A Parallel Decision Tree-Based Method for User Authentication Based on Keystroke Patterns

Yong Sheng, *Senior Member, IEEE*, Vir V. Phoha, *Senior Member, IEEE*, and Steven M. Rovnyak, *Member, IEEE*

***Abstract*—We propose a Monte Carlo approach to attain sufficient training data, a splitting method to improve effectiveness, and a system composed of parallel decision trees (DTs) to authenticate users based on keystroke patterns. For each user, approximately 19 times as much simulated data was generated to complement the 387 vectors of raw data. The training set, including raw and simulated data, is split into four subsets. For each subset, wavelet transforms are performed to obtain a total of eight training subsets for each user. Eight DTs are thus trained using the eight subsets. A parallel DT is constructed for each user, which contains all eight DTs with a criterion for its output that it authenticates the user if at least three DTs do so; otherwise it rejects the user. Training and testing data were collected from 43 users who typed the exact same string of length 37 nine consecutive times to provide data for training purposes. The users typed the same string at various times over a period from November through December 2002 to provide test data. The average false reject rate was 9.62% and the average false accept rate was 0.88%.**

***Index Terms*—Biometrics, computer security, keystroke patterns, Monte Carlo methods, network security, parallel decision trees, simulations, wavelet analysis.**

I. Introduction

GREAT efforts have been expended on the security of computer systems and networks, such as improving hardware and software reliability, preventing the proliferation of viruses, developing more secure access control, etc. One of the most active fields in computer security research is developing more secure authentication methods for user access.

Currently, there are three different types of authentication methods: those involving objects, knowledge, and biometrics. Of these, a biometric approach is the most secure and convenient authentication tool. Sometimes, the biometric method is split into two subcategories: those involving actions and those involving physiology [1]. Since most biometrics need specially designed devices to measure users' unique physical or behavioral characteristics, the authentication based on keystroke patterns becomes the cheapest.

Any research on identity authentication using keystroke patterns is based on the argument that for a regularly typed string, the user's characteristics shown in the form of keystroke press and release times can be quite consistent and unique [2]–[6]. There have been many attempts to apply statistical and some artificial intelligence techniques to authentication using keystroke patterns. However, the problem is still open, since no one has proposed a well-accepted approach. The difficulty is due to the dilemma of satisfying and/or balancing the following factors:

- *Effectiveness.* Two criteria are generally used to gauge the effectiveness of an approach [7]. One is called false accept rate (FAR) which is how often a wrong user is authenticated. The other is called false reject rate (FRR) which is how often the legitimate user is not accepted. In practice, FRR tends to increase when one tries to decrease FAR, and vice versa.
- *Efficiency.* This metric indicates whether the method is computationally expensive. The complexity of the algorithm is usually calculated for this purpose.
- A*daptability and robustness.* Since almost all research has used keystroke patterns collected during a fixed training period, it remains a challenge to make the method adaptive to changing typing patterns. The method also needs to be adaptive to turnover of users and to diversity of users, such as clerical professionals, handymen, etc.
- *Convenience.* The approach should make users feel comfortable. It will not be widely accepted if the user needs to type a lengthy string, memorize something difficult, or type a short string many times.

[1], [2], [4], [8]–[10] have used keystroke patterns collected over a period of time to authenticate users. Almost all of these works use statistical techniques or calculation of variations for user authentication. The work in [8] and [9] is for commercial purposes and sufficient data is not readily available from their studies. [10] proposes a method that uses partitions to split data into cluster domains based on the typing speed. Then, a reference profile for each user in its domain is built. Finally, it implements an optimized classifier based on probabilistic measures with an acceptance of authentic rate of 90%, which is equivalent to an FRR of 10%. The work in [1] appears simple, and the effectiveness of their approach is quite impressive. The authors first build a mean reference signature for each user based on eight sets of his keystroke patterns consisting of username, password, first name, and last name. Then the norm of the difference between any test signature and the mean reference signature is calculated to determine if the user is legal based on a predefined threshold. They reach an FRR of 16.67% and an FAR of 0.25%.

Manuscript received April 13, 2004; revised October 18, 2004. This work was supported in part by the Army Research Office under Grant DAAD 19-01-1-0646 and by Louisiana BoR under Grant LEQSF(2003-05)-RD-A-17. This paper was recommended by Associate Editor V. Govindaraju.

Y. Sheng and S. M. Rovnyak are with the Department of Electrical and Computer Engineering, Indiana University—Purdue University Indianapolis (IUPUI), Indianapolis, IN 46202-5132 USA (e-mail: yosheng@iupui.edu; srovnyak@iupui.edu).

V. V. Phoha is with Louisiana Tech University, Ruston, LA 71272 USA (e-mail: phoha@latech.edu).

Digital Object Identifier 10.1109/TSMCB.2005.846648

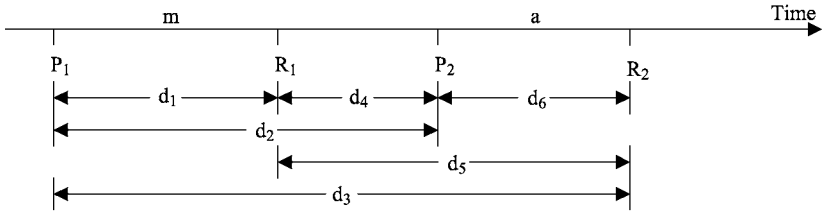

Fig. 1. Timestamps and keystroke feature representation.

