Biometric Authentication Using Mouse Gesture Dynamics

Bassam Sayed, Issa Traoré, Isaac Woungang, and Mohammad S. Obaidat, *Fellow, IEEE*

***Abstract*—The mouse dynamics biometric is a behavioral biometric technology that extracts and analyzes the movement characteristics of the mouse input device when a computer user interacts with a graphical user interface for identification purposes. Most of the existing studies on mouse dynamics analysis have targeted primarily continuous authentication or user reauthentication for which promising results have been achieved. Static authentication (at login time) using mouse dynamics, however, appears to face some challenges due to the limited amount of data that can reasonably be captured during such a process. In this paper, we present a new mouse dynamics analysis framework that uses mouse gesture dynamics for static authentication. The captured gestures are analyzed using a learning vector quantization neural network classifier. We conduct an experimental evaluation of our framework with 39 users, in which we achieve a false acceptance ratio of 5.26% and a false rejection ratio of 4.59% when four gestures were combined, with a test session length of 26.9 s. This is an improvement both in the accuracy and validation sample, compared to the existing mouse dynamics approaches that could be considered adequate for static authentication. Furthermore, to our knowledge, our work is the first to present a relatively accurate static authentication scheme based on mouse gesture dynamics.**

***Index Terms*—Behavioral biometrics, biometric authentication, computer security, identity verification, mouse dynamics.**

I. Introduction

WHILE THE primary focus in the design of biometric systems has been on improving their accuracy, the integration of such systems in existing information systems requires addressing additional considerations. Existing information systems rely on globally distributed networked infrastructures, involving a diversity of systems that collaborate in delivering a specific set of services. The inherent heterogeneity of these systems creates a layer of complexity that makes their design and operation a difficult undertaking. The integration of a biometric subsystem in such a complex system environment requires taking into account concerns such as social impact, usability, interoperability, resilience, and scalability. For instance, biometric systems carry private user information that could be misused if accessed by intruders, although such a threat could be mitigated using privacy-aware biometric cryptographic techniques [1]. Biometric systems could be the target of attacks, for instance, through forgeries [2]. Many biometric systems rely on complex data processing algorithms that may not scale when deployed in globally distributed network environments.

As a result, despite the wide usage of biometric technology for physical security, the adoption of biometrics in day-to-day use of computer systems has been slow [3]. Another reason for this limited usage of biometrics is the reliance on special-purpose hardware devices for biometric data collection. Although some computer vendors have started integrating the needed hardware into their products, the vast majority of machines currently available lack such special hardware devices. This limits the scope of where biometric technology can be used, as it will only be available for organizations that can buy the required additional hardware.

A new category of biometrics that is gaining in popularity is referred to in the literature as behaviometrics (for behavioral biometrics), where analysis focuses on studying the user's behavior while he interacts with a computing system for the purpose of identification [4]–[6]. One interesting example of behaviometrics is mouse dynamics biometrics [5]–[8]. Mouse dynamics biometric recognition involves extracting the behavioral features related to the mouse movements and analyzing them to extract a signature, which is unique for every individual, and as such can be used to discriminate different individuals. The main strength of mouse dynamics biometric technology is in its ability to continuously monitor the legitimate and illegitimate users based on their sessional usage of a computer system. This is referred to as *continuous authentication*. Continuous authentication, or identity confirmation based on mouse dynamics, is very useful for continuous monitoring applications such as intrusion detection.

However, unlike traditional biometric systems, mouse dynamics biometric technology may face some challenges when applied to static authentication, which consists of checking the user's identity at login time. The key challenge is the data capture process, which requires more time to collect sufficient amount of mouse movements for accurate user identity verification [9] than can reasonably be tolerated, or afforded, in a realistic login process.

We tackle this challenge by proposing, in this paper, a new mouse dynamics recognition framework that allows

Manuscript received October 4, 2011; revised March 14, 2012; accepted July, 26, 2012. Date of publication January 22, 2013; date of current version April 17, 2013.

B. Sayed and I. Traoré are with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8W 3P6, Canada (e-mail: bassam@ece.uvic.ca; itraore@ece.uvic.ca).

I. Woungang is with the Department of Computer Science, Ryerson University, Toronto, ON M5B 2K3, Canada (e-mail: iwoungan@scs.ryerson.ca).

M. S. Obaidat is with the Department of Computer Science, Monmouth University, West Long Branch, NJ 07764 USA (e-mail: obaidat@monmouth.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSYST.2012.2221932

performing the authentication in a short time and as such may be used for static authentication (at login time). We use mouse gestures to achieve our goal. A mouse gesture results from the combination of computer mouse movements and clicks in a way that the software recognizes as a specific command. In our work, during the enrollment phase the user draws a set of gestures several times on a computer monitor using a mouse. We extract features from the captured data, analyze them, and then train a neural network that is later used for identification. In the verification phase, the user will be asked to replicate a subset of the gestures drawn by her in the enrollment phase to test against her stored profile. Experimental evaluation of the proposed approach with 39 users yields false acceptance rate (FAR) of 5.26% and false rejection rate (FRR) of 4.59% with a test session length of 26.9 s.

The remainder of this paper is organized as follows. In Section II, we summarize and discuss related work. In Section III, we give an overview of mouse gesture dynamics and present the design of our detection system. In Section IV, we present the classification technique used to analyze the mouse gestures. In Section V, we present our experimental evaluation process and results. In Section VI, we give conclusions and summarize our future work.

II. Related Work

There is extensive research literature on the use of the computer mouse as an input device in the human computer interaction field for the purpose of user interface design improvement [10]. It was not until recently, however, that mouse dynamics emerged as a behavioral biometric technology.

In [11], it was established that the actions recorded for a specific user while interacting with a graphical user interface are intrinsic to that user. These actions are recorded passively and validated throughout the session. The authors initially evaluated their model by collecting data from 22 participants. Then, using a one-hold-out cross validation test to compute the performance of the proposed system, an FAR of 2.4649% and an FRR of 2.4614% were obtained [6]. These results were later confirmed by increasing the overall number of participants to 48 users [12]. Although the work accomplished in this research may potentially be used both for static and dynamic authentication systems, the primary focus of the study was initially on continuous authentication that requires the user to be logged into the system to start the monitoring. Static authentication will require the design of a special-purpose GUI and asking the user to perform predefined actions to login. The new interface, and set of actions, could present some challenges related to the length of time required to capture enough data for user recognition.

Gamboa *et al.* [5] performed similar research by conducting an experiment to capture user interaction based on the mouse while playing a memory game. Fifty volunteers participated in the experiment. A sequential forward selection technique based on the greedy algorithm was used to select the best single feature and then add one feature at a time to the feature vector.

Gamboa *et al.* [5] showed that the equal error rate (EER) progressively tends to zero as more strokes are recorded. This means that the more interaction data the system records, the more accurate the system should be. But, as we commented earlier, it might be difficult to use such a method for static authentication at login time since the authors reported that the memory game took from 10–15 min to complete on average.

In [7], the authors proposed an approach for user reauthentication based on the data captured from the mouse device. A system was implemented that continuously monitored the user-invoked mouse events and movements, and raised an alarm when the user behavior deviated from the learned normal behavior. A decision tree classifier was used in the decision process. An experiment was conducted with 11 users, in which a false positive rate of 0.43% and a false negative rate of 1.75% were obtained. The authors, however, mentioned that their method would fail if the user did not generate enough mouse movements and events. In a static authentication setting where a limited amount of data is available, their approach may also be ineffective.

The main issues with the above works on mouse dynamic are that the minimum amount of data required to achieve meaningful user identification is impossible to obtain within the time constraint of a typical login process. As such, the proposed approaches may be used for user reauthentication (after login) or for continuous authentication, but they may not be suitable for static authentication (at login time). To our knowledge, so far only two papers in the literature, published by Syukri *et al.* [13] and Revett *et al.* [9] have targeted the use of mouse dynamics for static authentication.

