Biometric Recognition Based on Free-Text Keystroke Dynamics

Ahmed A. Ahmed and Issa Traore

***Abstract*—Accurate recognition of free text keystroke dynamics is challenging due to the unstructured and sparse nature of the data and its underlying variability. As a result, most of the approaches published in the literature on free text recognition, except for one recent one, have reported extremely high error rates. In this paper, we present a new approach for the free text analysis of keystrokes that combines monograph and digraph analysis, and uses a neural network to predict missing digraphs based on the relation between the monitored keystrokes. Our proposed approach achieves an accuracy level comparable to the best results obtained through related techniques in the literature, while achieving a far lower processing time. Experimental evaluation involving 53 users in a heterogeneous environment yields a false acceptance ratio (FAR) of 0.0152% and a false rejection ratio (FRR) of 4.82%, at an equal error rate (EER) of 2.46%. Our follow-up experiment, in a homogeneous environment with 17 users, yields FAR = 0% and FRR = 5.01%, at EER = 2.13%.**

***Index Terms*—Biometrics, continuous authentication, free text recognition, keystroke analysis, neural networks.**

I. Introduction

A SUCCESSFUL authentication at the beginning of a computing session, also referred to as static authentication, does not provide any remedy against the session being hijacked later by some malicious user. Session hijacking involves an intruder seizing control of a legitimate user session after successfully getting a valid authentication session identifier while that session is still in progress. A possible remedy against such a scenario, referred to as continuous authentication (CA), consists of re-authenticating the user repeatedly and unobtrusively throughout the lifetime of the session.

Keystroke dynamics biometrics could be appropriate for CA because keystrokes can be collected unobtrusively using a standard keyboard throughout a session without any knowledge of the user. Keystroke dynamics is a behavioral biometric that aims to identify humans based on the analysis of their typing rhythms on a keyboard [1]–[5], [8] and [10]. The dynamics are extracted by measuring the dwell time (the length of time a key is held down) and fly time (the time duration from a key released to the next key pressed) for each keyboard action. Fig. 1 illustrates the dwell and fly times. The data is then translated to a set of constructs (monographs, digraphs, tri-graphs or n-graphs), each describing the human behavior in performing the described action. A monograph represents the action of pressing on a key on the keyboard. The action can be described by the key code as well as the dwell time. A digraph represents a typing action performed by the user from a specific key to another key on the keyboard. The digraph action is described by the two key codes of the from/to keys and the fly time between these keys.

One of the challenges of keystroke analysis is the unstructured and sparse nature of the information conveyed by keystrokes, which consists essentially of timing data. Furthermore, such information is characterized by strong variability due to the inherent instability of human behavioral characteristics as well as changes in environmental conditions (e.g. hardware) when collecting such data. To cope with such instability, most research on keystroke analysis has targeted samples based on structured, predefined text, also referred to as fixed text [1], [2] and [10]. Fixed text analysis could be used for static authentication at login time; however, it is not suitable for CA because the user will be required to write predefined phrases many times during the session. In contrast, recognition of text freely typed by users during their normal interaction with a computer, referred to as free text analysis [4], could be used for CA since data collection and analysis can be performed transparently and continuously throughout the login session without the knowledge of the user. Free text analysis faces a key challenge due to the fact that several monographs or digraphs could be missing from the enrolment samples making detection even harder. Furthermore, the appropriate application of keystroke analysis for continuous authentication faces two additional challenges, namely efficiency and effectiveness. Efficiency refers to the run-time efficiency of the detector, specifically its scalability and the extent to which it achieves timely detection. Effectiveness refers to the degree to which the system accurately detects identity impersonation or misappropriation of authentication credentials in the system, while maintaining an acceptable level of false alarms. Free text analysis of keystrokes is a relatively new area of research in which an extremely limited number of papers has been published to date [3]–[5] and [8]. Despite the encouraging results achieved so far, addressing at the same time both effectiveness and efficiency in free text analysis is still an open research challenge.

In this paper, we introduce a new approach for keystroke analysis that is suitable for free text detection. As mentioned

Manuscript received February 10, 2011; revised May 4, 2012 and March 1, 2013; accepted April 5, 2013. This paper was recommended by Associate Editor H. Ishibuchi.

The authors are with the Electrical and Computer Engineering Department, University of Victoria, Victoria, BC, V8W 3P6, Canada (e-mail: itraore@ece.uvic.ca; aahmed@ece.uvic.ca).

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/TCYB.2013.2257745

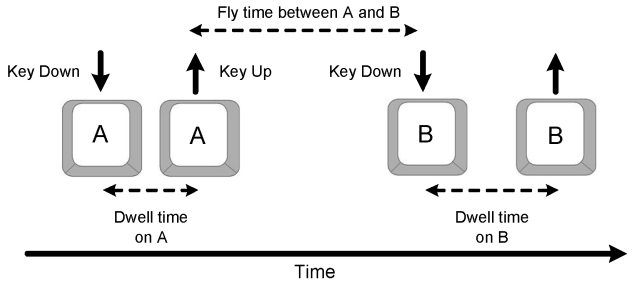


Fig. 1. Dynamics collected through typing activity.

earlier, an important limitation of keystroke analysis is the small amount of information conveyed by the keystroke data. This is even more of a concern in the free text analysis of keystrokes, where quite often unseen or unexpected behavioral patterns must be established or predicted based on available information. To deal with this situation, we introduce a new free text analysis technique, which can predict missing values, allowing closing the created information gap. The proposed approach combines monograph/digraph approximation with neural network analysis for free text recognition. Unlike the existing implementations of free text analysis systems, which are based on limited or fixed-text enrollment methods, the enrollment process of the proposed detection system is performed with few restrictions and a completely free text sample. Experimental evaluation, involving 53 users, of the proposed system yielded very promising performance figures with a FAR = 0.0152% and FRR = 4.82%. In order to study the impact of confounding factors, a follow-up experiment has been conducted in homogeneous environment yielding FAR = 0% and FRR = 5.01%.

The rest of the paper is structured as follows. In Section II, we summarize and discuss related work on free text analysis. In Section III, we describe user identity modeling based on free text and how corresponding signatures are used for user enrolment. We also illustrate the key components of the proposed keystroke dynamics signature using sample users' sessions. In Section IV, we describe our approach to check user identity based on our proposed free text model. In Section V, we describe and discuss the results obtained in a free experiment organized to evaluate the proposed approach. In Section VI, we present a controlled experiment organized as a follow up on the previous (free) experiment to study the impact of confounding factors. Finally, in Section VII, we make concluding remarks and outline future work.

II. Related Work

A. Summary

Since the early 1980s, a rich body of research has been produced in the area of keystroke dynamics recognition [1]–[5]. Most of these works target fixed text detection and focus on using this technology for static (user) authentication or access control [1], [2] and [10].

For dynamic and passive monitoring, as required for CA, we need to be able to detect the user without requiring him to enter a predefined message or text; therefore, free text detection is essential for such a purpose. Free text detection presents huge challenges, which partially explains the limited amount of related work available in the literature.

Monrose and Rubin authored one of the earliest works on the free text detection of keystrokes [5]. The authors collected, over a period of seven weeks, typing samples from 42 users performing structured and unstructured tasks in various computing environments. In order to reduce the computational cost of the recognition process, they reduced the search time by clustering the collected data using a variation of the maximin-distance algorithm. The typing speed, or number of words typed per minute, in a given profile was used as the clustering criteria. An obvious limitation of using such an approach for clustering is the need to re-cluster the data as the system is used. Several distance measures were used to compute pattern similarities and dissimilarities including Normalized Euclidean distance, and weighted and non-weighted maximum probability measures. The reference profile was constructed by computing the mean and standard deviations for the timing samples. Although Monrose and Rubin obtained a 90% correct classification for fixed text detection, they obtained at best (using the weighted probability measure) only a 23% correct classification for free text detection.

In the approach adopted by Dowland *et al.* [3] typing samples were collected by monitoring users during their regular computing activities, without any particular constraints imposed on them. A user profile was determined by calculating the mean and standard deviation of digraph latency and by considering only the digraphs occurring a minimum number of times across the collected typing samples. By collecting and analyzing data for five users, they achieved correct acceptance rates in the range of 60%.

Gunetti and Picardi used the degree of disorder of an array to discriminate between two different typing samples in free text [4]. The degree of disorder of an array with respect to its ordered counterpart is defined as the sum of the distances between the positions of each of its elements and the position of the same element in the ordered array.

User authentication was performed by computing the mean distances of the provided sample from the profiles of all the legal users. A necessary condition for the sample to be recognized as belonging to the claimed identity was that the corresponding mean distance be the smallest compared to the mean distances of the sample to all the other legal users profiles. So, authentication depended not only on the claimed identity but also on the other legal users.

Experimental evaluation was conducted by asking users to enter their login identifier and then freely type some text through a web-based interface. Participants consisted of 40 users considered as legal users who provided 15 typing samples each, and 165 users considered as imposters who provided one sample each. The average length of the samples varied from 700 to 900 characters. Overall, they achieved a FAR of 0.00489% and a FRR of 4.8333%, and an equal error rate (EER) of 1%. Villani *et al*. developed a long-text-input keystroke biometric system, targeting applications such as identifying perpetrators of inappropriate e-mail communi-

cation or fraudulent Internet activity [8]. They conducted an experiment involving 118 subjects, which was divided into 6 experimental groups according to the type of keyboard used and the performed task (copy task or free text). Their results showed a wide variation; the calculated accuracy ranged from 44.2% when the enrollment was done on a laptop with fixed text and the testing was done on a desktop with free text, to 100% when a copy task was performed in both the enrollment and testing phases on a laptop (using fixed text).