[11] uses fuzzy algorithms to develop computer access security systems. [12] uses back-propagation models for weight updates in neural networks (NNs) to train from sample data and then classify a user. [13] compares the performance of almost all widely used NN and pattern recognition algorithms, including fuzzy ARTMAP, radial basis function networks, learning vector quantization NN paradigms, back-propagation with a sigmoid transfer function, hybrid sum-of-products, sum-of-products, potential function, and Bayes' rule algorithms. [10] points out the problem of retraining every time a new user is introduced, and the possibility that such retraining requirements may be prohibitively expensive. Some approaches divided the database into smaller groups of subjects and then retrained the network.

Most of the above work is for static authentication, similar to logging in at the start of a session. [5] proposes a dynamic method which can authenticate users over a period of typing. This paper only concentrates on static authentication. Most of the previous work applies a combination of biometrics and knowledge, implying that username/password is still a part of the authentication.

We propose a parallel decision tree (DT)-based method which requires the user to pick his user name from an available user list and to type a common short string for authentication. Each user needs to provide nine samples of the same common string for training purposes. Additional simulated data is generated from the nine samples based on a well-accepted assumption that a Gaussian distribution governs the features in the keystroke patterns. For each user, eight DTs are trained using real and simulated data including features derived from wavelet transforms. Once the training is done, a parallel DT for each user is constructed using those eight DTs. The criterion for the parallel DT to authenticate a user is that at least three of its DTs approve.

The novelty of this paper is the combination of a Monte Carlo (MC) method to generate additional training data together with parallel DTs to solve an important biometric problem. Because separate DTs are constructed for each user, it may be possible to add new users without reprogramming the entire authentication system after the database has become sufficiently large. Commercially available DT construction algorithms are fast and they have a parameter to balance the tradeoff between FAR versus FRR. The parallel DT method provides another mechanism for adjusting the FAR versus FRR. The use of wavelet analysis for feature extraction is shown to increase performance comparable to the use of Fourier analysis. Experimental results in this study are promising compared with other techniques in the literature. The practical characteristics of the approach and the promising experimental results suggest that the method could potentially be used for other biometric authentication systems.

II. Data Collection and Processing

For any research involving pattern recognition techniques, one should conduct an experiment to collect two sets of data. One is for training or reference, and the other is for testing or verification. From November to December 2002, an experiment was conducted to collect reference and verification data from 43 users, mainly graduate students. Many of the students were from foreign countries and not all of them knew the purpose of the experiment. All participants had been using computers for at least three years and some for over ten years.

Given a common string, namely, "master of science in computer science", each participant was required to provide nine sets of reference keystroke patterns to set up his account. However, since users were only asked to provide verification data at their own convenience, the number of verification keystroke patterns provided by each user varied from 0 to 102 with a total of 873 sets.

To collect data, a program with a friendly user interface, coded in Microsoft Foundation Classes (MFC), was provided to users. One can practice the common string as many times as desired to get some degree of familiarity. Once the user decided to provide his reference keystroke patterns, he was required to type it nine consecutive times. This is more practicable than collecting reference patterns over a period of time, since an authentication system is unlikely to be accepted in the real world if it requires a few weeks to set up an account for a user.

Note that no capital letters were used in the string, since a user's typing habits of capital letters varies. One may prefer right-shift, left-shift, or caps lock to type letters in uppercases which might lead to different keystroke sequences, whereas the method requires all users to provide the same sequences of keystrokes. Pressing "Delete" or "Backspace" forced the user to type the string from the beginning.

For any event during the typing sequence, the MFC program recorded its timestamp, such as key press time and key release time. For any two consecutive letters typed, there are four timestamps associated with key press time of the first key (P_1), key release time of the first key (R_1), key press time of the second key (P_2), and key release time of the second key (R_2). The features of these two keystrokes can be represented by all six combinations of time differences of four timestamps, namely, $\mathrm{d}_1 : \mathrm{R}_1 - \mathrm{P}_1$, $\mathrm{d}_2 : \mathrm{P}_2 - \mathrm{P}_1$, $\mathrm{d}_3 : \mathrm{R}_2 - \mathrm{P}_1$, $\mathrm{d}_4 : \mathrm{P}_2 - \mathrm{R}_1$, $\mathrm{d}_5 : \mathrm{R}_2 - \mathrm{R}_1$, and $\mathrm{d}_6 : \mathrm{R}_2 - \mathrm{P}_2$, as shown in Fig. 1 using an example of "ma".

Some features have well-known names, for instance, d_1 and d_6 are called key press times, d_2 is called latency of key press time, d_5 is called latency of key release time, etc. Note that d_4 can be negative if the user pressed the second key before he

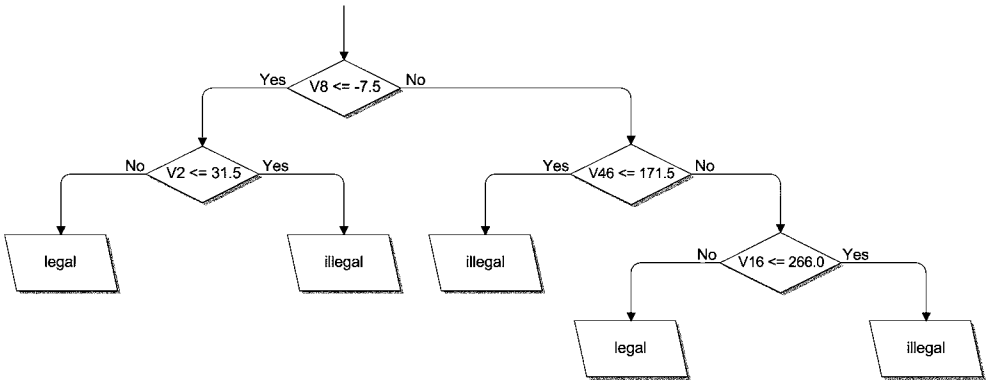

Fig. 2. Structure of a DT for the 24th user (trained on raw data only).

released the first one. Of six features, only three are independent, one can figure out the remaining three by applying addition or subtraction operations. In this paper, only d_1, d_4, and d_6 are used as independent features in pattern recognition, since our initial trials showed that including more dependent features could not significantly improve the performance of the proposed system. Moreover, not all d_1's, d_4's, and d_6's were considered. Spaces between words were discarded, as the user may pause for recollection of what has to be typed next [3], [4].