A system that may potentially be used for static authentication, proposed by Syukri *et al.* [13], utilizes signatures drawn using a mouse for user identification. The extracted features were analyzed using geometric average means. The authors conducted two experiments involving 21 users, in the first of which a static database was used, and in the second a dynamically updated database was used. The performance achieved consisted of FRR = 9% and FAR = 8% for the static database, and FRR = 7% and FAR = 4% for the dynamic database.

More recently, Revett *et al.* [9] proposed a new mouse dynamic analysis approach for static authentication, named *mouse lock*, which exploits the analogy of a safe, in which the numbers are replaced with graphic thumbnail images. To login, using a mouse, the user is required to click in a password that consists of five images. Experimental evaluation conducted with six participants yielded, according to the type of images, FAR and FRR values of approximately 2%–5%. Although these results represent significant improvement in terms of performance, they are still preliminary, as indicated by the authors. It should also be noted that only six users were involved in the experimental evaluation, which is a very low number.

In this context, we propose a new approach in which the user draws, at login time, a few gestures (in a relatively short time) from which the dynamics are collected and analyzed for authentication purposes. To our knowledge, our work is the first to present a relatively accurate static authentication scheme based on mouse gesture dynamics. Existing gesture-based authentication schemes use other input devices (typically a stylus) [14].

Our work relates to the general field of identifying users based on their handwriting. A rich body of research has been produced on discriminating users based on their natural handwriting [15], [16]. This research is articulated in two main directions: signature verification and writer identification. While most of the prior work has focused on natural handwriting, there has recently been some interest in discriminating users based on their handwriting in a structured input language (such as Graffiti). Under this perspective, our work is closely related to the study conducted by Roberts *et al.* [17] on discriminating users based on their handwriting characteristics in Graffiti while using PDAs. The goal of the authors was to confirm or deny a claimed identity based on how the user reproduces a nonsecret challenge string. Evaluation of the proposed approach with 52 participants yielded FAR = 0.04% and FRR = 0.64% for strings of four characters, and EER=0% for 16-character strings. These results are extremely encouraging, although it must be noted that they were obtained in closed settings, under the assumption that no outsider can access the biometric system. This is an important difference with our work, in which no restriction is made on the types of users accessing the system. An obvious difference is the input device used for data collection; while, in the above work, Graffiti characters are drawn using a stylus, our focus is on gestures produced using the computer mouse.

Our work also relates to the general field of graphical passwords that have gained interest in the last few years [18], [19]. Similar to traditional passwords, in the existing graphical password schemes, the user is not only expected to memorize and remember the graphical passwords, he is supposed to hide it during the login process to mitigate possible shoulder surfing attacks. In contrast, our proposed scheme relies solely on the user biometric information; there is no need for the user to memorize or hide the gesture.

III. Mouse Gesture Detection

In this section, we give an overview of mouse gestures and present the design of our gesture detection system.

A. System Design and Pilot Experiment

Our approach to user authentication based on mouse gestures consists of presenting to the user one or several gestures drawn from an existing gesture set and asking him to replicate the gestures a certain number of times. The produced replications are then compared against templates produced by the user during the enrollment phase.

The raw data collected from the drawing area consist of the horizontal coordinate (x-axis), vertical coordinate (y-axis), and the elapsed time in milliseconds at each pixel. Each gesture replication for a given gesture can be defined as a sequence of data points and each data point can be represented by a triple $< x, y, t >$ consisting of the X-coordinate, Y-coordinate, and elapsed time, respectively. The jth replication of a gesture G can be represented as a sequence $G_j = \{< x_{1j}, y_{1j}, t_{1j} >, < x_{2j}, y_{2j}, t_{2j} >, \ldots < x_{nj}, y_{nj}, t_{nj} >\}$, where n is referred to as the gesture size (GS) and each $< x_{ij}, y_{ij}, t_{ij} >$ where $(1 \leq i \leq n)$ is a data point. The size of a gesture is the number of data points recorded for it. Fig. 1 illustrates an example of a drawn gesture.

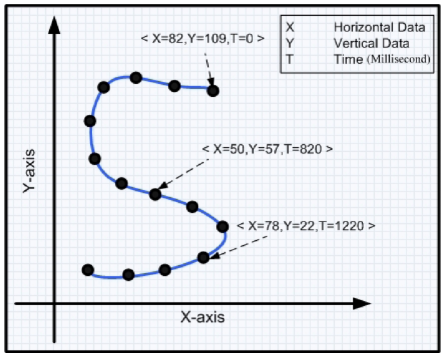

Fig. 1. Example of a drawn gesture involving $n = 14$ data points.

In order to design our system we conducted, in the early stages of our research, a pilot experiment that involved six users. The main purpose of the experiment was to explore whether it is possible to differentiate between individuals based on their behavioral biometrics while drawing mouse gestures. The participants in the pilot experiment were asked to replicate eight different types of gestures by drawing each gesture 20 times. The same eight gestures were used consistently for all the participants and the only requirement was to draw them in one stroke.

Based on the pilot experiment, we observed the following.

1) The average gesture size drawn in one stroke was 64 data points.
2) Some participants started to get used to the experiment and started to draw the gesture faster, which is a departure from their normal behavior.
3) The raw data contained some noise, such as data points with the same time stamp that must be filtered.
4) Although the users were told to be as consistent as possible while drawing the gestures, as expected, the variability in shape and size were clearly a major source of inconsistency.

Based on the data collected in the pilot study, we were able to design our gesture data acquisition and analysis framework. Our framework consists of four modules:

1) gesture creation module;
2) data acquisition and preparation module;
3) feature extraction module;
4) classification module.

We describe modules 1–3 in the subsequent subsections, and module 4 in Section IV.

B. Gesture Creation

The gesture creation module is a simple drawing application used to ask the participant to freely draw a predefined set of gestures. The main purpose of this module is to make the participant draw the gestures in his own way to replicate them later on. It is important to note here that the gestures are not tied to any language and they do not necessarily have a meaning. They can be any drawing that can be produced in a uni-stroke. Also, the gesture creation module serves as a

practice step for the participants to get familiar with the idea of drawing mouse gestures.

The gesture creation module assists the user in two different ways. First, it normalizes the input to the center of the drawing area. Even though the shifting of the drawn gesture is done, the data is stored without saving these changes.

Second, the module normalizes the gesture spacing to achieve a size of 64 data points. The 64 data points were based on the pilot experiment that we did in the early stages of our research. As mentioned earlier, we were able to determine the average size of drawing the predefined set of gestures in one stroke.

C. Data Acquisition and Preparation

The data acquisition and preparation module involves three main components, namely, data acquisition, data preparation, and data smoothing.

1) *Data Acquisition:* The data acquisition component loads the gestures, created initially by the user using the gesture creation module, and presents them to the user to replicate. The data acquisition module records the user interaction while drawing the gesture. The module essentially records the horizontal coordinates denoted by x_{ij}, vertical coordinates denoted by y_{ij}, and the elapsed time in milliseconds starting from the origin of the gesture t_{ij}, where n is the gesture size, $1 \leq i \leq n$, and j is the gesture replication number. For each user, the program creates a directory that will contain the user replications for the different gestures. Each gesture must be replicated a specific number of times (e.g., 20 times). The user has to wait 3 s between consecutive replications. The idea behind this waiting time is to prevent the user from drawing the gesture too fast. The module asks the user to release the mouse between each successful replication during the wait time. The main reason why we implemented the wait time and mouse release is based on one of the observations made in the pilot experiment. We assumed that the wait time and mouse release will force the users to maintain their normal behavior each time they replicate the gesture.

2) *Data Preprocessing:* The data acquisition module preprocesses the collected raw data from the computer mouse in such a way that some noise patterns are ignored or dropped. This is required since the data produced by the pointing devices is usually jagged and irregular. This kind of preprocessing will filter data resulting from a common problem of the pointing devices, specifically data points generated with the same timestamp where $t_i = t_{i+1}$.