B. Discussion

It appears from the above summary that except the work by Gunetti and Picardi [4], all the free text analysis schemes proposed in the literature (to our knowledge) are limited by their high error rates [3], [5] and [8]. However, a key limitation of Gunetti and Picardi's approach is the reliance on other users for the construction of a user profile in the identity verification step for a given sample. In their approach, authentication requires checking the received sample against the profiles of all the legal users before a final decision can be made. This raises some serious efficiency and scalability issues. Ideally, authentication will involve checking a received sample against only the claimed identity. Involving the other legal users in the verification process can negatively impact performance and scalability in large networks. This may pose some issues in continuous authentication applications where computation in real-time is required. In the experiments conducted by Gunetti and Picardi, the comparison of a new sample against 40 profiles containing 14 samples each takes about 140 s on a Pentium IV at 2.5 GHz.

In our approach, not only are the accuracy results obtained in our evaluation comparable to the results achieved in [4], but authentication depends solely on the claimed identity. The training that takes place during the enrollment phase depends solely on samples received from the user being profiled. Overall, both the enrollment and verification involve a relatively small amount of computational power and processing time.

As mentioned above, a key characteristic of keystroke analysis is the small amount of information conveyed by the keystroke data. This is even more of a concern in the free text analysis of keystrokes, where quite often unseen or unexpected behavioral patterns must be established or predicted based on available information.

To deal with this situation, the most straightforward approach, which is the one adopted by existing free text analysis schemes, consists of using a large enough sample size when constructing the user profile so as to cover as much discriminating information as possible. Likewise, Gunetti and Picardi's enrollment procedure required a large sample of more than 12 000 digraphs for each enrolled user. Using a large sample size to construct the user profile, however, means that enrollment can take a longer time than what can be afforded in security applications running on the Internet. This means that during the enrollment, an alternative security medium must be used to guarantee the integrity of received data; but in any case, from a security standpoint, the shorter the enrollment period, the better. To deal with this situation, we use a reduced sample size in modeling the user profile and propose a digraph/monograph approximation technique to address the created information gap; our technique can enroll a user with a sample size of 1500 digraphs. The proposed approximation technique allows predicting missing digraphs. Furthermore, the enrollment does not depend on other user profiles and relies solely on samples from the user being profiled; so detection can start with only one enrolled user. These features make our technique more flexible and allow for a significant reduction in the enrollment period compared to existing approaches.

III. Behavior Modeling

Like all other biometric systems, our proposed system operates in two modes: enrollment and verification. In this section, we focus on the enrollment mode while the subsequent section is dedicated to the identity verification mode. More specifically, we start by discussing the challenges underlying free text enrollment and then give a general overview of the enrollment process and architecture, followed by a more detailed description of the proposed approach, which consists of approximating missing values by combining a key mapping technique with neural network analysis. The approach is described by presenting successively digraph signature modeling, monograph signature modeling, and corresponding neural network architectures.

A. Enrolment Challenges and Process

The enrollment process in any keystroke dynamics analysis system requires the existence of an enrollment sample, which is used to compute the behavioral profile of the user. This sample should contain all possible key combinations in order to recognize effectively the user based on an expected or unexpected set of user inputs. The first challenge facing a free text analysis system is the ability to enroll a user using a non-predefined set of data. This allows for non-intrusive monitoring of the user on-the-fly by totally hiding the recognition process from the user. Another challenge is to be able to provide information about a user identity based on a minimal amount of non-deterministic input. This is very important in practice as in most cases it will not be possible to force an attacker to provide more data and a decision should be made based only on the data available. Ideally, the enrollment sample must cover all expected keys. In addition, previous implementations of keystrokes dynamics required the coverage of all expected digraphs. The occurrence of a non-expected digraph in the provided data (or monitored session) will be ignored. We propose in this paper a new digraph approximation technique referred to as the sorted time mapping (STM) technique, which removes this limitation from the enrollment process. The technique assumes that it is possible to enroll the user by covering the most frequently occurring keys (or most frequently occurring monographs), but not all the expected digraphs. It further assumes that it is possible to approximate the remaining digraphs based on the relations between the monitored ones. A particular characteristic of free text analysis is that data may be collected from a variety of keyboards. So to study coverage, we need to look broadly at the collective data from the different types of keyboards involved. In this

regard, let K and D denote the set of all unique keys and the set of all possible digraphs, respectively. We define the set of most frequently occurring keys, denoted K^p, as the set of all the keys with probability of occurrence greater than or equal to a threshold p, where p is a number such that $0 \leq p \leq 1$. So, $K^p = \{k \in K | P(k) p\}$, where $P(k)$ represents the occurrence probability of key k. Our general approach consists of selecting the enrollment sample in such a way that 100% of K^p is covered, with no particular requirement on the amount of digraphs being covered. Infrequent keys (i.e. outside K^p) missing from the enrollment sample, which occur in the monitored data, will be ignored, and so will be any digraph that contains at least one such key. Missing digraphs containing keys occurring in the enrollment sample will be approximated using a key mapping technique combined with a neural network to determine the expected timing information for the digraphs. The key mapping technique is used to prepare the data for the neural network. The neural networks approximate the missing digraphs and model the user behavior based on the encoded set of digraphs and monographs.

The key mapping technique sorts the key codes based on associated average time, and accordingly maps them in corresponding order. The sorting process is important as it assists the neural network in building and approximating the relation between the keys based on the behavioral distance between them expressed in time. Without this mapping it would not be possible to model the relation, since the actual key code does not reflect such a relation.

Data presented as input to the neural network should be conditioned so that the neural network is able to understand the relation between the inputs and model the behavior. Neural networks can fail to converge, or may provide an over fitted approximation, if the inputs are not properly presented. The sorting technique will encode the keys according to how fast they are accessed by the user. Such encoding will help the neural network to generalize and properly model the relation between the digraphs.

Fig. 2 illustrates the enrollment process. Raw data collected from a specific user's sessions are processed and converted to monograph and digraph formats. Also, at this stage, outliers are removed from the mono and digraph sets before passing the data to the next stage. Outlier removal occurs only on the enrollment data; no outlier removal happens for the test data. We use Peirce's criterion for the elimination of outliers. Peirce's criterion is a rigorous method based on probability theory that can be used to eliminate outliers in a rational way. It consists of an iterative process which starts with calculating the mean and standard deviation for the data set. In the first iteration an assumption is made that the data has only one doubtful observation. Based on that assumption and using the Peirce's table, the maximum allowable deviation is calculated and all entries with greater deviations are then eliminated from the dataset. The number of doubtful observations will be incremented by one in each of the next iterations. In each of these iterations, the maximum deviation will be recalculated while maintaining the same mean and standard deviation of the original dataset. The process will be repeated until no more entries are eliminated (see [7] for more details about this criterion). After removing outliers, generated monographs and digraphs are sent as a batch to the monograph and digraph sorting modules. Each of those modules will process the data and calculate a mapping table. Calculated monograph and digraph mapping tables are considered part of the user's signature and stored for future use.

The next step of the enrollment process consists of training a neural network with processed monographs and digraphs after applying the mapping orders calculated in the previous step on the monograph and digraph sets. Two neural networks (Mono and Di) are trained for each of the enrolled users. The weights of the trained networks are considered part of the user's signature and are stored for future use during the identity verification process.

B. Digraphs Signature

The digraph mapping process involves sorting key codes based on average fly times for related digraphs, leading to what we call in this paper the digraph mapping table. Let K_e and Z_+ denote the set of all key codes in the enrollment sample and the set of nonnegative integers, respectively. The digraph mapping process is defined as a function DKO: $K_e \rightarrow Z_+$, which returns for a given key code a rank based on the average fly time for all digraph entries containing that key in their to field. In the digraph mapping table, key codes (not digraphs) are sorted in ascending order according to the average fly times to these keys. Given a key, the average fly time to that key (converted into key order) represents how accessible the key is to the hands of the user compared to the other keys on the keyboard. Based on this consideration, we use the digraph key order as an intrinsic characteristic of a key that is used to approximate all digraphs containing that key, regardless of the position of the key in the digraphs (to or from).

For example, to illustrate the mapping process, let us assume that $K_e = A, B, C, D$, and that the enrollment sample consists of the following text *DACBACDDAADBA*. The list of digraphs provided in the enrollment sample is $\Delta = \{DA, AC, CB, BA, AC, CD, DD, DA, AA, AD, DB, BA\}$.

The digraphs $\{AB, BB, BC, BD, CA, CC, DC\}$ are not covered in the enrollment sample and need to be approximated. The approximation process starts by calculating the average fly-times for all of the digraphs in the provided sample. The digraph approximation process is summarized in Fig. 3. In the example, the enrollment sample is described as a 4×4 digraph time matrix, in which the rows and columns are labeled by the from and to keys, respectively.