A vector was built for each set of keystroke patterns, which includes 32 key press times and 26 time-intervals between two consecutive keys. For the common string used in the experiment, the vector was constructed as below:

$$\begin{aligned}\mathrm{V} = [&\mathrm{t_m}\ \mathrm{t_{ma}}\ \mathrm{t_a}\ \mathrm{t_{as}}\ \mathrm{t_s}\ \mathrm{t_{st}}\ \mathrm{t_t}\ \mathrm{t_{te}}\ \mathrm{t_e}\ \mathrm{t_{er}}\ \mathrm{t_r}\ \mathrm{t_o}\ \mathrm{t_{of}}\ \mathrm{t_f}\ \mathrm{t_s}\ \mathrm{t_{sc}}\ \mathrm{t_c}\\ &\mathrm{t_{ci}}\ \mathrm{t_i}\ \mathrm{t_{ie}}\ \mathrm{t_e}\ \mathrm{t_{en}}\ \mathrm{t_n}\ \mathrm{t_{nc}}\ \mathrm{t_c}\ \mathrm{t_{ce}}\ \mathrm{t_e}\ \mathrm{t_i}\ \mathrm{t_{in}}\ \mathrm{t_n}\ \mathrm{t_c}\ \mathrm{t_{co}}\ \mathrm{t_o}\ \mathrm{t_{om}}\\ &\mathrm{t_m}\ \mathrm{t_{mp}}\ \mathrm{t_p}\ \mathrm{t_{pu}}\ \mathrm{t_u}\ \mathrm{t_{ut}}\ \mathrm{t_t}\ \mathrm{t_{te}}\ \mathrm{t_e}\ \mathrm{t_{er}}\ \mathrm{t_r}\ \mathrm{t_s}\ \mathrm{t_{sc}}\ \mathrm{t_c}\ \mathrm{t_{ci}}\ \mathrm{t_i}\\ &\mathrm{t_{ie}}\ \mathrm{t_e}\ \mathrm{t_{en}}\ \mathrm{t_n}\ \mathrm{t_{nc}}\ \mathrm{t_c}\ \mathrm{t_{ce}}\ \mathrm{t_e}]^{\mathrm{t}}\end{aligned}$$

where the entries with one-letter subscripts represent key press times corresponding to d_1 or d_6. Entries with two-letter subscripts represent time-intervals between successive key press times similar to d_4. To distinguish entries with the same subscript, $\mathrm{v}_1, \mathrm{v}_2, \ldots$, and v_{58} are introduced to represent entries $1, 2, \ldots$, and 58, respectively. Since each entry represents a feature, the total number of features used to represent a user's keystroke pattern is 58, and each user had nine reference vectors plus a variable number of verification vectors.

III. Pattern Recognition and Decision Trees

Since the method in [1] is relatively simple but effective, we tried their approach. Using our reference vector in place of their reference signature, we obtained an FRR of 37.0% and an FAR of 7.72%, which are too large to be accepted. Apparently, our vectors containing keystroke features of a common string do not contain as much information as their signatures, which include keystroke latencies of username, password, first name, and last name. Thus, their approach cannot be borrowed to solve our problem. The approach we propose uses DTs and wavelet transforms.

DTs are a type of learn-by-example pattern recognition technique and have been applied to many data mining problems. Artificial NNs could also be used for a pattern recognition tool, and are more general than DTs. NNs can associate their input vectors with a continuous range of output values, whereas DTs are only suited for classification problems having a small number of output categories such as legal/illegal. However, for problems applicable to both, our earlier research showed that DTs usually require much less computation time for the training than NNs do [14]. Even though this conclusion was drawn from research on an electrical engineering problem, the computational efficiency is a property that seems to generalize across domains. This efficiency is helpful for the present application, which is very computation intensive in the training procedure. The on-line processing time, which corresponds to the time needed to verify a user, of either an NN or a DT is practically fixed and negligible.

A DT for a particular user was trained to recognize him as the only legal user and all other users as illegal. Thus, only nine raw cases out of 387 in each training data set have the desired output as legal. A case refers to an input-output pair, which contains an input vector along with the desired output, which is the output the classifier is supposed to assign this input vector. To quantify outputs, we let 1 indicate that the desired output for a case is *legal* and let 0 indicate that the desired output for a case is *illegal*. Each DT has a test set of data including 873 cases. The number of cases having the desired output as 1 varies from one user to another. The test cases for a user's DT having 1 as the desired output included all cases he provided for verification regardless of the username he selected at the time. All the remaining cases, namely, those provided by others for verification, were assigned a desired output of 0 regardless of the username they selected at the time. In actual implementation, the user will have to select his own name in order to specify the DT that is trained to recognize his pattern with an expected FRR of 9.62%. If he selects another username, then he can expect a rejection rate of $100\% - 0.88\% = 99.12\%$.

The commercial DT software CART was used for training DTs [15]. The CART software allows the user to specify some parameters, such as relative misclassification costs and complexity costs. The former allows the user to specify, for ex-

ample, that it is two times worse to classify an illegal user as a legal one than to make an error in the other direction. The complexity cost allows the user to prevent DTs from having too many nodes and over-fitting the training data. A classifier that simply memorizes the data will probably not work well on new test data. Using default parameters in training, we obtained an FRR of 36.7% and an FAR of 3.25%. No significant improvement was observed while trying different values of relative misclassification or complexity costs. Fig. 2 shows a DT obtained for the 24th user containing four nodes and five leaves. To make them meaningful, we converted outputs of 0's and 1's into "illegal" and "legal", respectively. Note that the unit of values is ms ($1\ \mathrm{ms} = 10^{-3}$ second) and a negative value indicates that a key was pressed before its previous key was released.