After preprocessing the raw data, the data acquisition module applies two types of normalizations to the input data. The first is center normalization and the second is size normalization. The center normalization shifts the gesture to the center of the drawing area as implemented in the gesture creation module. We have also to normalize the size so that the final size of the gesture is equal to the size of the template gesture in order to compare the two gestures as explained later. We accomplish the normalization by only accepting gestures drawn by the participant whose size is greater than or equal to the size of the template gesture. If the gesture size is bigger than the template size, we apply the k-means algorithm [20] to cluster the data points into 64 clusters. We use the standard Euclidean distance as the distance measure between the data points in three dimensions $< x, y, t >$. Afterward, the centroids of the 64 clusters are used to form the new gesture.

3) *Outlier Removal and Data Smoothing:* To build the user profile, we remove outliers from the data and then smooth it. Data smoothing is generally used to eliminate noise and extract real patterns from data. In our framework, we use smoothing to smooth the data among the different replications obtained for each user. Generally, humans cannot draw the same gesture with the same exact detail twice under the same conditions. This will result in some variability in the replicas produced by the same individual for the same gesture. Data smoothing allows us to smooth such variability and minimize its effect on the learning process. We use the robust version of the standard weighted least-squares regression (WLSR) method to smooth the data.

We use Peirce's criterion [21] to eliminate the outliers. Peirce's criterion method is a robust statistical based method that does not make any assumptions about the data. Peirce's criterion can also deal with the case where the data have several suspicious values.

Peirce's criterion determines outliers by computing empirically the maximum allowable deviation from the sample mean. Given m the sample size and n the number of outliers, the ratio R between the maximum allowable deviation to the standard deviation of the sample is obtained from Peirce's criterion table. The maximum allowable deviation is computed as $d_{max} = R \times \sigma$, where σ is the standard deviation of the sample. A data item x_i is considered an outlier if $\mid x_i - x_m \mid > d_{max}$, where x_m is the sample mean.

Using the above method and starting from $n = 1$, outliers are removed sequentially by incrementing the number of possible outliers, while keeping the original values of the mean, standard deviation, and sample size. The process is repeated until no other data item needs to be eliminated. We refer interested readers to [21] for more details about the Peirce's criterion.

We apply the outlier removal and data smoothing steps only to the horizontal (x-axis) and vertical (y-axis) coordinates data (excluding time). We construct a vector that aggregates the same occurrence of the first data point from each of the different replications. Then, we apply the Peirce's criterion and WLSR method to the data in the vector to produce clean and smoothed data. We repeat the process for each of the remaining data points of the gesture. Note that the smoothing occurs only on the training samples (while building the user profile) and not on the test data.

Algorithm 1 is implemented by the outlier removal and data smoothing module and assumes the following.

1) Let m be the number of replications.
2) Let n be the size of the gesture.
3) Let $p_{ij} = (x_{ij}, y_{ij})$ be a data point, where $1 \leq j \leq m$, $1 \leq i \leq n$.
4) Given a gesture G, we denote by G_j the jth replica $G_j = (p_{1j}, p_{2j}, \ldots, p_{nj})$.
5) Let P_i denote a vector containing the ith data point from each of the different replications, where $i = 1, 2, \ldots, n$: $P_i = (p_{i1}, p_{i2}, \ldots, p_{im})$.

Algorithm 1 Smooth($VG \leftarrow \{G_1, G_2 \ldots G_m\}, n, m$)

Require: Integers ($n > 1$) and ($m \geqslant 1$)

Ensure: The value of $V'G \leftarrow \{G'_1, G'_2 \ldots G'_m\}$ smoothed data.

1: $TV \leftarrow \varnothing$ {Temporary vector}
2: **for** $i \leftarrow 1$ to n **do**
3: $P'_i \leftarrow WLSR(P_i)$
4: $TV \leftarrow TV \bigcup \{P'_i\}$
5: **end for**
6: $V'G \leftarrow \{TV\}^T$ {Transpose TV}
7: **return** $V'G$

TABLE I
FEATURE EXTRACTED FROM RAW DATA

Feature Description	Notation	Definition
Horizontal coordinate	x	x-axis data
Vertical coordinate	y	y-axis data
Absolute time	t	–
Horizontal velocity	hv	$v_{hor} = \frac{\Delta x}{\Delta t}$
Vertical velocity	vv	$v_{ver} = \frac{\Delta y}{\Delta t}$
Tangential velocity	tv	$v = \sqrt{v_{hor}^2 + v_{ver}^2}$
Tangential acceleration	ta	$v' = \frac{\Delta v}{\Delta t}$
Tangential jerk	tj	$v'' = \frac{\Delta v'}{\Delta t}$
Path from the origin in pixels	l	–
Slope angle of the tangent	θ_i	$\theta_i = \arctan(\frac{y_i}{x_i})$
Curvature	c	$c = \frac{\Delta \theta}{\Delta l}$
Curvature rate of change	δc	$\delta c = \frac{\Delta c}{\Delta l}$

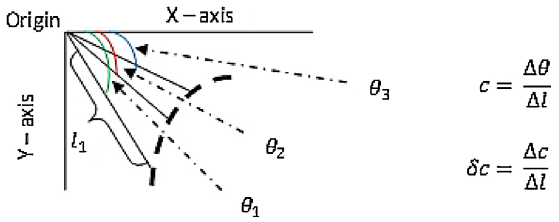

Fig. 2. Angle of curvature and its rate of change for a portion of a drawn gesture.

D. Feature Extraction

Based on the pilot study, the output of the data acquisition module by itself is not enough to form a unique signature for each user. The feature extraction module extracts 12 features from the raw data. The complete list of the extracted features is provided in Table I. Features selection was conducted by analyzing sample data from our exploratory study, and identifying the features that exhibit strong reproducibility and discriminative capabilities (i.e., large interclass variability and small intraclass variability).

Fig. 2 illustrates the angle of the tangent with the x-axis and the length of the path from the origin to the data point. Both measures are used to compute the curvature and its rate of change with respect to the length from the origin. Fig. 3 compares, for two arbitrary users from the pilot study, two features extracted from two replications from User 1 against one replication from User 2 for the same gesture (M gesture). Note how the extracted features from the two replications of User 1 are close to each other compared to the features extracted from User 2.

IV. MOUSE GESTURE ANALYSIS

In this section, we present our gesture classification approach and summarize the testing parameters involved.

A. Classification Technique

In order to classify the gestures, we first tried the principal component analysis technique on sample data, yielding poor performance. We also explored the usage of the feed-forward back propagation multilayer perceptrons network; however, the training step using this type of neural network was exhaustive and time consuming. The justification behind this outcome is due to the back propagation nature of the training process. We actually stopped the training process when it exceeded five hours (on a computer system with a 2 GHz Core 2 Duo CPU and 2 GB RAM) for only a population of two users. We tried the learning vector quantization (LVQ) neural network using a monolithic design, which also turned out to be ineffective. However, LVQ using a modular design gave satisfactory results.

1) *Modular Neural Network Design:* A monolithic neural network is considered as an unstructured black box that does not have the flexibility and modularity to solve a subset of certain problems. In [22], the monolithic nature of a neural network is proven to be one of its major performance limiting characteristics. The modular neural network design addresses this limitation by introducing more flexibility and structure in the neural network architecture.

In our design, we grouped the input features into four feature sets FS_i, where $i = 1, 2, \ldots, 4$ and we designed the neural network so that it has four modules NNM_i, *where* $i = 1, 2, \ldots, 4$. Each of the four feature groups will be an input for a module of the neural network $FS_i \rightarrow NNM_i$. The grouping of the features into the sets was based on the logical relations between the features. We grouped the features into two spatial sets, one spatiotemporal set, and one temporal set as follows:

1) spatiotemporal Set (FS_1) : x, y, and t;
2) spatial Set (FS_2) : l, c, and δc;
3) temporal Set (FS_3) : hv, vv, tv, ta, and tj;
4) spatial Set (FS_4) : l and θ.

Note that feature l is used in two different modules.

Typically, the LVQ neural network paradigm consists of three layers: an input layer, a hidden layer that is sometimes referred to as the Kohonen layer, and an output layer. The number of neurons in the output layer of the standard LVQ architecture is always equal to the number of different classes characterizing the input data. This means that the output of the neural network is an index for one of the classes in the training population.