Each cell in the matrix contains the fly times for the different occurrences of the corresponding digraph. For instance, there are two instances of digraph AC with fly times 65 ms and 75 ms, respectively. There are several empty cells, which correspond to missing digraphs that must be approximated. The next step in the approximation process is to calculate for each key β in the enrollment sample the average fly time of all the digraphs ending with that key, which is denoted $t_f^{?\beta}$. In our example, we have $\left\{t_f^{?A} = 110\,\text{ms}, t_f^{?B} = 130\,\text{ms}, t_f^{?C} = 70\,\text{ms}, t_f^{?A} = 170\,\text{ms}\right\}$. The digraph key orders (DKO) for the monitored keys are

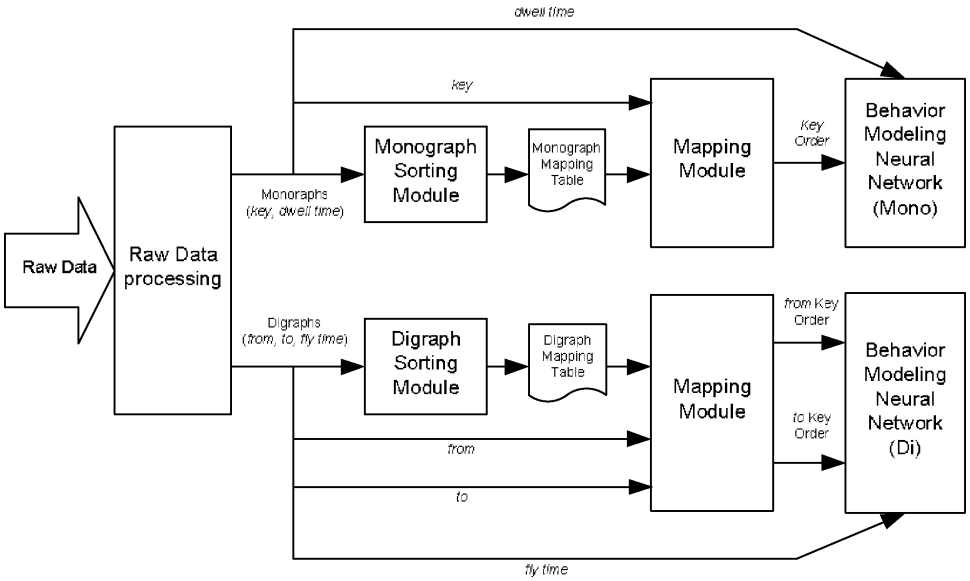


Fig. 2. User enrollment process. The user profile is built by extracting and processing concurrently monographs and digraphs.

obtained by sorting the above fly times, giving for the example DKO = 2, 3, 14..

The fly time for a missing digraph is obtained from the output of the trained digraph neural network (Fig. 2). The digraph neural network takes as input the DKO for the to and from keys of a (missing) digraph and then returns as output an estimate of the corresponding average fly time. For instance, the fly time for the missing digraph *AB*, for the enrolled user, can be approximated as $t_f^{AB} = Di\,(\mathrm{DKO}\,(A)\,,\mathrm{DKO}\,(B)) = Di\,(2,3)$.

The digraph neural network is trained with the actual fly times of the digraphs instances involved in the enrollment sample. Table I shows the training data set in the above example.

The DKO and the weights of the digraph neural network are stored as the digraph signature for the user. The digraph neural network will be covered in detail in Section III-D. To demonstrate the effectiveness of the digraph approximation technique, we studied the changes of the approximation matrix as a result of removing a specific digraph from the enrollment sample. The approximation matrix is constructed by presenting all possible digraph combinations to the trained digraph neural network and tabulating their corresponding approximated fly times. Table IIa shows a 4×4 subset of the digraph matrix of an arbitrary user from the free experiment described (below) in Section V. Each cell in this table shows the fly time for the corresponding digraph. These times are calculated after training the digraph neural network with the digraphs provided in the user enrollment sample (consisting of 1500 keystrokes). The fly times for the digraphs occurring in the enrollment sample are marked in bold. The rest of the fly times marked in normal font are for digraphs that are missing from the enrollment data. The neural network approximates the fly times of these missing digraphs based on their relations to other digraphs. For example for digraph (310, 470) which occurred in the user's training data, the approximated fly time is 108.17 ms. For missing digraph (310, 60), the fly time has been estimated to be 113.57 ms. Next, we removed all instances of digraph (310, 470) from the enrollment data and retrained the digraph neural network. Table II(b) shows the approximated fly times. The fly time of digraph (310, 470) is now approximated to be 116.09 ms instead of 108.17 ms (i.e. a 6.82% difference from the previous approximation). The table also shows the approximated fly times for the rest of the digraphs. The deviations between corresponding digraphs in Tables IIa and IIb are relatively low. We repeated this process again by excluding digraph (310, 771) and similar results were found. Table IIc shows the approximated fly times. Fly time for digraph (310, 771) was approximated to be 119.12 instead of 110.4 in Table IIa. For other digraphs the differences from results in Table IIa are low as well. Table III shows the digraph mapping table generated from the sorting module for two different users selected arbitrarily from the free experiment presented later in Section V. The table shows the first 60 keys in each user's mapping table, as well as the key codes and the key order assigned to them after the sorting process. For example, key code 280 is mapped to key order 37 for User 1, and at the same time is mapped to key order 46 for User 2. We can see a big difference in the mapping results between the two users.

C. Monographs Signature

As mentioned above, our monograph analysis approach considers only the keys present in the enrollment sample. Unlike digraph analysis, all missing monographs will be ignored during the analysis (as well as any digraph that contains such keys). In this paper, we use a representation for monographs which mirrors the approach used for digraphs.

This allows us to provide a homogeneous global monograph/digraph signature for the user. The main idea behind this approach is to model the user behavior by exploiting an underlying relation between the keys, which in the case of monographs is their ranking based on average dwell times.

TABLE I

DIGRAPH NEURAL NETWORK TRAINING DATASET FOR THE EXAMPLE IN FIG. 3

Input	(2,2)	(2,1)	(2,1)	(2,4)	(3,2)	(3,2)	(1,3)	(1,4)	(4,2)	(4,2)	(4,3)	(4,4)
Output	50	65	75	150	150	110	150	200	100	140	110	160

TABLE II

4 X 4 SUBSET OF THE APPROXIMATION MATRIX FOR AN ARBITRARY USER. TO AND FROM KEYS ARE REPRESENTED BY THEIR KEY CODES (IN ITALIC). THE CELLS CONTAIN THE FLY TIMES IN MILLISECONDS FOR DIGRAPHS WHICH ARE PRESENT AND MISSING FROM THE ENROLLMENT SAMPLE, MARKED IN BOLD AND NORMAL FONTS, RESPECTIVELY. (A) INCLUDING ALL (ORIGINAL) ENROLLMENT DATA IN THE NEURAL NETWORK TRAINING. (B) EXCLUDING DIGRAPH (310 470) FROM THE TRAINING. (C) EXCLUDING DIGRAPH (310 771) FROM THE TRAINING

(a)

To / From	*470*	*771*	*60*	*140*
310	**108.17**	**110.4**	113.57	**145.04**
731	**150.85**	**139.39**	134.7	**153.38**
340	191.23	184.21	182.81	195.78
320	222.38	**224.51**	230.73	244.3

(b)

To / From	*470*	*771*	*60*	*140*
310	116.09	**111.1**	115.5	**145.33**
731	**150.81**	**140.2**	133.1	**153.2**
340	196.33	188.6	183.5	193.2
320	225.7	**224.92**	231.4	244.7

(c)

To / From	*470*	*771*	*60*	*140*
310	**108.8**	119.12	118.2	**147.22**
731	**150.94**	**142**	136.2	**154.35**
340	192.2	187.46	185.23	197.9
320	222.63	**224.2**	231.1	244.22

The user behavior for monographs is a 2-D relation between the key order and its dwell time. Although, a simple table will be enough to express this relation, the role of the neural network in our model is to ensure the consistency of the adjacent readings and the smoothness of the whole curve. In this case, the weights of the trained neural network will be stored as part of the user's signature instead of storing the whole monograph table.

The monograph mapping table is similar to the digraph table; however, the sorting order is different. Similar to the digraph mapping, the monograph mapping process is defined as a function MKO: $K_e \rightarrow Z_+$, which returns for a given key code a rank based on corresponding monograph timing characteristics. To calculate the monograph mapping table, the average dwell time will be calculated for all entries of each unique monograph (or key code) in the provided batch.

TABLE III

DIGRAPH MAPPING TABLES FOR TWO DIFFERENT USERS. NOTE THAT THE MAPPING CONSISTS OF THE (KEY ORDER, KEY CODE) PAIR; THE FLY TIME COLUMN IS PROVIDED HERE JUST FOR INFORMATION PURPOSE

User 1

Key Order	Key Code	Average Fly Time (ms)	Key Order	Key Code	Average Fly Time (ms)
1	801	4.1466	31	30	106.04
2	420	4.7788	32	720	155.66
3	360	5.6749	33	110	229.21
4	160	7.0099	34	751	331.27
5	440	7.1626	35	140	462.08
6	450	8.6244	36	330	617.33
7	791	9.4897	37	**280**	790.24
8	460	10.073	38	820	974.01
9	80	11.521	39	760	1162.7
10	90	12.298	40	500	1351.1
11	180	12.978	41	350	1533.6
12	230	14.454	42	390	1704
13	750	15.558	43	721	1855.2
14	200	15.954	44	400	1981
15	771	17.488	45	20	2078.8
16	190	19.059	46	510	2150.8
17	370	19.471	47	580	2202.7
18	300	20.674	48	170	2240.7
19	240	22.335	49	40	2270
20	480	24.045	50	50	2294.1
21	260	24.377	51	831	2315.3
22	150	25.808	52	60	2334.8
23	520	30.862	53	120	2353.2
24	250	39.94	54	210	2370.8
25	130	53.341	55	470	2387.7
26	730	73.916	56	310	2404.1
27	810	75.3	57	711	2419.9
28	340	77.106	58	490	2435.1
29	800	95.23	59	320	2449.7
30	380	100.2	60	220	2463.9