IV. Parallel Decision Trees

When a single DT solution cannot reach a certain level of effectiveness for some problems, one can try parallel DTs. For classification problems having two output categories like ours, an effective parallel DT solution is based on probability theory.

Assume that the length of the input vector for the DT is n, and that the output has two categories. The performance of the DT is determined by the rate of success, p, which is defined as the number of successfully classified cases divided by the total number of cases. Since the output is either success or failure, the rate of failure, q, is equal to $1 - \mathrm{p}$. If the performance of a DT is unacceptable, one can simply split the input vector of length n into two subvectors each having length $n/2$ and train two DTs. Given the rates of success of these two DTs, p_1 and p_2, respectively, and the assumption that these two subvectors are independent, one can figure the success rate of the parallel DT as $\mathrm{p}' = 1-(1-\mathrm{p}_1)\cdot(1-\mathrm{p}_2)$, if either subtree's approval means an approval for the parallel DT. If n is not too small, then neither p_1 nor p_2 will be much less than p. Thus p' can be greater than p. A very simple example can illustrate this property. Assume $\mathrm{p} = 0.8$, $\mathrm{p}_1 = 0.6 < \mathrm{p}$, and $\mathrm{p}_2 = 0.7 < \mathrm{p}$, then $\mathrm{p}' = 1 - (1 - 0.6) \cdot (1 - 0.7) = 0.88 > \mathrm{p}$. Similarly, one can try splitting the original vector into four subvectors to construct a parallel DT containing four DTs with a success rate of $\mathrm{p}'' = 1 - \prod_{i=1}^{4}(1 - p_i)$, where $\mathrm{p_i}$ is the success rate of the i^{th} subvector.

For the present application, we need to consider both FRR and FAR. Almost all previous researchers admit that it is not easy to improve one while holding the other unchanged let alone to improve both simultaneously. A method may be judged acceptable if it can improve one rate a lot while sacrificing the other only slightly.

We assume that the length of the original vector is n and that it has been split into m subvectors of equal length to train separate DTs. Let $\mathrm{p_i}$ denote the i^{th} DT's FRR and $\mathrm{q_i}$ denote its FAR, where i ranges from 1 to m. Of m DTs, it is very common that some DTs output 1, and the others output 0. For this case, it is necessary to combine these outputs into a single decision. Let the parallel DT output 1 if at least j of m DT's output 1, where $1 \le j \le m$. For the parallel DT constructed by those m DTs, its theoretical FRR and FAR can be calculated using the following equations:

$$\begin{aligned}\mathrm{FRR} &= 1 - \sum_{i=j}^{m} \sum_{C_m^i} \underbrace{(1-p_{k_1})(1-p_{k_2})(1-p_{k_3})\ldots(1-p_{k_i})}_{i} \\ &\quad \times \underbrace{p_{l_1} p_{l_2} p_{l_3} \cdots p_{l_{m-i}}}_{m-i} \\ \mathrm{FAR} &= \sum_{i=j}^{m} \sum_{C_m^i} \underbrace{q_{k_1} q_{k_2} q_{k_3} \cdots q_{k_i}}_{i} \\ &\quad \times \underbrace{(1-q_{l_1})(1-q_{l_2})(1-q_{l_3})\ldots\left(1-q_{l_{m-i}}\right)}_{m-i} \end{aligned} \tag{1}$$

where, in each equation, the second sum sign is only associated with the number of additions needed to be performed which is C_m^i or $\binom{m}{i}$. Here it denotes the sum adds all combinations together. For example, assume that a parallel DT is constructed by four DTs with FRRs of p_1, p_2, p_3, p_4, and FARs of q_1, q_2, q_3, q_4, respectively. Moreover, if we set the criterion that at least one DT output of 1 will result in approval from the parallel DT, then its theoretical FRR and FAR can be calculated

$$\begin{aligned}\mathrm{FRR} &= 1 - \sum_{i=1}^{4} \sum_{C_4^1} (1 - p_k) p_{l_1} p_{l_2} p_{l_3} \\ &= 1 - \sum_{i=1}^{4} [(1-p_1)p_2p_3p_4 + (1-p_2)p_1p_3p_4 \\ &\qquad + (1-p_3)p_1p_2p_4 + (1-p_4)p_2p_3p_4] \\ \mathrm{FAR} &= \sum_{i=1}^{4} \sum_{C_4^1} q_{k_1} q_{k_2} q_{k_3} (1 - q_l) \\ &= \sum_{i=1}^{4} [q_1q_2q_3(1-q_4) + q_1q_2q_4(1-q_3) \\ &\qquad + q_1q_3q_4(1-q_2) + q_2q_3q_4(1-q_1)]. \end{aligned} \tag{2}$$

Theoretically, splitting approaches can be applied many times. Thus, it seems that the classifier's performance can be improved steadily. However, more splitting means more training, and the approach may not be accepted if the computation is too expensive. Moreover, the performance could not be improved infinitely, since the assumptions for splitting methods eventually do not hold when the length of the subvector is small enough. For instance, a subvector may only contain less classifiable features, the independence of subvectors may no longer hold, etc. Therefore, in this application, the splitting methods were only allowed to be applied up to three times, which means at most eight subvectors can be obtained from splitting the vector with length 58. Since the quotient of dividing 58 by 8 is not an integer, the proposed methodology splits it into eight subvectors of roughly equal length.