In the modular design, the architecture of the neural network modules is the same for all the users in the training population. Each module of the neural network is an independent system by itself and follows the typical LVQ design. In our modular

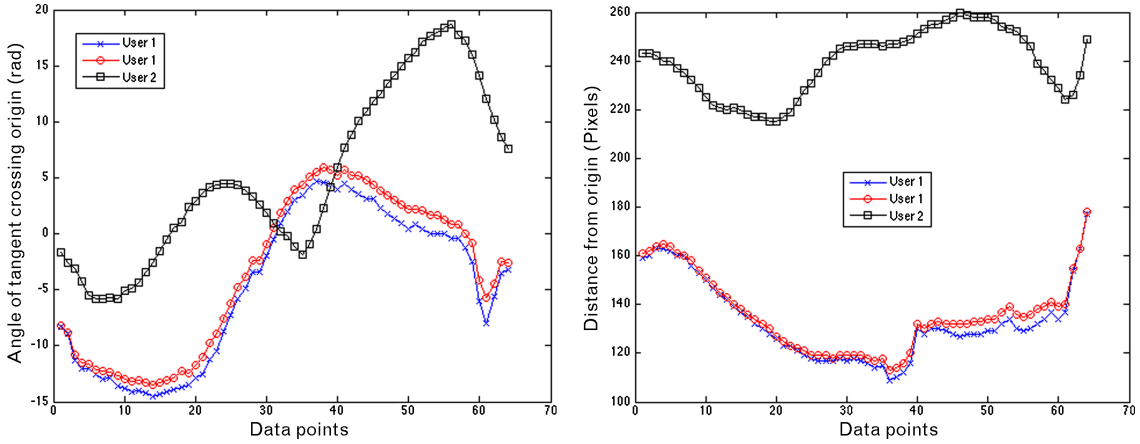

Fig. 3. Comparing angle of curvature and distance from original features of two replicas belonging to User 1 and one replica belonging to User 2 for the same gesture.

neural network design the output layer has two neurons, as the network is classifying two classes. The first class is the current user class index (i.e., self), and the second class represents the remaining users in the training population (i.e., nonself). We will explain in a later section how the training of the neural network works.

The number of neurons in the hidden layer was determined through trial and error, starting with a small number of neurons and increasing the number of neurons until no improvement in performance was observed. Based on the data collected in the pilot study and different setups of the neural network architecture, we came to the conclusion that 72 neurons in the hidden layer were acceptable for each of the modules. Hence, the number of neurons in the hidden layer is exactly the same for all the modules. While the hidden layer and the output layer designs are the same for all the modules in the network, the input layer is an exception. In the standard LVQ input layer design, each sample must be paired with its label (i.e., self or nonself) and then presented to the LVQ neural network. In our case, the sample is a particular gesture replica represented by a set of features. Since the feature values are computed for each data point in the drawn gesture, the total number of input values for a given neural network module is equal to the number of features in the corresponding feature set multiplied by the gesture size. Hence, the number of neurons for the input layer is expressed mathematically as follows: $N_{input}(NNM_i) = F_i \times GS$, where F_i is the number of features in feature set $FS_i, (1 \leq i \leq 4)$ and GS is the gesture size. Every module in the neural network is considered a separate classifier. The decision of each module does not affect the other modules. That being said, the final decision is based on the aggregated output of the four modules. The output of each module is aggregated through a voting scheme yielding the final decision.

Any of the four modules has only one of two possible output values: self (i.e., user recognized) or nonself (user not recognized). Let λ denote a decision threshold corresponding to the minimum number of modules that must classify a given sample as self for the user to be recognized as genuine. The selection of a specific value of λ is a design decision. For instance, a value of $\lambda = 3$ means that at least three modules must classify the sample as self for the user to be recognized as genuine; a value of $\lambda = 4$ means that all four modules must classify the sample as self for the user to be recognized as genuine. Therefore, even if three modules classify the sample as self, the user will still be rejected. Cases where $\lambda \leq 1$ are excluded for obvious reasons, because they could lead to scenarios where a sample would be classified as genuine while none or only one of the modules has classified it as such. When $\lambda = 2$, at least two modules must classify the sample as self for the sample to be accepted, which means that in the case where exactly two modules classify the sample as self the overall decision will be to accept the user as genuine. A more restrictive design could be adopted by rejecting cases where exactly two modules would classify the sample as self (while the other two classify it as nonself); such cases are already covered by having $\lambda = 3$. Therefore, the value of λ must be selected in the range $[2, 4]$.

2) *Neural Network Training:* Using the LVQ2 algorithm developed by Kohonen for training [23], we follow the regular procedure of training the LVQ neural network by presenting the sample data along with its labels. In our framework, there is a separate profile for each gesture for each user. It is important to note that while all the gestures share the same modular neural network architecture, explained in the previous section, each gesture for each user is associated with a separate set of weights derived through training. For each gesture for each user, the associated weights will represent the corresponding profile.

In our pilot study, all the gesture replications drawn by each user were divided into two separate sets, a training set and a validation set. We apply a hierarchical training procedure to train the neural network. The main objective of the hierarchical training is to improve the performance of the network. We form a three-level tree-like hierarchy from the training set by averaging the different replications drawn by the user. The averaging of the replications follows a specific procedure. The procedure starts by dividing the replications in the training set into groups of equal size that form the bottom most level. The size of each group must be less than the size of the

training set. The average of each group is calculated to form the second level of the hierarchy. We call the second level, the level of the sub-means. The root of the hierarchy, which is the overall mean of the replications, is formed by calculating the average of the sub-means in the second level that forms the top most level. After constructing such a hierarchy, the training of the modular neural network proceeds from the root to the bottom level of the hierarchy. In other words, we construct the hierarchy in a bottom-up approach and then train the network in a top-down manner.

The rationale behind such a technique is that starting the training of the neural network with the overall mean of the replication allows the weights of the neural network to get quickly adjusted to the center of the current user cluster of replications. Then, the weights get adjusted to the sub-means in the second level until reaching the actual replications of the user in the bottom level. In the pilot experiment, we noticed that by applying the hierarchical training technique, the training error in some cases converged to zero, which is optimal.

We train the neural network through both positive and negative training. Positive training is based on the set of replications drawn by user X. However, the selection of the negative training set is not straightforward. One way of conducting the negative training is by considering all, or a subset of the replications of the remaining users in the training population as the negative training set. For instance, if we have n users in the training population the negative training set for each user may be the replications of the remaining $n-1$ users. Clearly, this is not the optimal solution from a scalability perspective. Ideally, the negative training set should contain the replications of the nearest subject to the current user. This should be good enough for the neural network to draw the decision boundary between the current user and the closest subject, which, in turn, allows the network to discriminate between the current user and all the subjects in the training population. In our implementation, we used (1) (see below) to pick the nearest and the second nearest users to conduct the negative training. The same formula was used by Lee *et al.* [15] to measure the distance between the feature vectors in their proposed system for online human signature verification. Let $m(a, i)$ and $\sigma^2(a, i)$ be the sample mean and the sample variance of feature i computed from a specific set of gesture replications for user a, respectively. Then, the distance measure for feature i between users a and b for a specific gesture is defined as follows:

$$d_i(a, b) = \frac{|m(a, i) - m(b, i)|}{\sqrt{\sigma^2(a, i) + \sigma^2(b, i)}}. \quad (1)$$

Based on the above formula, the distance between user a and user b is defined as follows:

$$d(a, b) = \sum_{i=1}^{n} d_i(a, b). \quad (2)$$

We say that user b is the nearest subject to user a in population P if

$$d(a, b) = min_{x \in P \text{ and } x \neq a} d(a, x).$$

We take the gesture replications of the nearest and second nearest users in population P as the negative training samples.

TABLE II
SYSTEM VARIABLES FOR THE DATA ACQUISITION MODULE USED IN THE TEST PHASE

Variable	Description
α	Number of gesture replications that need to be performed by the user for a specific gesture.
β	Number of α replications that need to be accepted in order for that given gesture to be considered successful.
ζ	Number of different types of gestures the user must draw in the test phase.