User 2

Key Order	Key Code	Average Fly Time (ms)	Key Order	Key Code	Average Fly Time (ms)
1	170	24.102	31	450	33.291
2	480	24.161	32	721	35.359
3	360	24.448	33	400	35.522
4	350	24.53	34	140	37.935
5	760	25.117	35	730	40.171
6	800	25.296	36	801	41.07
7	440	25.836	37	30	44.816
8	340	26.614	38	70	45.772
9	470	26.741	39	330	49.222
10	130	27.385	40	200	54.34
11	791	28.096	41	40	60.219
12	300	28.691	42	80	66.908
13	60	28.707	43	240	74.457
14	810	28.753	44	160	82.912
15	490	28.782	45	390	92.319
16	831	28.874	46	**280**	102.72
17	120	28.92	47	110	114.15
18	90	29.076	48	580	126.66
19	310	29.145	49	500	140.27
20	380	29.193	50	220	155.01
21	750	29.385	51	210	170.91
22	720	29.538	52	190	187.97
23	570	29.598	53	771	206.2
24	820	29.623	54	250	225.61
25	510	29.739	55	260	246.17
26	20	29.748	56	751	267.88
27	460	29.803	57	520	290.7
28	320	30.474	58	420	314.59
29	230	31.68	59	180	339.49
30	370	31.776	60	790	365.34

Enrollment sample with missing digraphs

To / From	A	B	C	D
A	**50**		**65** **75**	**150**
B	**150** **110**			
C		**150**		**200**
D	**100** **140**	**110**		**160**

1. Compute Average Fly Time for Digraphs ending with given character

To Key	A	B	C	D
Fly Time	110	130	70	170

2. Compute Digraph Key Order (DKO) by sorting the average fly times

DKO	2	3	1	4

3. Estimate missing digraphs using DKOs as inputs to digraph neural network (Di)

Enrollment sample with estimates of missing digraphs

To / From	A	B	C	D
A	**Di(2,2)**	Di(2,3)	**Di(2,1)**	**Di(2,4)**
B	**Di(3,2)**	Di(3,3)	Di(3,1)	Di(3,4)
C	Di(1,2)	**Di(1,3)**	Di(1,1)	**Di(1,4)**
D	**Di(4,2)**	**Di(4,3)**	Di(4,1)	**Di(4,4)**

Fig. 3. Digraph approximation process. The cells in the first matrix (top) contain the actual fly times in milliseconds of the different occurrences of corresponding digraphs.

The monographs will be sorted according to their average dwell times yielding the monograph key orders (MKO). For instance, considering the previous example, if we assume that the average dwell times in milliseconds of the keys in K_e = A, B, C, Dare 7, 15, 8, 9, respectively, we will have MKO = 1, 4, 2, 3.

The monograph neural network is trained using the individual dwell times of the specific occurrences of each of the keys in K_e. The neural network takes as input the MKO of a given key and returns as output the corresponding dwell time. The MKO and the weights of the monograph neural network are stored as part of the user's signature. Table IV shows the monograph mapping table generated from the sorting module for the same two users whose digraph mapping is presented in Table III. Since the key codes in Table IV are sorted incrementally according to dwell time, we expect to see a different sorting order for the same user when compared to his own digraph mapping table (Table III). For instance, key code 280 for User 1 is mapped to key order 10 and 37, in the monograph and digraph mapping tables, respectively. Similar to Table III, we can see a clear difference in the monograph mapping results between the two users.

D. Enrolment Neural Networks

We use neural networks to model the user behavior based on the encoded sets of monographs and digraphs. Although the neural network architecture remains the same for all users, the weights are user specific. We describe, in this subsection, the proposed neural network architecture and provide a visual representation of the monograph and digraph signatures produced in order to illustrate the similarity or dissimilarity in behavior for different users' sessions.

1) *Neural Network Architectures:* Given an *n-graph* $k_1 \ldots k_n$ received during a computing session, let M denote a function that returns the order of a key code as defined in the mapping table associated with the claimed identity profile (i.e. M generically represents DKO or MKO). The expected time T for the received sample is computed as

$$T = \left(\left(\sum_{j=1}^{N_h} w_{2j} \cdot \frac{1}{1 + e^{\left(\sum_{i=1}^{n} w_{ij} M(k_i) \right) - b_{1j}}} \right) - b_{21} \right)$$

where $M(k_i)$ (the order of key code k_i) represents the inputs to the neural network. w_{lj} and b_{lj} represent, respectively, the weight and bias of node j of layer l ($l = 1$ for the hidden layer, $l = 2$ for the output layer). Finally, n and N_h are the numbers of nodes in the input and the hidden layers, respectively. In our study, we focus only on monograph and digraph analysis, so $n = 1$ and 2, respectively. The monograph behavior modeling network will be trained with the mapped keys as an input and the corresponding dwell time as an output. For the digraph neural network, mapped (from, to) key combinations are sent as inputs to the network, while the fly time is used as the training output. Fig. 4(a) and (b) shows the neural network architectures for the monograph and digraph networks, respectively. Both networks are feed forward, multilayer perceptrons (MLP). We use the Levenberg–Marquardt algorithm for training (using MATLAB v7.0) in order to expedite the learning process, and a split-sample validation technique to investigate various neural network configurations based on a selected subset of available data (10 users from the experiments described in Section V, 20 000 keystrokes each). Based on this investigation, we are able to conclude that the following architectures are sufficient for producing excellent results:

1) The monograph network consists of two layers with (n =) 1 input node representing the mapped key code;

TABLE IV
MONOGRAPH MAPPING TABLES FOR TWO DIFFERENT USERS. NOTE THAT THE MAPPING CONSISTS OF THE (KEY ORDER, KEY CODE) PAIR; THE DWELL TIME COLUMN IS PROVIDED HERE FOR INFORMATION PURPOSES

User 1

Key Order	Key Code	Average Dwell Time (ms)	Key Order	Key Code	Average Dwell Time (ms)
1	160	2.54	31	510	70.12
2	440	21.66	32	831	70.26
3	40	52.712	33	450	70.3
4	260	58.1	34	400	70.38
5	170	59.23	35	300	70.44
6	110	61.623	36	240	70.51
7	771	63.021	37	420	70.58
8	120	63.28	38	720	71.23
9	721	63.36	39	200	71.33
10	**280**	63.47	40	500	71.74
11	140	63.61	41	580	72.61
12	250	63.801	42	180	74.26
13	70	63.93	43	130	75.37
14	520	64.31	44	460	75.75
15	20	64.36	45	480	76.49
16	760	64.65	46	790	77.23
17	310	64.731	47	730	78.11
18	30	64.8	48	60	78.32
19	490	65.14	49	570	78.79
20	190	65.21	50	791	79.24
21	390	65.71	51	380	79.63
22	750	66.82	52	90	83.51
23	360	67.36	53	340	87.3
24	330	67.51	54	751	89.48
25	801	68.17	55	350	92.22
26	370	68.36	56	220	97.18
27	800	68.92	57	230	100.27
28	810	69.74	58	210	108.61
29	470	69.87	59	80	118.214
30	820	70	60	320	134.49

User 2

Key Order	Key Code	Average Dwell Time (ms)	Key Order	Key Code	Average Dwell Time (ms)
1	360	60.21	31	820	120.61
2	190	70.29	32	800	120.73
3	751	79.4	33	90	121.31
4	370	82.14	34	400	121.5
5	210	85.43	35	300	121.73
6	801	92.9	36	350	122.3
7	810	98.1	37	20	122.82
8	460	100.2	38	340	123.3
9	480	102.71	39	40	124.46
10	771	104.99	40	250	124.571
11	490	107.63	41	320	124.8
12	50	108.33	42	440	124.9
13	510	108.91	43	390	125.26
14	240	109.29	44	791	125.41
15	110	110.4	45	230	125.72
16	80	111.06	46	470	126.7
17	180	112.68	47	520	128.51
18	140	113.12	48	200	130.2
19	160	114.38	49	730	133.47
20	760	115.22	50	450	133.6
21	**280**	116.1	51	711	133.67
22	30	116.42	52	720	133.95
23	60	117.12	53	721	134.82
24	750	117.29	54	420	136.37
25	150	117.78	55	330	136.92
26	120	118	56	380	137
27	500	119.18	57	831	138.21
28	220	119.65	58	580	141.61
29	170	120.2	59	260	144.14
30	310	120.45	60	130	148.47

the output of the network represents the fly time associated with the input key. The hidden layer consists of ($N_h =$) 16 nodes.

2) The digraph network also consists of two layers; the number of input nodes is ($n =$) 2, representing the *from* and *to* mapped keys. The output layer consists of one node, which represents the fly time of the input digraph.

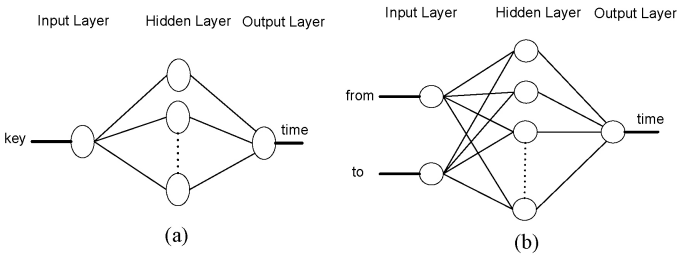


Fig. 4. Neural network architectures. (a) Monograph behavior modeling. (b) Digraph behavior modeling.

The hidden layer consists of ($N_h =$) 40 nodes.

The *Tan-Sigmoid* transfer function was used in all neural network nodes, except the input and output nodes where a linear transfer function was used. For both networks, inputs and outputs are normalized so that they fall in the [0,1] range. Normalization is performed using the MINMAX algorithm. At enrollment stage, the networks were trained with a set of 1500 keystrokes for each user. On a dual Xeon 2.8-GHz server with 2 Gigabytes of RAM, the training of the neural networks takes on average 1.9124 s and 1.1563 s for the monograph and digraph networks, respectively.