Another important factor which needs to be considered is j, which defines the minimum number of DTs required to output 1 in order for the parallel DT to approve a user. By reviewing previous research, we found that most proposed systems have larger FRRs compared with their FARs. For instance, the method in [1] reaches an FRR of 16.67% and an FAR of 0.25%; the approach

in [6] obtains an FRR of 8.1% and an FAR of 2.8%; etc. Because our biometric approach could be combined with requiring a user to remember a username, thus providing additional security, our goal is to lower FRR while sacrificing a little bit of FAR. Assuming the individual DTs have balanced FRR and FAR, we can examine the implications of different thresholds for splitting the feature vector one to three times.

- For two subvectors, the parallel DT will have balanced FRR and FAR for $\mathrm{j} = 1$. Since j cannot be chosen any lower than this number, we select $j = 1$.
- For four subvectors, the parallel DTs will have balanced FRR and FAR when $\mathrm{j} = 2$. We select the next smaller threshold, which is $\mathrm{j} = (4/2) - 1 = 1$.
- For eight subvectors, the parallel DTs will have balanced FRR and FAR when $\mathrm{j} = 4$. We select the next smaller threshold, which is $\mathrm{j} = (8/2) - 1 = 3$.

In summary, we chose $j = \max\{(m/2) - 1, 1\}$, where m is the number of subvectors. Table I shows FRRs and FARs for splitting the feature vector one to three times. Performance was averaged over all users in order to make the evaluations meaningful.

Observation of these FRRs and FARs shows that the convergence of them is not as good as we expected. It seems that using a parallel DT approach alone cannot solve our problem. Consequently, we introduced an effective method for training set enhancement that is not new but has never been used in this area.

TABLE I
COMPARISON OF FRRS AND FARS FOR ONE TO THREE SPLITTINGS

	j	FRR	FAR
Two DTs	1	13.6%	8.86%
Four DTs	1	3.21%	19.9%
Eight DTs	3	6.53%	5.86%

V. MONTE CARLO (MC) METHODS AND SIMULATIONS

The MC method is a widely used techniques in engineering and science problems. Generally speaking, any research involving the use of random numbers can be considered "Monte Carlo". MC methods have been used in computer science for applications such as network traffic studies, parallel and distributed computing, signal and image processing, etc. One major application of MC methods is generating random cases in simulation studies based on known/unknown probabilistic distributions.

Since collecting more data may result in unacceptable inconvenience or expense, or it may simply be unrealistic or impossible, the effectiveness of pattern recognition solutions to many problems can be limited by insufficient data. In our research, nine sets of keystroke patterns were collected for training purposes. Our system based solely on these data as described in the previous section shows performance which could not match the results of other researchers. One of the simplest ways to improve performance is by asking users to provide more keystroke patterns for training purposes. However, doing so is inconvenient. Moreover, the number of sets needed for a particular performance goal may reach an unacceptable or unrealistic level, such as tens or hundreds of sets of data for each user. The method in [16] requires an average number of 223 training patterns from each user. [6]' approach uses 30 sets of reference data. Thus, we attempted to increase performance by generating a certain amount of simulated keystroke patterns for each user.

Many previous researchers used the assumption that keystroke features are governed by a Gaussian (normal) distribution either explicitly or implicitly in their research [3], [5], [6], [10]. Generating simulated vectors each containing 58 features is also based on this assumption. For each user, a mean reference vector, which is similar to the mean reference signature proposed by [1], and a standard deviation reference vector are first calculated. The approach, which is also similar to the method for removal of outliers proposed by [1], includes the following steps: First, the mean and standard deviation are calculated for sample values of the features representing each substring. Then, for each feature, the mean is compared with the nine values of that feature and any outliers defined as the datum whose value is beyond the mean $\pm$ three standard deviations are discarded. Finally, for each feature, the mean and standard deviation of the remaining values are calculated. This process is repeated for each user to produce 43 pairs of mean reference vectors and standard deviation vectors.

For each user, simulated data are then generated for each feature based on the mean and standard deviation of that feature. Since most general-purpose programming languages only provide uniform random numbers, further transformation is needed to produce Gaussian random numbers. Given a pair of independent random numbers uniformly distributed in the interval of [0,1), say x_1 and x_2, two Gaussian random numbers with mean of μ and standard deviation of σ, say y_1 and y_2, can be calculated using

$$\begin{aligned} y_1 &= \mu + \sqrt{-2\sigma^2 \ln(1 - x_1)} cos(2\pi x_2) \\ y_2 &= \mu + \sqrt{-2\sigma^2 \ln(1 - x_1)} sin(2\pi x_2) \end{aligned} \tag{3}$$

where ln represents the natural logarithm, which is the logarithm having base e.

For each user, two random values are thus generated for each feature. All the first random values for each feature were collected into one vector and all the second random values into another vector. Either of the vectors is called a simulated vector to distinguish it from the nine reference vectors which are provided by the user. By generating more Gaussian random numbers, a certain number of simulated vectors are therefore created for a user.

The performance of DTs is sensitive to the number of cases representing each class. For example, a DT trained with a set including 50 cases of output 1 and 50 cases of output 0 will have a lower FRR than a DT trained with a set including 10 cases of output 1 and 90 cases of output 0. When training a DT for a particular user, the cases representing his typing have the desired output set to legal whereas the desired output for cases representing all other users are set to illegal. Without additional simulated data, the cases representing illegal users for constructing an individual DT greatly outnumber the cases representing the legal user. In order to balance the number of cases representing

TABLE II
FRRs AND FARs OBTAINED FROM MONTE CARLO AND PARALLEL DT METHODS

	j	FRR	FAR
Two DTs	1	12.7% (12.1%)	2.78% (2.78%)
Four DTs	1	4.70% (2.58%)	7.66% (8.05%)
Eight DTs	3	9.16% (4.82%)	1.44% (0.36%)

Note: Values in parentheses are theoretical ones.

each class, the DT training set for each user includes his nine reference cases and 3771 simulated cases with output 1 plus 3780 cases with output 0. The cases with output 0 are from the other 42 users each contributing nine reference cases and 81 simulated cases. Thus, 7560 cases were used to train each user's DT.