Then, we apply the hierarchical training technique discussed previously. We repeat the same procedure for each user in the training population. Finally, all the derived profiles are stored to be used later in the test phase. Note that (on a computer system with a 2 GHz Core 2 Duo CPU and 2 GB RAM) it takes 4 min to build a profile for a single-gesture template.

B. Test Sessions and Parameters

At the beginning of the testing session, the user will be asked to draw a number of different types of gestures selected from the set of gestures provided during the enrollment phase; let ζ denote such a number. The user will be asked to provide a number of replications for each of the selected gestures; let α denote such a number. For a reproduction session of a gesture to be considered valid, a minimum number of the provided replications should match the profile successfully; let β denote such a number. Table II summarizes the above-mentioned variables.

V. EXPERIMENTAL EVALUATION

We present, in this section, the experimental evaluation of the proposed framework. We start by describing the experimental conditions and procedures, and then present, analyze, and discuss the obtained results.

A. Apparatus

All the participants used the same Toshiba Satellite X205-SLi1 laptop to enroll in our experiments. The hardware configuration of the laptop was an Intel Core 2 Duo processor clocked at 2 GHz, with 2 GB of physical memory, running Microsoft Windows 7 configured with a resolution 1440×900 native screen resolution. All the participants used a Microsoft Explorer optical mouse to replicate the different gestures; even the same mouse pad was used during the experiment. The sampling rate was 125 Hz, which is the Microsoft Windows XP/Vista/7 OS default sampling rate for USB based mouse devices.

The software involved in our experiments was already deployed on the laptop. The software, written in C++, consisted of a gesture creation tool and an enrollment tool. The gesture creation tool is used to create the gesture templates and store them with the user credentials in a database. The enrollment tool loads the templates from the database and allows the participant to enroll against them. The replications resulting from the enrollment are stored in the replications database.

B. Participants

The main objective of our experiment was to be able to recognize individuals based on their mouse gestures. Ideally, the system should be able to recognize, with a high degree of accuracy, the behavior of each user while replicating a specific gesture. To achieve such a goal, 39 volunteers were involved in our experiment. The participants were from various backgrounds, with ages ranging from 19 to 54 years old. Participants' skill levels ranged from novice users using computers only occasionally to individuals using computers on a regular basis as part of their professions (e.g., university faculty members, students, engineers).

Participants in the experiment were divided into two groups: a group of 29 individuals representing legal users and a group of ten individuals representing impostors.

C. Method and Data

Before starting the experiment, the participants were shown a sample gesture set as an example of uni-stroke gestures. The idea was to get the participants familiar with drawing gestures. All the legal participants used the same laptop to draw the same set of prechosen gestures. The gestures replications along with the participants' credentials were stored in a user database. There was only one requirement that was to draw such gestures in one stroke, as in practice, programs that make use of mouse gestures typically implement them in a uni-stroke form.

The legal participants in the experiments were asked to draw the following five gesture templates: G, Y, M, Z, and 5. The gestures were chosen to include combinations of lines, angles and curves. The main reasoning for this choice was, the more angles and curves the gesture has, the more it will require muscle tension and concentration from the users. This, in turn, imposes the intrinsic behavior of the human motor control while drawing such gestures.

Each of the 29 legal users was asked to draw the gesture templates first and then replicate each gesture template 30 times. The 30 replications were provided over six sessions, each involving exactly five replicas, with an average of one day interval between consecutive sessions. In a session, each gesture was replicated five times in sequence. For each (of the five) gesture templates, 150 replications per participant were collected, giving in total 4350 genuine replications spread over 174 legal sessions.

Each of the ten impostors was asked to forge the (five) gesture templates of at least five legitimate users selected randomly from the above 29 legal users. Care was taken to ensure that each legal user would be targeted by exactly five different impostors. An impostor was shown a sample gesture for a given legitimate user and asked to forge it by providing five replicas. This resulted in 3625 forged replications spread over 725 impostor sessions.

D. Evaluation Process

During a test session, the impostor is asked to replicate each gesture a certain number of times; for this evaluation, we used $\alpha = 5$. For each of the legal users, we selected five sessions out of the six sessions for training purposes and the remaining one was used for testing. This means that $\alpha = 5$ and β can range between ($1 \leq \beta \leq 5$). Both α and β were explained previously in Table II. It is important to note that while both the test and training data undergo the same preprocessing step by the data acquisition module (see Section III-C2), only the training data undergo the smoothing step (see Section III-C3).

We start by building a reference profile (per gesture template) for each of the n legitimate users, using 25 replicas from each user as the enrollment samples ($n = 29$). Note that none of the replicas from the impostors will be used in this process.

To calculate the FRR, the remaining five replicas (not involved in the enrollment) of each of the legal users U_j, $1 \leq j \leq n$, is tested against their own reference profile. This corresponds to a total number of n trials. A false rejection FR will occur if the number of matching replicas is below the threshold β

$$FR = \begin{cases} 1, & \text{if number of matching replicas} < \beta \\ 0, & \text{otherwise.} \end{cases}$$

The global FRR is computed as the ratio between the total number of false rejections over all the test trials and the total number of test trials n

$$FRR = \frac{\sum_{j=1}^{n} count(\{FR\}_j)}{n} \times 100$$

where j ($1 \leq j \leq n$) is the test trial index.

During the experiments, each of the legal users was attacked by five different impostors; the impostors attempted to forge each legal user's (five) gestures five times; this corresponds to five attack sets per legal user $a = 5$.

To calculate the FAR, each of the impostor samples was compared against the profile of the corresponding legal user being targeted in the forgery. The total number of test trials in this case is $N = 5 \times a \times n$. A false acceptance FA will occur if the number of matching replicas is above the threshold β

$$FA = \begin{cases} 1, & \text{if number of matching replicas} \geq \beta \\ 0, & \text{otherwise.} \end{cases}$$

The global FAR is computed as the ratio between the total number of false acceptances over all the test trials and the total number of test trials N

$$FAR = \frac{\sum_{j=1}^{N} count(\{FA\}_j)}{N} \times 100$$

where j ($1 \leq j \leq N$) is the test trial index.

As mentioned before, the system variable α was selected and fixed to equal five replicas; however, we varied the values of the other two system variables to measure their impact on the framework performance. Namely, the voting system variable λ that ranged from ($2 \leq \lambda \leq 4$) and β that ranged from ($1 \leq \beta \leq \alpha$). The global $FAR_{\beta,\lambda}$ and $FRR_{\beta,\lambda}$ were calculated accordingly for each system variable value.

TABLE III
RECOGNITION PERFORMANCE FOR G GESTURE

λ		β = 1	β = 2	β = 3	β = 4	β = 5
λ = 2	FAR	16.83	10.98	6.84	4.11	1.40
	FRR	0	3.44	17.24	39.65	72.98
λ = 3	FAR	8.44	3.54	1.64	0.66	0.07
	FRR	50	65.51	74.13	83.33	93.67
λ = 4	FAR	2.18	0.54	0.16	0	0
	FRR	89.08	93.67	97.12	99.42	100