2) *Monograph Modeling:* Monograph modeling consists of studying the relation between the pressed keyboard key and the time spent on this key, and comparing this to the time spent on pressing other keys. The monograph neural network tries to model this relation for all monitored keyboard keys. Fig. 5 shows the monograph reference signature for the two users mentioned in Section III-C and the monograph data for one of each user's sessions. Each point on the curve represents the dwell time spent when pressing the corresponding mapped x-axis key code. The data is compared to the two reference signatures.

We note a significant deviation from the reference signature when the data is compared to a nonself [Fig. 5(b) and (d)] and a very low deviation when the user's session data is compared to his own reference signature [Fig. 5(a) and (c)].

The deviation indicates a noticeable difference in behavior. Although the two signature curves are different from each other, we note that both curves have an increasing pattern; this is expected, as keys on the x-axis were already sorted according to their corresponding dwell times. The curves were built based on two different mapping tables (Table IV), which means that the points on the x-axis do not correspond to each other for the two reference signatures.

The same applies for the session data. The session data in Fig. 5(a) and (d) represent the same user session (User 1); however, the points are located differently on the chart as they were sorted with two different mapping tables (User 1's for (a) and User 2's for (d)) so that they could be compared to User 1's and user 2's signatures, respectively.

3) *Digraph Modeling:* Digraph modeling is done by the digraph neural network, which tries to build a relation between the input keys combination and the time needed to fly between them. The network also helps in approximating this time for the newly observed combinations. Fig. 6 shows

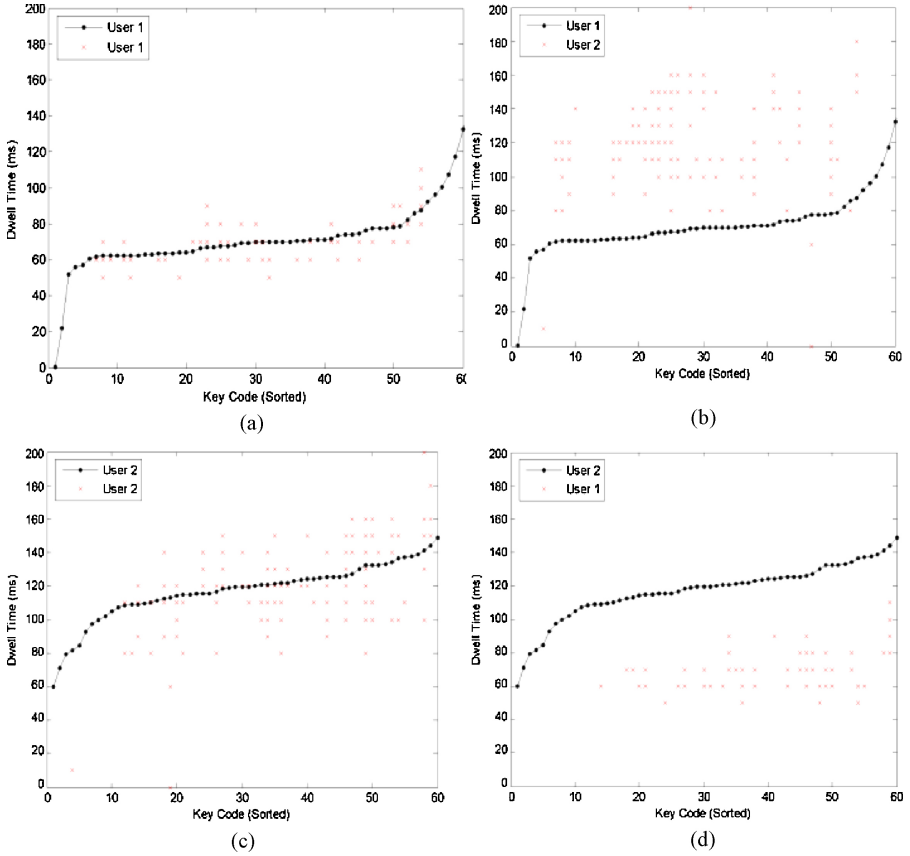


Fig. 5. Comparing session data to a monograph reference signature for two different users. (a) One of User 1's sessions compared to his reference signature. (b) User 2's session compared to User 1's reference signature. (c) One of User 2's sessions compared to his reference signature.(d) User 1's session compared to User 2's reference signature.

digraph signatures for the two users. Each point on the surface represents the fly time from a key on the x-axis to another key on the y-axis. The surfaces were constructed by enumerating all possible (x, y) key combinations [for the first 60 keys in the mapping table (Table III)] and feeding these combinations to the trained digraph neural network for each user. The output of the network is plotted as the z-axis value. We can note the obvious difference between the surfaces as each represents the behavior of a different user.

The network is able to build a relation between the keys and estimate values for the digraph combinations (x, y) that are not included in the training data. The generated surfaces demonstrate this approximation. For all surfaces, we also notice an increasing pattern as the value on the x- and y-axes increases. This pattern, however, does not indicate any similarity as the order of keys on the x- and y-axes for each user is different from the others.

Fig. 7 illustrates digraph comparison. The figure shows the part of the reference signature that is related to the key code 280 for the two users. The curves in Fig. 7(a) and (b) represent the fly time needed for the user to move from the mapped key code on the x-axis to the key 280. This curve is part of the digraph reference signature that is presented in Fig. 6. The points on the figure represent the instances of digraphs ending with the key code 280 in a user session. Fig. 7(a) shows two sessions for two different users compared

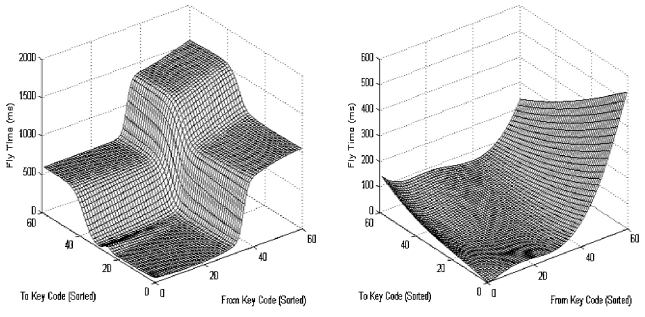


Fig. 6. Digraph reference signatures for two different users.

to the reference curve of one of them; Fig. 7(b) shows the same sessions of the two users compared to the reference signature of the second user. We note the high deviation of the session points from the nonself reference signature and the closeness to the self-reference signature for the two users.

IV. Identity Verification

The user signature resulting from the enrollment mode consists of the mapping tables and the weights of the trained neural networks. This signature is stored and used for the

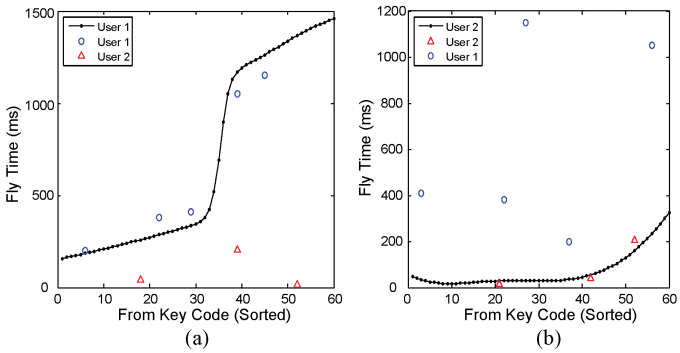


Fig. 7. Comparing session data to digraph reference signature for two different users. (a) Against User 1's reference signature. (b) Against User 2's reference signature.

identity verification process, which is described in Fig. 8. In this mode, the two neural networks (trained in the enrollment mode) will be loaded with the weights stored in the legitimate user's signature. Each monograph and digraph in the monitored session will pass through the corresponding mapping module, which utilizes the mapping table provided by the user's signature. The output of each module will be passed as inputs to the corresponding Mono/Di trained neural network, which will output the dwell/fly time needed for the legitimate user to perform the provided Mono/Digraph action. The differences between those times and the original dwell/fly times of the monitored mono/digraphs represent the deviations from the legitimate user's behavior. The calculated deviations will be passed to the decision unit that is responsible for making the ultimate decision about a user's identity. The decision unit will calculate the absolute average deviation (δ_{mono}, δ_{di}) for the mono and digraph sets, which is described by the following equations:

$$\delta_{\text{di}} = \frac{\sum_{i=1}^{N_{\text{di}}} \frac{|f_i - F_i|}{F_i} \times 100}{N_{\text{di}}}$$

and

$$\delta_{\text{mono}} = \frac{\sum_{i=1}^{N_{\text{mono}}} \frac{|d_i - D_i|}{D_i} \times 100}{N_{\text{mono}}}$$

where N_{mono} and N_{di} are the total number of monographs and digraphs in the session data set, respectively; d_i and f_i are the dwell and fly times for the ith mono and digraph records in the user's session, respectively; D_i and F_i are the dwell and fly times resulting from the user's mono and digraph neural networks, respectively (when the networks are fed, respectively, with the ith monograph and digraph mapped key codes). The value of the average deviation calculated for a monitored session will represent how close the user behavior computed in this session is to the legitimate user's behavior. The lower this number, the more confident the system is that this session belongs to the same user. The monograph and digraph decision scores can be merged using a suitable decision fusion scheme. We use in this paper a linear combination scheme, described by the following equations:

$$O = \begin{cases} \text{true} & \text{if } D\text{thr} \\ \text{false} & \text{Otherwise} \end{cases}, \quad D = B\delta_{\text{mono}} + (1 - B)\delta_{\text{di}}$$

where O is the output of the decision unit, D is the weighted percentage deviation, B is the dependability factor, and thr is the decision threshold limit. The decision unit will classify the user session as belonging to the enrolled user when its output O evaluates to true as a result of achieving D lower than the selected threshold limit. Setting $B = 0.5$ will make the decision unit treat the results of the monograph analysis and the digraph analysis equally. Increasing this number above this value will make the decision unit lean more toward the result of the monograph analysis, and decreasing it below this limit will make it lean more toward the result of the digraph analysis.