Up to a point at which overtraining begins to occur, the performance of a DT is somewhat correlated with the number of nodes in it. Having more nodes means the DT extracts more features in training, and hence attains better performance. For example, the 24th user's DT trained using both reference cases and simulated cases has 24 nodes and 25 leaves which is too large to display here. It is much more complicated than the one shown in Fig. 2, which is only trained with reference cases.

Table II shows the results from the combination of MC and parallel DT approaches. Compared with the pure parallel DT method, a parallel DT method combined with MC simulations can greatly decrease FARs while keeping FRRs still acceptable. The results are promising because many applications can accept a classifier with an FRR of 9.16% and an FAR of 1.44%. For applications that require an FAR less than 1% while still keeping the FRR less than 10%, it can still be reached with a little more computation.

Table II shows the actual FRRs and FARs had more deviation from theoretical predictions for larger numbers of splitting. This trend results from fewer features in each subvector and from decreasing validity of the independence assumption. Both factors can be overcome by lengthening the common string, but an excessive length may be difficult for users to accept. The independence problem can also be overcome by representing the feature values using a different set of basis vectors. In other words, we can apply some transformation, such as wavelet transform.

VI. WAVELET ANALYSIS AND TRANSFORMS

Wavelet analysis [17] provides a countable infinite basis for $L^2(\mathbf{R})$ and in many wavelet systems the elements of this basis are orthogonal to each other and normalized. The discrete wavelet transform (DWT) uses a new finite basis to represent discrete-time signals of a given length. The number of samples limits the number of independent wavelet coefficients similar to the discrete Fourier transform (DFT). For 16 samples, there can be four levels $(d_0 - d_3)$ and a total of 16 coefficients in the wavelet decomposition. We applied the discrete-wavelet transform after splitting the original sequence into four subvectors of length 15, 14, 15, and 14, respectively, and padding with 0's to reach a length of 16 for each subvector. If $\{y[k]$ where $k = 0$ to $15\}$ are discrete-time samples, the Haar wavelet coefficients can be calculated as

$$
\begin{aligned}
c_0[0] &= 2^{-4}\sum_{k=0}^{15} y[k] \\
d_0[0] &= 2^{-4}\left[\sum_{k=0}^{7} y[k] - \sum_{k=8}^{15} y[k]\right] \\
d_1[0] &= 2^{-\frac{7}{2}}\left[\sum_{k=0}^{3} y[k] - \sum_{k=4}^{7} y[k]\right] \\
d_1[1] &= 2^{-\frac{7}{2}}\left[\sum_{k=8}^{11} y[k] - \sum_{k=12}^{15} y[k]\right] \\
d_2[0] &= 2^{-3}\left[\sum_{k=0}^{1} y[k] - \sum_{k=2}^{3} y[k]\right] \\
d_2[1] &= 2^{-3}\left[\sum_{k=4}^{5} y[k] - \sum_{k=6}^{7} y[k]\right] \\
d_2[2] &= 2^{-3}\left[\sum_{k=8}^{9} y[k] - \sum_{k=10}^{11} y[k]\right] \\
&\vdots \\
d_3[5] &= 2^{-\frac{5}{2}}\left[y[10] - y[11]\right] \\
d_3[6] &= 2^{-\frac{5}{2}}\left[y[12] - y[13]\right] \\
d_3[7] &= 2^{-\frac{5}{2}}\left[y[14] - y[15]\right].
\end{aligned} \tag{4}
$$

Each wavelet coefficient of a discrete signal can be calculated as a weighted sum of samples in other wavelet systems as well.

For each case, let $(c_0[0], d_0[0], d_1[0], d_1[1], d_2[0], \ldots, d_2[3], d_3[0], \ldots, d_3[7])$ represent the concatenation of all 16 coefficients from performing wavelet decomposition on a subvector. After being associated with its desired output, a case represented in a different basis system is obtained. The wavelet decomposition was performed for every case and eight DTs were trained for each user: four DTs trained to process transformed subvectors and four DTs trained to process untransformed subvectors. Since the wavelet transform is linear and invertible, the resulting vectors will be independent if the time-domain vectors are independent. Table III shows the performance of the parallel DT constructed with four original subvectors and four wavelet subvectors. For comparison, we also included a parallel DT with four original subvectors and four Fourier coefficient subvectors.

TABLE III
FRRs AND FARs OBTAINED FROM THE PARALLEL DT CONTAINING EIGHT SUBVECTORS

	j	FRR	FAR
Wavelet	3	9.62%	0.88%
Fourier	3	7.79%	1.44%

Receiver operating characteristic (ROC) curves are a useful technique for summarizing and visualizing the performance of classifiers [7], [18]. Our proposed methodology provides three different mechanisms for adjusting the tradeoff between

TABLE IV
FARS AND (1-FRR)'S OBTAINED FROM PARALLEL DTS TRAINED FROM EIGHT SUBVECTORS

Ratio (j/8)	o8*		o4w4**		o4f4***	
	FAR	1 - FRR	FAR	1 - FRR	FAR	1 - FRR
1/8	25.74%	99.43%	15.26%	98.85%	23.58%	99.43%
2/8	6.29%	97.48%	4.15%	96.56%	6.11%	97.37%
3/8	1.44%	90.84%	0.88%	90.38%	1.41%	92.21%
4/8	0.30%	78.58%	0.19%	78.35%	0.27%	83.51%
5/8	0.05%	58.30%	0.03%	60.71%	0.05%	64.03%
6/8	0.01%	35.97%	0.00%	40.55%	0.01%	41.70%
7/8	0.00%	17.87%	0.00%	23.60%	0.00%	22.68%
8/8	0.00%	3.67%	0.00%	8.82%	0.00%	6.64%

*: o8 indicates all 8 subvectors come from splitting the original data including raw data and simulated data.

