are $N_{sc}^2 T^3$ (as $N = N_{sc}T$)..(2)

Hence from (1) and (2), we can see that the modified HMM parameters reduced the order of computations by T^2.

REFERENCES

1. A. J. Mansfield and Wayman, J.L. Best Practices in Testing and Reporting Performance of Biometric Devices, Center for Mathematics and Scientific Computing, National Physics Laboratory, Queens Road, Teddington, Middlesex, TW 11 0LW, August 2002.

2. Ajit V. Rao and Rose, K. Deterministically Annealed Design of Hidden Markov Model Speech Recognizers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *9* (2).

3. B. H. Juang and Rabiner, L.R. Hidden Markov Models for Speech Recognition. *American Statistical Association and the American Society for Quality Control*, *33* (3). 251-272.

4. Bakis, R. Continuous Speech Recognition via centisecond acoustic states. *Meeting of the Acoustic Society of America*.

5. Bartlow, N. Username and Password Verification through Keystroke Dynamics *Computer Science*, West Virginia University, Morgantown, West Virginia, 2005.

6. D. Bleha and Obaidat, M. Dimensionality reduction and feature extraction applications in identifying computer users. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, *21*. 452-456.

7. D. Umphress and Williams, G. Identity verification through keyboard characteristics. *Int. J. Man-Machine Studies*, *23* (3). 263-273.

8. Dainele Gunetti and C. Piccardi *Keystroke Analysis of Free Text*. *ACM Transactions on Information and System Security*, *8* (3). 312-347.

9. F. Monrose, M. K. Reiter and Wetzel, S., Password hardening based on keystroke dynamics. in *6th ACM Conference on Computer and Communication Security*, (1999), ACM Press, 73-82.

10. F. Monrose and Rubin, A., Authentication via keystroke dynamics. in *Proceedings of ACM Workshop*, (1997), ACM, 48-56.

11. Fabian Monrose and Rubin, A.D. Keystroke dynamics as a biometric for authentication. *Future Generation Computer System*, *16*. 351-359.

12. Fawcett, T. ROC Graphs: Notes and Practical Considerations for Researchers. *Kluwer Academic Publishers*.

13. J. Leggett, G. Williams, M. Usnick and Longnecker, M. Dynamic identity verification via keystroke characteristics. *International Journal of Man-Machine Studies* (35). 859-870.

14. J. Leggett and Williams, G. Verifying identity via keystroke characteristics. *International Journal of Man-Machine Studies*, *28* (1). 67-76.

15. Jafar Adibi, Wei-Min Shen and Noorbakhsh, E. Self-Similarity for Data Mining and Predictive Modeling: A Case Study for Network Data. *PAKDD*.

16. John A. Robinson, Vicky M. Liang, J. A. Michael Chambers and Mackenzie, C.L. Computer user verification using login string keystroke dynamics. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans*, *28* (2). 236-241.

17. Kenji Kita, Takeshi Kawabata and Saito, H. HMM Continuous Speech Recognition using Predictive LR Parsing. *Acoustics, Speech, and Signal Processing ICASSP-89.*, *2*. 703-706.

18. Lawrence Rabiner and Juang, B.H. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

19. Lin, D.T., Computer Access Authentication with neural network based keystroke identity verification. in *International Conference on Neural Networks*, (Houston, Texas, 1997), 174-178.

20. M. K. Reiter, F. Monrose and Wetzel, S., Password hardening based on keystroke dynamics. in *6th ACM Conference on Computer and Communications Security*, (Singapore, 1999), ACM, 73-82.

21. M. S. Obaidat and Macchairolo, D.T. A Multilayer Neural Network System for Computer Access Security. *IEEE Transactions on Systems, Man and Cybernetics*, *24* (5).

22. M. S. Obaidat and Sadoun, B. Verification of Computer Users Using Keystroke Dynamics. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, *27* (2). 261-269.

23. Marcus Brown and Rogers, S.J. User Identification via Keystroke Characteristics of typed names using Neural Networks. *International Journal of Man-Machine Studies*, *39*. 999-1014.

24. Obaidat, M.S., A verification methodology for computer systems users. in *Proceedings of the 1995 ACM Symposium on Applied Computing*, (Nashville, Tennessee, 1995), ACM Press, 258-262.

25. Paul, D.B. Speech recognition using hidden Markov models. *Lincoln Lab. Journal*, *3* (1). 41-62.

26. Phoha, V.V., Kumar, P. and Vuyyuru, S.K. Keystroke Typing Rhythm as an Input Vector for Authentication. *Manuscript under revision for submission to IEEE Transactions on Systems,Man and Cybernetics, Part B*.

27. R. Gaines, W. Lisowski, S. Press and Shapiro, N. Authentication by Keystroke Timing: Some Preliminary Results *Tech. report R-256-NSF*, Tech report R-256-NSF, RAND, Santa Monica, CA, 1980.

28. Rabiner, L. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*.

29. Rick Joyce and Gupta, G. Identity Authentication Based on Keystroke Latencies. *Communications of the ACM*, *33* (2). 168-176.

30. S. Bleha, C. Slivinsky and Hussein, B. Computer-access security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12* (12). 1217-1222.

31. Shrijit S. Joshi and Phoha, V.V., Investigating Hidden Markov Models capabilities in Anomaly Detection. in *43rd ACM SE Conference*, (Kennesaw State University, Kennesaw, Georgia, USA, March 2005), ACM.

32. Vinar, T. Enhancements to Hidden Markov Models for Gene Finding and Other Biological Applications *Computer Science*, University of Waterloo, Waterloo, Ontario, Canada, 2005, 162.

33. Yong Sheng, Vir V. Phoha and Rovnyak, S.M. A Parallel Decision Tree-Based Method for User Authentication Based on Keystroke Patterns. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, *35* (4). 826-833.