V. Free Experiment

In order to evaluate the proposed recognition system, we conducted an experiment involving several participants. We describe in this section the experimental procedures, analyze the data collected[1] and discuss the results obtained.

A. Experimental Setting

1) *Method:* We conducted our experiment in a fully uncontrolled setting, where the participants were asked to deploy our data collection client on their own machine and use their computer for their routine activities. Although they provided informed consent, data collection was performed transparently, and the collected data was sent to our server located in our lab. No restrictions were imposed on them regarding the environment or the kind of application they could use. Unlike in [4] where participants had to type text through a web-based interface, participants in our experiment ran various applications including word processors, video games, web browsers, etc. Participants connected to the server from their home, their office, or from travel locations (using their laptops). Participants in our experiment consisted of 53 individuals—two-thirds male and a third female—with ages ranging from 13 to 48 years, with varying typing skills, ranging from professionals using a computer every day in their daily activity to others who only use computers occasionally.

2) *Apparatus:* Our keystroke dynamics recognition system has been implemented as a client/server application. Data collection is performed by a thin client, which runs transparently in batch mode on the user's machine and lasts for the duration of login sessions. The client monitors keystroke events through windows operating systems APIs. Two types of events are generated, *Key Down* and *Key Up*. Event details include the keyboard scan code which is used to identify the key and the time of the event. Dwell and fly times are calculated by subtracting the times of the corresponding events. The sampling rate for this setup is 1 ms. The collected data is sent over the Internet to the server located in our lab for

[1]The collected dataset is publicly available, and can be obtained at http://www.uvic.ca/engineering/ece/isot/datasets/index.php.

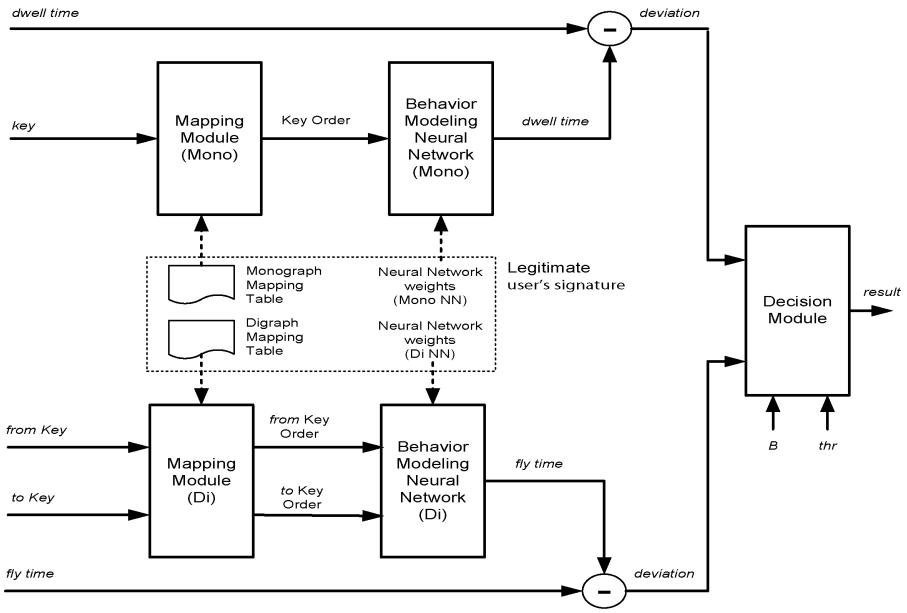


Fig. 8. Identity verification process. Verification is conducted by checking separately the monograph and digraph models and then merging the outcome.

storage and processing. Client machines varied from a Pentium II 266-MHz processor to a Pentium IV 1.5-GHz processor. For this experiment, only a Windows version of the client was delivered to the participants, with different versions of the Microsoft Windows Operating System deployed on client machines including Windows 98SE, Windows ME, Windows 2000, and Windows XP. The server was a Pentium III 450-MHz processor with 256 Mbytes of RAM, running the Windows 2000 operating system, which was connected to the Internet.

3) *Data Collected:* Data was collected over a period of about 5 months. For the duration of the experiment, 53 participants provided a total of 9 544 426 keystrokes with an average of 209 hours of activity per user. Collected data consisted of an average of 18 008 keystrokes per user, with a maximum of 484 827 keystrokes and a minimum of 2222 keystrokes per user.

B. Enrollment Sample Size

By taking a broad look at the nature of the data collected in the free experiment, the number of unique keys found is 107 (set K in Section III-A). The 'space' key achieves the highest probability of occurrence (9.6%) followed by the 'A' key (9.3%). Out of the 107 unique keys, 60 keys have a probability of occurrence greater than 0.1% and 83 keys have a probability of occurrence greater than 0.01%. The number of unique digraphs (set D in Subsection IIIA) is 3959 which represents less than 35% of the set of all possible digraph combinations. Out of the 3959 unique digraphs, 400 digraphs have a probability of occurrence greater than 0.1%, and 900 digraphs have a probability of occurrence greater than 0.01%. The double backspace is the most common digraph achieving 2.85%. As discussed earlier (Section IIIA), the enrollment sample must cover K^p, the set of most frequently occurring keys. In order to determine K^p, we need to set the value of the threshold p. In doing so, the objective is to include as many unique keys as possible while reducing the number of infrequent keys which tend to unnecessarily degrade the performance of the enrollment process. In this paper, we select $p = 0.001$, which as mentioned above, corresponds to the 60 most frequently occurring keys. We computed the key coverage with respect to K^p for the participants in the free experiment. Fig. 9(a) and (b) illustrates the key coverage, with respect to K^p, for three different users arbitrarily selected from the participants. Fig. 9(a) shows, for each user, the relation between the percentages of the number of keys involved in the collected digraphs as the sample size grows. From the figure, we note that the key coverage percentage reaches 97% for the three users at 1100 digraphs and 100% at 1350 digraphs. Fig. 9(b) shows the number of unique digraphs utilized as the sample size increases for the same three users. The figure shows how this number changes when increasing the processed sample size to 8000 actions. The curve follows what visually looks like an exponential distribution, and starts to saturate when the sample size reaches 7000 actions. According to Fig. 9(a) and (b), our selected enrollment sample size of 1500 keystrokes covers 100% of K^p and only a fraction of the digraphs, while the missing ones will be approximated using our proposed technique upon their occurrence. The above enrolment sample size of 1500 keystrokes was later validated over the entire user population involved in the free experiment.

C. Evaluation Results

1) *Main Evaluation:* To evaluate the proposed recognition system, we conducted a leave-one-out cross-validation test. The test was repeated 53 times by considering each time one of the participants in our experiment as a legal user while the remaining were considered imposters. The enrollment process was conducted using the first 1500 digraphs for each user. The remaining digraphs were divided into sets of 500 actions

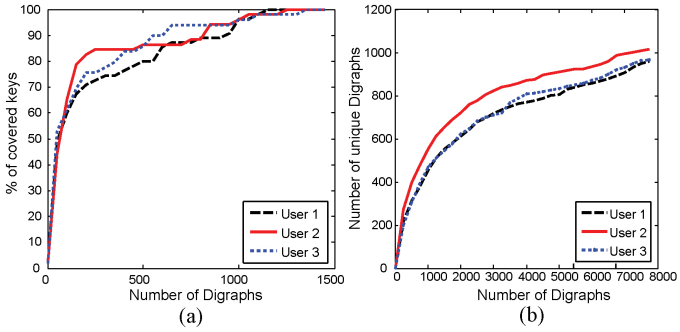
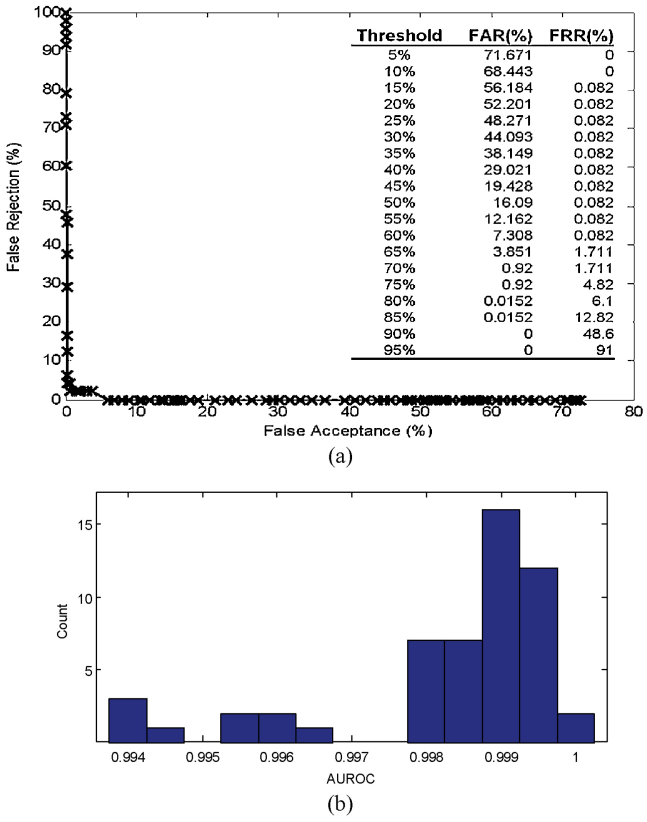