**: o4w4 indicates 4 subvectors come from splitting the original data including raw data and simulated data; the other 4 contain their wavelet transforms.

***: o4f4 indicates 4 subvectors come from splitting the original data including raw data and simulated data; the other 4 are their Fourier transforms.

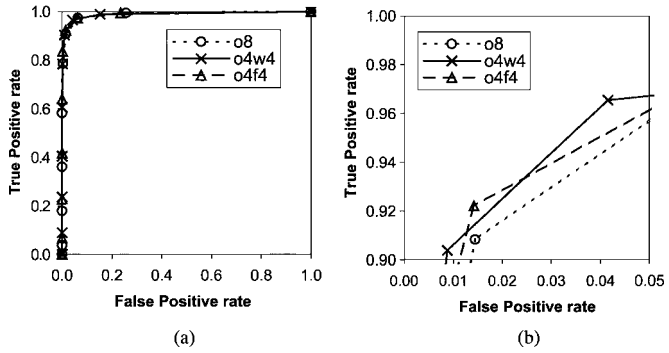

Fig. 3. ROC curves.

the FAR and FRR metrics. CART software allows the specification of a relative misclassification cost. The number of cases simulated for each class could also be used to adjust the different error rates. We decided to vary the number of parallel DTs required to agree on verifying a user in order to generate ROC curves. Table IV shows FAR and (1-FRR) depending on the number of concurring DTs required for verifying a user. Comparative results are provided for schemes using DWT and DFT transforms of subvectors.

Plotting 1-FRR versus FAR produces the ROC graphs shown in Fig. 3. Fig. 3(a) indicates that all three curves are very close to each other. For this application, we pay close attention to the region of $\mathrm{FAR} < 5\%$ and $\mathrm{FRR} < 10\%$, which is shown in Fig. 3(b). It shows that either wavelet transform or Fourier transform can improve the performance in this region because of their higher curve positions.

An additional experiment was conducted to explore the tradeoff between string size and accuracy. The size of the string was reduced from 32 characters to eight characters by selecting the substring "computer". A parallel DT was constructed with four original subvectors and four wavelet subvectors. An FRR of 13.97% and an FAR of 9.19% were obtained. The results show that a string size on the order of 30 characters is necessary to attain an acceptable accuracy.

VII. ANALYSIS

The complexity of the training algorithm depends on the number of characters in the common string, which is limited for a practical application. Our string contains 37 characters including spaces. Such a length is comparable to the length in [1] of the combination of username, password, first name, and last name. If an upper boundary is set for the length of the common string, we can consider its length as constant. Thus, the complexity of the wavelet decomposition is also constant. The complexity of the training algorithm also depends on the number of users, denoted n. The number of training cases determines the complexity of training DTs for a fixed-length input vector. The number of cases for each DT equals $180(\mathrm{n}-1)$. The overall complexity for training is therefore $\mathrm{O}(\mathrm{n}^2)$ for all users where n represents the number of users. The initial setup of the corresponding security systems is time consuming, as a lot of users' keystrokes need to be recorded and their classifiers need to be built. The complexity is polynomial indicating that the learning algorithm is efficient. The on-line calculation time of four wavelet transforms and eight DT classifications is fixed and negligible.

Once a sufficient number of typing samples have been collected, it may not be necessary to retrain the entire system to add new users. It could be sufficient to train only DTs specific to new users. The verification for existing users would then be as robust against false acceptance of new users as it would be against false acceptance of nonusers. Having separate DTs for individual users also makes it easier to update the system when individual typing patterns change over time.

Our proposed approach shows promise for a purely biometric authentication system based solely on keystroke patterns. The method has an FRR of 9.62%, which means a legal user may be denied once in ten logins, and an FAR of 0.88%, which means the opportunity of an imposter attaining authorized is less than 1%. The convenience of this method is outstanding because a user only needs to pick a username and type a given string of length 30–40 nine times to set up his account. The user does not even need to memorize any password or username for login. What he needs to do is select his username from a list, type the given string, and hit Enter.

VIII. CONCLUSION AND FUTURE WORK

A new technique for user authentication based on keystroke patterns has been proposed. To attain sufficient data to train more effective DTs, approximately 19 times as much simulated data was generated in addition to the data provided by users. By splitting the keystroke feature vectors, wavelet analysis was performed on four 16-element subvectors and eight DT classifiers were trained for every user. For each user, a parallel DT was constructed based on the eight DTs. The parallel DT accepts a user as legal, if at least three DTs output 1. The training data, except simulated data, were provided by 43 users, who each typed a given common string of length 37, including spaces, nine consecutive times. The test data were provided by users without any particular requirement. Since some users provided a small

number of test samples, meaningful results were obtained by averaging the performance over all users. Results of 9.62% FRR and 0.88% FAR were obtained by rigorous testing.

Future work could include exploring the possibility that new users could be added without retraining the entire system once sufficient data has been collected. Future work could also include checking the generic significance of this methodology by testing it on larger groups and by applying it to other biometric authentication systems.