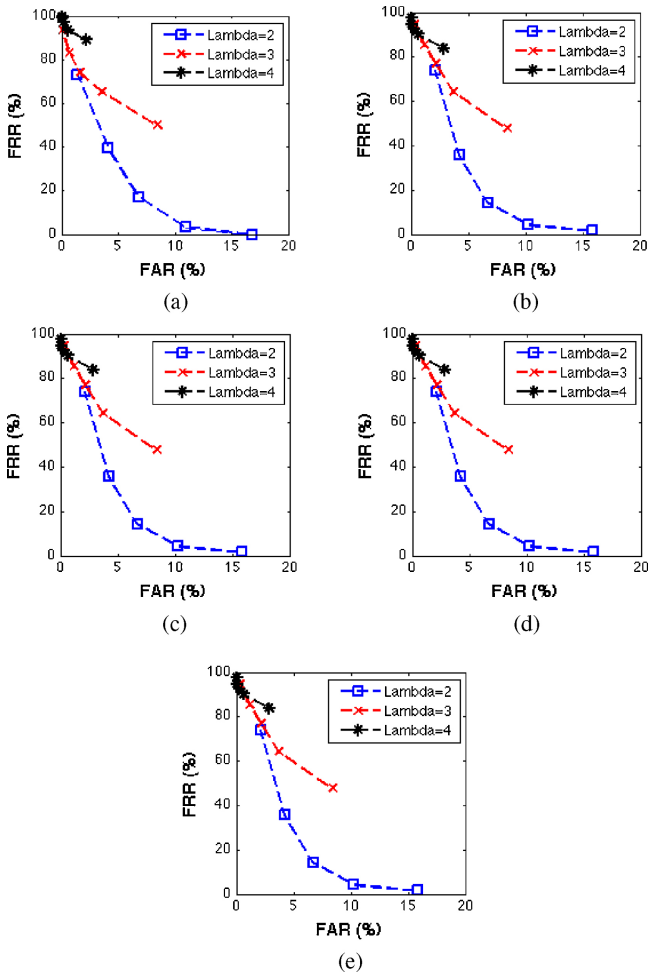

Fig. 4. ROC curves of G, Y, Five, M, and Z gestures. λ is the minimum number of modules that need to agree for voting in the neural network classifier. (a) ROC curve for the G gesture. (b) ROC curve for the Y gesture. (c) ROC curve for the Five gesture. (d) ROC curve for the M gesture. (e) ROC curve for the Z gesture.

E. Evaluation Results

We applied the above-mentioned evaluation method six times separately for each of the five gestures ($\zeta = 1$) involved in our experiment, and computed global FRR and FAR while varying α and β. Each time one session was used as the test session exactly once and the remaining five sessions were used to build the user profile. The obtained results were averaged and are shown in Tables III–VII and depicted using receiver operating characteristic (ROC) curves in Fig. 4.

As we can see from the results, all the gestures are close to each other in performance. Some of the best operating points obtained in those cases vary for the FAR between 8.65% and 11.17%, while the FRR stand between 3.44% and 11.49%, when $\beta = 2$ and $\lambda = 2$.

In any case, it appears from the above results that using a single gesture may not be very effective on its own. In this regard, we modified our evaluation method so that it includes two gestures in the test session ($\zeta = 2$) as opposed to only one gesture ($\zeta = 1$) in the previous evaluation. The number of replicas drawn by the individuals remained fixed at $\alpha = 5$ for each gesture; however, we varied the β system variable. This means that the user will be required to draw in total ten replicas in the test session. Let β_1 be the number of accepted replicas from the first gesture and β_2 be the number of accepted replicas from the second gesture. A false acceptance will occur if any combination of β_1 and β_2 satisfies $(\beta_1 + \beta_2 \geq \beta,$ where $\beta_1, \beta_2 \leq \alpha)$, and false rejection will occur if any combination of β_1 and β_2 satisfies $(\beta_1 + \beta_2 < \beta,$ where $\beta_1, \beta_2 \leq \alpha)$. Tables VIII, IX, and Fig. 5(a), (b) illustrate the performance results and ROC curves obtained for the combination of the Z gesture with G gesture at one time and Five gesture with M gesture at a second time. We observed an improvement in performance with (FAR = 5.93%, FRR = 8.62%) for (Z, G) combination, and (FAR = 6.04%, FRR = 10.91%) for (Five, M) combination, when $\beta = 5$ and $\lambda = 2$.

Furthermore, we studied the combination of three and four gestures ($\zeta = 3, 4$). The previously mentioned method for combining the results of two gestures is applied with the

TABLE IV
RECOGNITION PERFORMANCE FOR Y GESTURE

λ		β = 1	β = 2	β = 3	β = 4	β = 5
λ = 2	FAR	15.83	10.20	6.73	4.25	2.11
	FRR	1.72	4.59	14.36	35.63	74.13
λ = 3	FAR	8.44	3.73	2.21	1.18	0.33
	FRR	47.70	64.36	77.01	85.63	94.82
λ = 4	FAR	2.85	0.61	0.26	0	0
	FRR	83.90	90.22	92.52	94.82	97.70

TABLE V
RECOGNITION PERFORMANCE FOR NUMBER FIVE GESTURE

λ		β = 1	β = 2	β = 3	β = 4	β = 5
λ = 2	FAR	14.74	9.01	5.63	3.11	1.18
	FRR	1.72	11.49	22.41	45.40	77.01
λ = 3	FAR	6.80	2.16	0.61	0.21	0
	FRR	39.08	61.49	72.98	84.48	94.82
λ = 4	FAR	1.18	0.11	0	0	0
	FRR	93.67	98.27	98.85	99.42	100

TABLE VI
RECOGNITION PERFORMANCE FOR THE M GESTURE

λ		β = 1	β = 2	β = 3	β = 4	β = 5
λ = 2	FAR	16.81	11.17	6.65	3.44	0.61
	FRR	1.14	5.17	17.81	37.93	75.86
λ = 3	FAR	8.89	2.99	1.16	0.28	0
	FRR	41.37	52.87	63.79	79.31	96.55
λ = 4	FAR	2.23	0.23	0.09	0	0
	FRR	86.20	92.52	96.55	99.42	99.42

TABLE VII
RECOGNITION PERFORMANCE FOR Z GESTURE

λ		β = 1	β = 2	β = 3	β = 4	β = 5
λ = 2	FAR	14.91	8.65	5.42	3.21	0.87
	FRR	1.14	5.17	21.26	50.57	84.48
λ = 3	FAR	6.06	2.02	1.04	0.42	0
	FRR	49.42	67.24	78.73	88.50	97.70
λ = 4	FAR	1.18	0.09	0	0	0
	FRR	87.93	94.25	98.27	99.42	100

TABLE VIII
RECOGNITION PERFORMANCE FOR Z GESTURE AND G GESTURE COMBINED

λ		β = 3	β = 4	β = 5	β = 6	β = 7
λ = 2	FAR	12.22	8.57	5.93	3.95	2.57
	FRR	1.14	2.29	8.62	18.96	35.05
λ = 3	FAR	3.17	1.93	1.08	0.41	0.20
	FRR	54.59	66.09	76.43	83.90	90.22
λ = 4	FAR	0.25	0	0	0	0
	FRR	95.40	98.85	100	100	100

TABLE IX
RECOGNITION PERFORMANCE FOR FIVE GESTURE AND M GESTURE COMBINED

λ		β = 3	β = 4	β = 5	β = 6	β = 7
λ = 2	FAR	12.52	9.44	6.04	3.94	1.97
	FRR	1.14	3.44	10.91	21.83	35.63
λ = 3	FAR	2.64	1.21	0.41	0.16	0.11
	FRR	41.95	50.57	69.54	83.33	91.37
λ = 4	FAR	0.13	0.09	0.02	0	0
	FRR	95.40	98.85	78.71	100	100

addition of a third and a fourth gesture. Let β_3 correspond to the number of accepted replicas of the third gesture, and β_4 correspond to the number of accepted replicas of the fourth gesture. Tables X and XI and Fig. 5(c) and (d) illustrate the performance results and ROC curves obtained for combining the Z, G, and M gestures for the first test, and Z, G, M, and Y gestures for the second test. When $\beta = 7$ and $\lambda = 2$, we obtain for the (Z, G, M) combination FAR = 6.39% and FRR = 4.59%. We obtain for the (Z, G, M, Y) combination FAR = 5.26% and FRR = 4.59% with ERR=5.11% when $\beta = 10$ and $\lambda = 2$, which is a net improvement in the system performance as the number of gestures increase. We excluded gesture Five from the above combination because it achieves the worst performance compared to the other gestures. For instance, by computing the half total error rate (HTER) (i.e., the sum of the FAR and FRR divided by two), we can notice that the HTER for the gesture Five is 10.25%, while for the other gestures it ranges between 6.91% and 8.17%.