Fig. 9. Key coverage statistics for three different users. (a) Evolution of the key coverage percentages as the sample size increases; 100% coverage is achieved when the sample size is at 1350 and above. (b) Evolution of the number of unique digraphs as the sample size increases; the number of unique digraphs saturates at 7000 sample size and above.

and used to simulate self and imposter sessions.[2] Given a user profile P consisting of his 1500 first digraphs, and S the set of sessions derived from his remaining digraphs, each session $s \in S$ (treated as a new sample) was compared against P and used to attack the remaining users' profiles in our database. It was expected that in the first case the system would recognize the session as belonging to the user, while in the second case the session would be rejected. This process was repeated for each of the 53 users. Fig. 10(a) shows the receiver operating characteristic (ROC) curve for the experiment. The curve shows the relation between the false acceptance rate and the false rejection rate when setting the tuning factor B to 0.5 for an equal dependability on the mono and digraph analysis, and varying the threshold limit from 1% to 100%. The optimal threshold limit can be selected from the curve based on an optimization criterion that involves minimizing the values of FAR and FRR, and considering the relative cost of a false acceptance versus a false rejection. A false rejection is a source of frustration, while a false acceptance represents a security breach. While false acceptance seems to be more important than false rejection, in practice, most organizations are primarily concerned by the latter, since denying access to legitimate users can be damaging for the organization image or disruptive for their operations. When giving FAR priority over FRR, for the given curve, such criteria was achieved when setting the threshold limit to thr = 77%. At this point the optimal performance achieved by our detector was a FAR of 0.0152% and a false rejection rate FRR of 4.82%. The equal error rate (EER) was calculated as 2.46%. To demonstrate the accuracy of the recognition system for the individual users, we compute the area under ROC curve (AUROC) for each user. AUROC corresponds to the probability that the biometric system will assign a higher score to a randomly chosen genuine sample than to a randomly chosen imposter sample. AUROC values close to 1.0 indicate that the system has high accuracy. Fig. 10(b) shows the histogram of the AUROC for the user population

[2]Based on the collected data, to generate 500 and 1500 keystrokes, it takes, respectively, on average 2 min (with standard deviation 0.5 min) and 7 min (with standard deviation 0.8 min).



Threshold	FAR(%)	FRR(%)
5%	71.671	0
10%	68.443	0
15%	56.184	0.082
20%	52.201	0.082
25%	48.271	0.082
30%	44.093	0.082
35%	38.149	0.082
40%	29.021	0.082
45%	19.428	0.082
50%	16.09	0.082
55%	12.162	0.082
60%	7.308	0.082
65%	3.851	1.711
70%	0.92	1.711
75%	0.92	4.82
80%	0.0152	6.1
85%	0.0152	12.82
90%	0	48.6
95%	0	91

Fig. 10. (a) Receiver operating characteristic curve for the main experiment and sample points. (b) Histogram of the AUROC for the participants of the main experiment.

in the free experiment.[3] For this experiment, the AUROC range from 0.994 to 1.0 with most of the users falling close to 0.999 which indicates good performance. Various methods have been proposed in the literature to assess the confidence of biometric systems [12], [13]. According to Bengio and Mariethoz [13], standard statistical tests cannot be used to measure the statistical significance of person authentication models, because several of the performance metrics used to assess those models, such as EER or half total error rate (HTER) are aggregates of two measures (e.g. FAR and FRR). Using the method proposed by Bengio and Mariethoz [13], the confidence interval (CI) is calculated by approximating the distribution of the number of errors to a normal distribution with standard deviation σ. We used the HTER formula to calculate the CI. The HTER was found to be 2.4176% with $\sigma = = 0.0015$. The confidence intervals calculated around this HTER were found to be ± 0.242, ± 0.2883, and ± 0.379 for three different confidence levels of 90%, 95% and 99%, respectively.

2) *Effect of the dependability factor:* The parameter B, introduced in Section IV was used to balance the dependency on each of the monograph and digraph-based evaluation techniques. Prior to the selection of the value of this parameter, it was very important to study its effect on the accuracy of the system. In order to study the effect of this parameter,

[3]The AUROC values are computed for standard ROC curve plotting the true acceptance rate (TAR = 1- FRR) against the FAR.

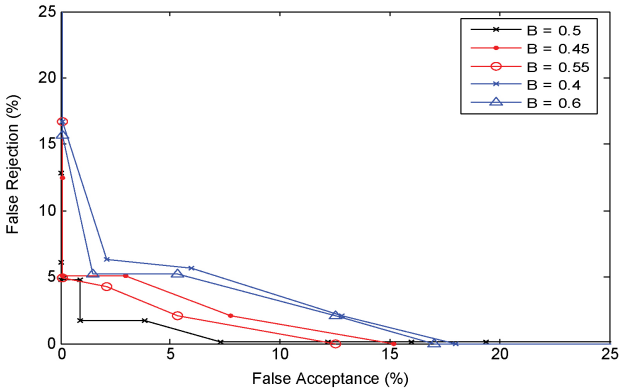


Fig. 11. ROC curves for the five tests showing the effect of the dependability factor B on the accuracy of the system.

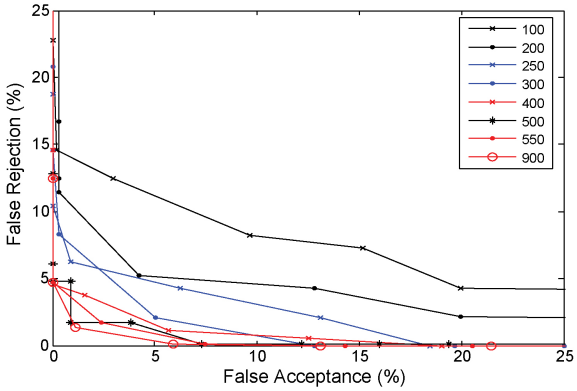


Fig. 12. ROC curves for the eight tests, where the number of keystrokes in the monitoring session increases from 100 to 900 keystrokes, showing how FAR and FRR vary as the threshold value thr varies.

we performed the evaluation process on the data collected in our main experiment for all 53 users. We applied the same evaluation technique described in the previous section, and repeated this process five times with different values for the parameter B, varying from $B = 0.4$ to $B = 0.6$. Fig. 11 shows the ROC curves obtained for the five tests. We note that for the three tests ($B = 0.45$, $B = 0.5$, and $B = 0.55$) it was possible at a specific thr value to obtain a FAR which is very close to zero at low value for FRR ranging from 4.82% to 5.12%. When B gets lower ($B = 0.4$) or higher ($B = 0.6$) than this range, FAR starts to increase reaching 2.08% with the same FRR range. We can also notice the closeness of the pairs of curves where ($B = 0.4$ and $B = 0.6$) and ($B = 0.45$ and $B = 0.55$), indicating a very close effect for both of the monograph and digraph analysis models.

3) *Effect of Session Length:* As illustrated in Fig. 9(b), the number of unique keys increases as the sample size increases. As this number increases, the detection process should be able to capture more behavioral information of the user performing the task. It is important to study the effect of the sample size on the accuracy of the detection technique to determine the safest possible limit that can be selected and estimate the accuracy prior to the selection. To perform this study, we considered the data collected in our main experiment for the 53 users employing the same evaluation process described above. The users' data was divided into a number of sessions, each consisting of n keystrokes. To study the effect of the session length on the detection accuracy, we considered 8 different values for n ranging from $n = 100$ keystrokes to $n = 100$ keystrokes. FAR and FRR were calculated for each test by varying the threshold limit thr and setting $B = 0.5$. Fig. 12 shows the ROC curves for the eight tests. The first three tests ($n = 100$, $n = 200$ and $n = 250$) yield high FAR and FRR values for all threshold (thr) values. The lowest FAR (close to 0) is achieved when FRR falls in the range of 10% to 15%; this range decreases as the number of keystrokes in the session increases. The value of FRR reaches 7% for $n = 300$ and saturates at 4.82% when n becomes greater or equal to 400. The curve starts to achieve lower EER as n increases beyond this limit. Accordingly, we can conclude that the choice of $n = 500$ in our evaluation will achieve the aimed goal of decreasing FRR, as FAR tends to zero.

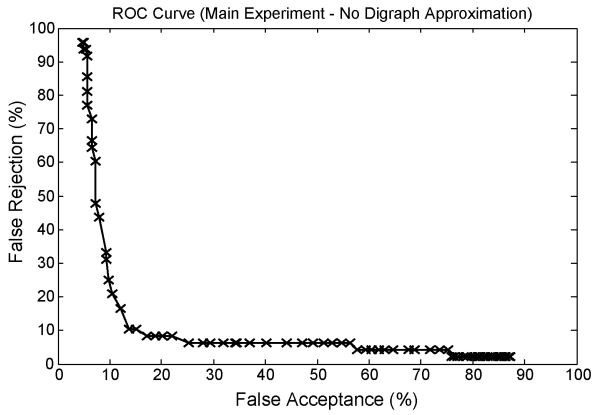


Fig. 13. ROC curve for the main experiment with no digraph approximation.

4) *Impact of Digraph Approximation:* To evaluate the impact of the digraph approximation, we repeated the main evaluation presented above by following exactly the same conditions but with no digraph approximation in effect. In this case, instead of processing all digraphs in a monitoring session, all digraphs that have no exact match in the enrollment data are dropped and not included in the calculation. Fig. 13 shows the corresponding ROC curve. As the figure shows, and compared to Fig. 10(a), removing the digraph approximation results in a significant degradation of the overall system accuracy.