ACKNOWLEDGMENT

The authors wish to thank S. Babu for his great efforts in coding the MFC program and collecting the data, which made their research possible and efficient. They also wish to thank all the users who participated in the experiments.

REFERENCES

[1] R. Joyce and G. Gupta, "Identity authentication based on keystroke latencies," *Commun. ACM*, vol. 33, no. 2, pp. 168–176, Feb. 1990.

[2] R. Gaines, W. Lisowski, S. Press, and N. Shapiro, "Authentication by Keystroke Timing: Some Preliminary Results," Rand Corporation, Rand Report R-256-NSF, 1980.

[3] D. Umphress and G. Williams, "Identity verification through keyboard characteristics," *Int. J. Man-Mach. Stud.*, vol. 23, no. 3, pp. 263–273, 1985.

[4] J. Leggett and G. Williams, "Verifying identity via keystroke characteristics," *Int. J. Man-Mach. Stud.*, vol. 28, no. 1, pp. 67–76, 1988.

[5] G. Leggett, J. Williams, and M. Usnick, "Dynamic identity verification via keystroke characteristics," *Int. J. Man-Mach. Stud.*, vol. 35, no. 6, pp. 859–870, 1991.

[6] S. Bleha, C. Slivinsky, and B. Hussein, "Computer-access security systems using keystroke dynamics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 12, pp. 1217–1222, Dec. 1990.

[7] A. J. Mansfield and J. L. Wayman, "Best Practices in Testing and Reporting Performance of Biometric Devices,", NPL Rep. CMSC 14/02, 2002.

[8] J. Garcia, "Personal Identification Apparatus," U.S. Patent 4 621 334, 1986.

[9] J. R. Young and R. W. Hammon, "Method and Apparatus for Verifying an Individual's Identity," U.S. Patent 4 805 222, 1989.

[10] F. Monrose and A. Rubin, "Authentication via keystroke dynamics," in *Proc. ACM Workshop*, 1997, pp. 48–56.

[11] B. Hussien, R. Mclaren, and S. Bleha, "An application of fuzzy algorithms in a computer access security system," *Pattern Recognit. Lett.*, vol. 9, pp. 39–43, 1989.

[12] M. Brown and S. J. Rogers, "User identification via keystroke characteristics of typed names using neural networks," *Int. J. Man-Mach. Stud.*, vol. 39, no. 6, pp. 999–1014, 1993.

[13] M. S. Obaidat and B. Sadoun, "Verification of computer users using keystroke dynamics," *IEEE Trans. Syst., Man, Cybern,*, pt. B, vol. 27, no. 2, pp. 261–269, Apr. 1997.

[14] S. M. Rovnyak, C. W. Taylor, and Y. Sheng, "Decision trees using apparent resistance to detect impending loss of synchronism," *IEEE Trans. Power Del.*, vol. 15, no. 4, pp. 1157–1162, Oct. 2000.

[15] Salford Systems website. (2004, Apr.). [Online] Available: http://www.salford-systems.com

[16] S. Cho, C. Han, D. H. Han, and H. I. Kim, "Web based keystroke dynamics identity verification using neural network: over 200 sets of training data," *J. Organiz. Comput. Elect. Commerce*, vol. 10, no. 4, pp. 295–307, 2000.

[17] C. S. Burrus, R. A. Gopinath, and H. Guo, *Introductions to Wavelets and Wavelet Transforms: A Primer*. Upper Saddle River, NJ: Prentice-Hall, 1998.

[18] T. Fawcett, *ROC Graphs: Notes and Practical Considerations for Researchers*. Dordrecht, The Netherlands: Kluwer, 2004.

Yong Sheng (S'00–M'03–SM'05) received the B.S. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, the M.S. degree in electrical engineering from China Electric Power Research Institute, Beijing, China, and the Ph.D. degree in computational analysis and modeling from Louisiana Tech University, Ruston, in 1989, 1992, and 2002, respectively. He also received the M.S. degree in computer science and the M.S. degree in mathematics from Louisiana Tech University in 2004.

From 1992 to 1997, he was with China Electric Power Research Institute as an Electrical Engineer, mainly conducting research. From 2002 to 2003, he was a Visiting Assistant Professor of Computer Information Systems at Grambling State University, Grambling, LA. He has been a Postdoctoral Fellow at Indiana University-Purdue University Indianapolis since 2004. His research interests include pattern recognition, computer and network security, mathematical models and computational techniques, biometrics, and power system protections.

Vir V. Phoha (M'96–SM'03) received the M.S. and Ph.D. degrees in computer science from Texas Tech University, Lubbock.

He is an Associate Professor of Computer Science at Louisiana Tech University, Ruston. His research interests include anomaly detection, network and Internet security, Web mining, control of software systems, intelligent networks and nonlinear systems.

Dr. Phoha is a member of the ACM.

Steven M. Rovnyak (S'89–M'95) was born in Lafayette, IN, in 1966. He received the B.A. degree in mathematics in 1988 and the B.S., M.S., and Ph.D. degrees in electrical engineering in 1988, 1990, and 1994, respectively, all from Cornell University, Ithaca, NY.

Currently, he is Assistant Professor of Electrical and Computer Engineering at Indiana University-Purdue University Indianapolis (IUPUI). Previously, he was an Assistant Professor of Electrical Engineering at Louisiana Tech University, Ruston, and spent two years as Postdoctoral Associate at Cornell University. He worked on Electric Power Research Institute projects while a graduate student and Postdoctoral Associate at Cornell. He has developed algorithms to process wide-area monitoring system data with grant support from Entergy Transmission. His research interests include the development of pattern recognition methodologies for protective relaying and for one-shot stability controls.

Dr. Rovnyak is a member of the Phi Beta Kappa and Phi Kappa Phi honor societies.