By analyzing the curves provided in Figs. 4 and 5 and the performance results provided in Tables III–XI, we can notice that as λ increases the FRR increases while the FAR decreases, regardless of the number of gestures combined. Similarly, as β increases the FRR increases while the FAR decreases. Intuitively, this can be explained by the fact that high values of λ or β lead to more restriction, which means greater possibility for false rejections. In contrast, lower values

TABLE X
RECOGNITION PERFORMANCE FOR Z, G, AND M GESTURES COMBINED

λ		β = 6	β = 7	β = 8	β = 9	β = 10
λ = 2	FAR	8.55	6.39	4.68	3.47	2.43
	FRR	1.72	4.59	8.62	16.66	31.03
λ = 3	FAR	1.28	0.68	0.36	0.18	0.09
	FRR	62.64	67.24	75.28	83.33	89.65
λ = 4	FAR	0	0	0	0	0
	FRR	99.42	100	100	100	100

TABLE XI
RECOGNITION PERFORMANCE FOR Z, G, M, AND Y GESTURES COMBINED

λ		β = 8	β = 9	β = 10	β = 11	β = 12
λ = 2	FAR	8.45	6.71	5.26	4.32	3.40
	FRR	1.14	1.72	4.59	9.77	14.36
λ = 3	FAR	1.40	1	0.64	0.32	0.20
	FRR	60.91	66.66	72.98	79.31	87.35
λ = 4	FAR	0.02	0	0	0	0
	FRR	99.42	100	100	100	100

TABLE XII
MARGIN OF ERROR FOR THE CI AROUND THE HTER RESULT OF Z, G, M, AND Y GESTURES COMBINED

δ	Z
90%	1.44%
95%	1.72%
99%	2.26%

of β or λ will lead to a more permissive design, which means greater possibility for false acceptances. We can also notice in some cases where $\lambda = 4$ (and β is greater than some value), the system is so restrictive that all samples are rejected regardless of whether these samples are from genuine users or impostors. Such exterme cases yield $FRR = 100\%$ and $FAR = 0\%$ as performances.

The assessment of confidence of the accuracy of biometric systems is not straightforward. Likewise, Gamassi *et al.* [24] presented a thorough discussion of the challenges surrounding the estimation of confidence of the accuracy of biometric systems. According to Bengio and Mariethoz [25], standard statistical tests cannot be used "as is" to measure the statistical significance of person authentication models, because several of the performance metrics used to assess those models, such as EER or HTER, are aggregates of two measures (e.g., FAR and FRR). We calculated the confidence interval (CI) for the combination of four gestures using the method proposed in [25]. In the proposed method, the distribution of the number of errors is approximated to a normal distribution with standard deviation σ. We used the HTER formula to calculate the CI around the result of the combined four gestures. Based on our experiments, we have HTER=4.93% and $\sigma = 0.0088$. The CI around an HTER is $HTER \pm Z$, where Z is the margin of error. Table XII depicts the margin of error at a different confidence level δ.

It must be noted that increasing the number of gestures comes at the expense of usability. As is often expected, a tradeoff must be made between security and usability.

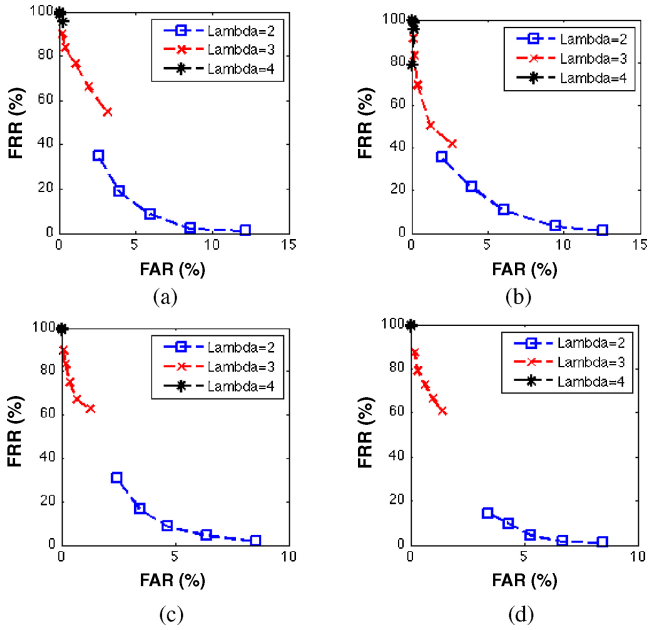

Fig. 5. ROC curves for combined gestures. (a) ROC curve for the Z and G gestures combined. (b) ROC curve for the Five and M gestures combined. (c) ROC curve for the Z, G, and M gestures combined. (d) ROC curve for the Z, G, M, and Y gesture combined with ERR = 5.11%.

TABLE XIII
LENGTH OF TEST SESSION FOR DIFFERENT GESTURE COMBINATIONS

Number of Gestures Combined	Gesture Production Time (s)	Time between Gestures (s)[a]	Gesture Verification Time (s)[b]	Total Time (s)
1	5	0.4	1.25	6.65
2	10	0.9	2.5	13.4
3	15	1.4	3.75	20.15
4	20	1.9	5	26.9

[a] Pause time performed by the data capture module between each gesture replication drawn by the user.
[b] Time taken by the modular neural network to recognize the gesture and output decision.

From the above results, it can be observed that although combining several gestures increases the challenge of the test session on both the legitimate and nonlegitimate users, both the FAR and FRR tend to decrease. On the other hand, combining gestures increases the length of the test session, as the average production time of a gesture is 1 s. Table XIII provides a breakdown of the duration of the different components of a test session according to the number of gestures combined. The time between gestures during verification that is set to 100 ms can be lowered if necessary; note that the time between each gesture replication during enrollment was set to 3 s for the reasons explained earlier. The session length for four gestures is 26.9 s, which, in relative terms, is not too long a time when compared to performance results obtained for some existing graphical password schemes proposed in the literature. Likewise, a recent survey by Briddle *et al.* covering several existing graphical password schemes shows that their login times vary from 6 to 180 s [26].

VI. CONCLUSION

In this paper, we highlighted the challenges faced by mouse dynamics biometric technology when it is applied to static authentication and proposed a new framework based on mouse gesture dynamics that achieves encouraging results toward addressing those challenges. The proposed framework uses a modular design of the LVQ neural network for classification. We conducted an experimental evaluation of the framework involving 39 users, yielding FAR = 8.65%, FRR = 5.17% when only one gesture was used, FAR = 8.57%, FRR = 2.29% when two gestures were combined, FAR = 6.39%, FRR = 4.59% when three gestures were combined, and FAR = 5.26%, FRR = 4.59% when four gestures were combined. Our study clearly indicated that the combination of several gestures be used to achieve acceptable performance in static user authentication. A tradeoff, however, must be made between usability and security, which necessarily indicates some limitations in the maximum number of gestures that could be used. The results obtained in our study are encouraging and highlight the promise of mouse dynamics for static authentication. However, the results obtained with four gestures show that more work must be done to increase the accuracy of our proposed scheme before it can be used in practice for static authentication.

As mentioned earlier in the related work section, while there are a few promising works on mouse dynamics biometric analysis, most of these proposals require a significant amount of data for authentication, and as such may be used for continuous authentication or user reauthentication, but are not adequate for static authentication at login time. Among the few proposals that can be used for static authentication, either relatively low accuracy was achieved, as in the work of Syukri *et al.* [13], or a relatively small sample size was used, as in the work of Revett *et al.* [9]. In this paper, we addressed the above issues by improving accuracy and using a greater sample size. On one hand, while Syukri *et al.* obtained (FAR = 8%, FRR = 9%) with a static database, we achieved about half of these error rates using the same type of database (i.e., withour updating). On the other hand, while our error rates are comparable to those obtained by Revett *et al.*, our user population is much larger.

In our future work, we intend to enhance the accuracy of our proposed scheme by revisiting various aspects. For instance, we plan to investigate a different decision scheme other than the simple voting used. For example, if we know that one of the neural network modules is more accurate than the others, we might investigate a weighted voting scheme. The interesting part would be how to scale the weights per user as opposed to a system wide weight. We also plan to extend and strengthen our current feature set by exploring some new features with the purpose of improving ultimately the accuracy of our proposed authentication scheme. In particular, we plan to identify relevant features from existing handwriting schemes and investigate how these features can help us achieve better accuracy. Furthermore, we plan to investigate new analysis techniques that would allow us reducing the number of gesture replicas used in our enrollment process, which is currently very high.

Since our proposed system is entirely software based, integration in a complex system environment (e.g., e-commerce or e-learning portals) should be straightforward from an implementation perspective. This simply requires exposing, through an application programming interface, essential functions for enrollment and verification. While the verification is relatively fast, it takes 20 min, on average, to train the user profile (during enrollment) for the five gesture templates based on our experimental platform (i.e., a computer system with a 2 GHz Core 2 Duo CPU and 2 GB RAM). Such a training time can be reduced drastically using a more powerful server and parallel computing. Using advanced software middleware technologies (e.g., web services) would allow addressing the interoperability challenges inherent in complex system enviroments.

However, critical challenges would still remain that should be addressed for our system to be fully operational in a complex system environment. One of these challenges is the protection of the proposed system against security attacks. Like many other biometric technologies, mouse dynamics can be the target of replay attacks. Such threats could be mitigated by strengthening the protection of the biometric templates using cryptographic and data shuffling techniques [1]. Mouse dynamics can also be the target of automated attacks, also referred to as generative attacks, where high quality forgeries can be generated automatically using a small set of genuine samples [2]. We plan, in our future work, to strengthen our system by investigating the impact of generative attacks against it.

REFERENCES

[1] S. Cimato, M. Gamassi, V. Piuri, R. Sassi, and F. Scotti, "Privacy-aware biometrics: design and implementation of a multimodal verification system," in *Proc. Annu. Comp. Sec. Apps. Conf.*, 2008, pp. 130–138.

[2] D. Lopresti, F. Monrose., and L. Ballard, "Biometric authentication revisited: Understanding the impact of wolves in sheep's clothing," in *Proc. 15th USENIX Sec. Symp.*, 2006.

[3] M. S. Obaidat and N. Boudriga, *Sec of e-Sys and Comp Networks*. Cambridge, MA: Cambridge Univ. Press, 2007.

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

[5] H. Gamboa and A. Fred, "A behavioral biometric system based on human-comp. inter," in *Proc. Conf. Biometric Tech. Human Identification*, vol. 5404. 2004, pp. 381–392.

[6] A. A. E. Ahmed and I. Traoré, "A new biometric tech. based on mouse dynamics," *IEEE Trans. Dependable Secure Comput.*, vol. 4, no. 3, pp. 165–179, Jul.–Sep. 2007.

[7] M. Pusara and C. E. Brodley, "User reauthentication via mouse movements," in *Proc. ACM Workshop Visualization Data Mining Comp. Sec. (VizSEC/DMSEC)*, 2004, pp. 1–8.

[8] N. Zheng, A. Paloski, and H. Wang, "An efficient user verification system via mouse movements," in *Proc. 18th ACM Conf. Comp. Commun. Sec.*, 2011, pp. 139–150.

[9] K. Revett, H. Jahankhani, S. de Magalhaes, and H. M. D. Santos, "A survey of user authentication based on mouse dynamics," in *Proc. ICGeS, CCIS'12*, 2008, pp. 210–219.

[10] P. Oel, P. Schmidt, and A. Shmitt, "Time prediction of mouse-based cursor movements," in *Proc. Joint AFIHM-BCS Conf. Human Comp. Inter.*, vol. 2. Sep. 2001, pp. 37–40.

[11] A. A. E. Ahmed and I. Traoré, "System and method for determining a comp. user profile from a motion-based input device," U.S. patent 10/555408, PCT/CA2004/000669, 2003.

[12] A. Nazar, I. Traoré, and A. Ahmed, "Inverse biometrics for mouse dynamics," *Int. J. Artif. Intell. Pattern Recognit.*, vol. 22, no. 3, pp. 461–495, May 2008.

[13] A. F. Syukri, E. Okamoto, and M. Mambo, "A user identification system using signature written with mouse," in *Proc. 3rd Australasian Conf. Inform. Sec. Privacy*, 1998, pp. 403–414.

[14] S. Patel, J. Pierce, and G. Abowd, "A gesture-based authentication scheme for untrusted public terminals," in *Proc. UIST*, Oct. 2004.

[15] L. L. Lee, T. Berger, and E. Aviczer, "Reliable online human signature verification systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 6, pp. 643–647, Jul. 1996.

[16] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 63–84, Jan. 2000.

[17] R. R. M. Roberts, R. A. Maxion, K. S. Killourhy, and F. Arshad, "User discrimination through structured writing on PDAs," in *Proc. 37th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2007, pp. 378–388.

[18] I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter, and A. D. Rubin, "The design and analysis of graphical passwords," in *Proc. 8th Conf. USENIX Sec. Symp. (SSYM)*, 1999, p. 1.

[19] C. Varenhorst. (2004). *Passdoodles: A Lightweight Authentication Method* [Online]. Available: http://people.csail.mit.edu/emax/papers/varenhorst.pdf

[20] S. Lloyd, "Least squares quantization in pcm," *IEEE Trans. Inform. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[21] S. Ross, "Peirce's criterion for the elimination of suspect experimental data," *J. Eng. Tech.*, vol. 20, no. 2, 2003.

[22] F. Azam, "Biologically inspired modular neural networks," Ph.D. dissertation, Virginia Polytechnic Instit. State Univ., Blacksburg, 2000.

[23] T. Kohonen, *Self-Organizing Maps* (Springer Series in Information Sciences, vol. 30), 3rd ed. Berlin, Germany: Springer, 2001.

[24] M. Gamassi, M. Lazzaroni, M. Misino, V. Piuri, D. Sana, and F. Scotti, "Accuracy and performance of biometric systems," in *Proc. Instrum. Meas. Tech. Conf.*, 2004, pp. 510–515.

[25] S. Bengio and J. Mariethoz, "A statistical significance test for person authentication," in *Proc. Odyssey: Speaker Language Recognition Workshop*, 2004.

[26] R. Biddle, S. Chiasson, and P. C. V. Oorschot, "Graphical passwords: Learning from the first twelve years," School Comp. Sci., Carleton Univ., Ottawa, ON, Canada, Tech. Rep. TR-11-01, Jan. 2011.

Bassam Sayed received the B.Sc. degree in computer science from Helwan University, Cairo, Egypt, and the M.A.Sc. degree from the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada, where he is currently pursuing the Ph.D. degree.

His current research interests include behavioral biometrics, Botnet detection, and web-based malware detection.

Issa Traoré received the Ph.D. degree in software engineering from the Institute Nationale Polytechnique-LAAS/CNRS, Toulouse, France, in 1998.

He has been with the Faculty of the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada, since 1999, where he is currently an Associate Professor and a Coordinator with the Information Security and Object Technology Laboratory.

Isaac Woungang received the M.S. and Ph.D. degrees in mathematics from the Université de la Méditerranée-Aix Marseille II, France, and the Université du Sud, Toulon & Var, France, in 1990 and 1994, respectively.

Since 2002, he has been with Ryerson University, Toronto, ON, Canada, where he is currently an Associate Professor of computer science and a Coordinator with the Distributed Applications and Broadband Laboratory, Department of Computer Science.

Mohammad S. Obaidat (F'05) received the M.S. and Ph.D. degrees in computer engineering with a minor in computer science from Ohio State University, Columbus.

He is an internationally renowned Academic/ Researcher/Scientist. He is currently a Full Professor of computer science with Monmouth University, West Long Branch, NJ. He has previously held the positions of Chair of the Department of Computer Science and Director of the Graduate Program, Monmouth University. He has published over ten books and 500 refereed technical articles.

Dr. Obaidat has received extensive research funding. He is the Editor-in-Chief of three scholarly journals and is also an editor and Advisory Editor of numerous international journals and transactions, including IEEE journals and transactions. He is a fellow of SCS. He has chaired numerous international conferences and given numerous keynote speeches all over the world.