VI. Controlled Experiment

The use of different applications by different participants in the previous (free) experiment represents a confounding factor that needs to be further investigated. In order to address the impact of confounding factors, we conducted a controlled experiment in which all the participants were asked to perform the same task, using the same application, running in the same environment, on the same machine. In this section, we describe the experiments and analyze obtained results.

A. Settings

In this experiment, all the participants were assigned the same task consisting of reading and sending their emails using Webmail. All the participants used the same Webmail application, specifically *SquirrelMail version 1.4.8-5.el4.centos.3* by the SquirrelMail Project Team. The participants were all from the same university, consisting of undergraduate and graduate students, as well as professors, all of whom were very used to typing on standard computer keyboards. All the participants used the same machine, an IBM Thinkpad Laptop (3 GHz Pentium 4 processor, with 1 GB of RAM, and running Windows XP), on which the data collection client was deployed. The collected data was sent through the network to the same server machine that was used and described in the previous experiment. Although the same client program was used as in the previous experiment, a small user interface was added on top of it consisting of both "Start" and "Stop" buttons. To start collecting data, the participant clicked on the "Start" button on the client user interface, opened a browser and accessed Webmail after authenticating herself. She could then read and/or send emails as long as necessary, but before moving on to other jobs, she was required to suspend the data collection process by clicking on the "Stop" button. This allows avoiding contaminating the collected data. The amount of data provided by each user was monitored throughout the data collection process. Participants were asked to continue using the system until a 5500 keystrokes limit was reached, where the machine was passed to the next participant. The experiment lasted for 81 days, and involved 17 participants who used the system for periods of time ranging from two to nine days per user.

B. Results

The same evaluation method, described in section V, was performed on the collected data. A leave-one-out cross-validation test was performed. The same amount of enrollment data (1500 actions) was used to enroll each user. The remaining data (8 sessions, 500 actions each) was used to simulate self and imposter sessions. Similar to Fig. 10(a), the dependability factor for this test was set to 0.5 and the threshold limit was varied from 1% to 100%. Fig. 14(a) shows the receiver operating characteristic curve obtained for the controlled experiment. We obtained an EER of 2.13%. According to the figure, the optimal performance can be achieved at threshold thr in the range of 63% to 74%, in which case we obtain FAR = 0% and a FRR = 5.01%. The half total error rate was calculated as HTER = 2.505% with $\sigma = 0.0084$. The confidence intervals calculated around this HTER were found to be ± 1.3762, ± 1.6397, and ± 2.2155 for three different confidence levels of 90%, 95% and 99%, respectively. Fig. 14(b) shows the histogram of the AUROC for the individual users. For this experiment, the AUROC values range from 0.9945 to 0.9995 which is very close to the results shown in Fig. 10(b). About 40% of the population falls in the 0.999 range, and the overall distribution still indicates good performance. The closeness of these values to the results obtained in our free experiment (in Section V) suggests that by fixing the operating environment and application, our proposed detection scheme can still be used to discriminate effectively between different users.

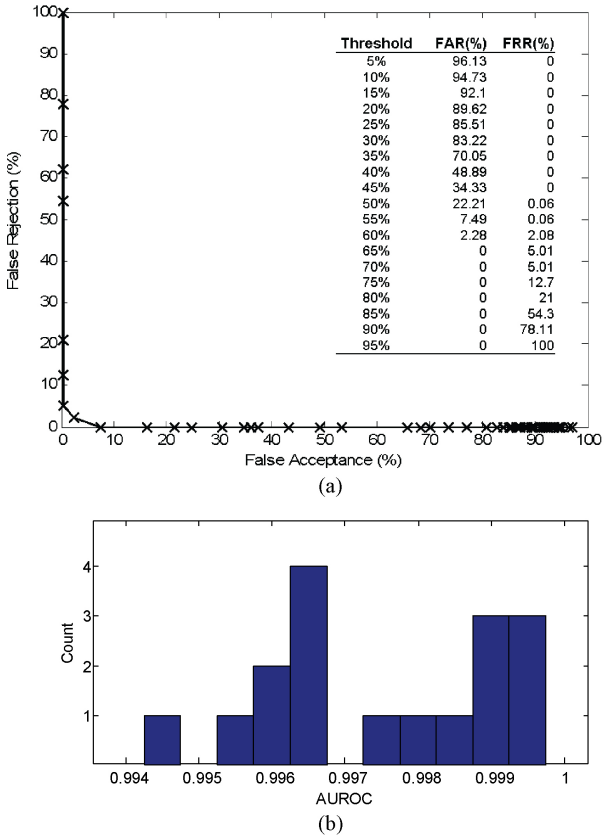


Fig. 14. (a) Receiver operating characteristic curve for the controlled experiment and sample points. (b) Histogram of the AUROC for the participants of the controlled experiment.

VII. Conclusion

In this paper, we presented a new detection technique for free text keystroke dynamics that combines monograph and digraph analysis, and neural networks. The results obtained from our experimental evaluation were promising although it must be noted that while our heterogeneous experiment involved 53 users, the follow-up experiment in a homogeneous environment was limited to only 17 volunteers. The scalability under such a scenario remains an open issue. To increase the confidence in our results, greater sample size should be involved. We intended to address this issue in our future work by enrolling more participants. The determination of the sample size for evaluating biometric systems is still an active area of research [12], [14]. Rules such as the "rule of 3" and "rule of 30" can help in determining lower bounds to the number of samples needed for a given level of accuracy. However, these rules are over-optimistic because of the underlying assumption that biometric errors are due to a single source of variability, which is not always the case. According to Wayman, in practice, "time and financial budgets, not desired confidence intervals, always control the amount of data that is collected for the test" [14]. Thus, it is recommended to

involve as many participants as possible, and then collect sufficient samples per volunteer so that the total number of samples exceeds the bounds set by the rule of 3 and rule of 30. Following these guidelines, we plan to enlist in the future a greater number of participants in our experiments, and collect multiple samples spread over a long period of time. The flexibility that the technique provides and its runtime efficiency make it even more suitable for CA application. However, the obtained performance results do not meet the European standard for access control, which requires a commercial biometric system to achieve a FAR less than 0.001% and a FRR less than 1% [6]. While none of the existing traditional biometrics technologies meet these requirements, other than Iris biometrics [6], we intend in future work to investigate how we can improve our technique in order to meet such a goal. A key challenge faced by any biometric system is the threat of forgery. According to Ballard *et al.*, "behavioral biometrics do not lend themselves as easily to surreptitious capture as they require a user to consciously perform an action (i.e., speaking a specific phrase)" [9]. We plan, in future work, to investigate how the proposed scheme can be protected against forgery using synthetic biometrics techniques. According to Giot *et al.* [11], a well-designed update mechanism could improve the effectiveness of a biometric authenticator. In order to enhance the accuracy and stability of our proposed scheme, we plan in our future work to investigate and incorporate in our models an adaptive enrolment scheme. Through this scheme the user profile will be updated on a regular basis to reflect changes in user behavior and environmental conditions over time.

REFERENCES

[1] F. Bergadano, D. Gunetti, and C. Picardi, "User authentication through keystroke dynamics," *ACM Trans. Inform. Syst. Security*, vol. 5, no. 4, pp. 367–397, Nov. 2002.

[2] 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, Dec. 1993.

[3] P. Dowland, S. Furnell, and M. Papadaki, "Keystroke analysis as a method of advanced user authentication and response," in *Proc. IFIP TC11 17th Int. Conf. Inform. Security: Visions Persp.*, May 7–9, 2002, pp. 215–226.

[4] D. Gunetti and C. Picardi, "Keystroke analysis of free text," *ACM Trans. Inform. Syst. Security*, vol. 8, no. 3, pp. 312–347, Aug. 2005.

[5] F. Monrose and A. Rubin, "Authentication via keystroke dynamics," in *Proc. Fourth ACM Conf. Comput. Commun. Security*, pp. 48–56, Apr. 1997.

[6] D. Polemi. "Biometric techniques: Review and evaluation of biometric techniques for identification and authentication, including an appraisal of the areas where they are most applicable," [Online]. Available at: ftp://ftp.cordis.lu/pub/infosec/docs/biomet.doc.

[7] S. Ross, "Peirce's criterion for the elimination of suspect experimental data," *J. Eng. Technol.*, vol. 20, no. 2, pp. 38–41, Oct. 2003.

[8] M. Villani, C. Tappert, N. Giang, J. Simone, St. H. Fort, and S.-H. Cha, "Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions," in *Proc. 2006 Conf. Comput. Vis. Pattern Recognit. Workshop (CVPRW'06)*, June 2006, p. 39.

[9] L. Ballard, D. Lopresti, and F. Monrose, "Forgery quality and its implication for behavioral biometrics security," *IEEE Trans. Syst. Man Cybernet.*, Part B, vol. 37, no. 5, pp. 1107–1118, Oct. 2007.

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

[11] R. Giot, M. El-Abed, B. Hemery, and C. Rosenberger, "Unconstrained keystroke dynamics authentication with shared secret," *Comput. Security*, vol. 30, no. 6–7, pp. 427–445, June 2011.

[12] M. E. Schuckers, *Computational Methods in Biometric Authentication*, Springer, 2010.

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

[14] J. Wayman, "Technical testing and evaluation of biometric devices," in *Biometrics: Personal Identification in Networked Society*, A. K. Jain, R. Bolle, and S. Pankanti, Eds., Kluwer Academic Publishers, 1999.

Ahmed Awad received the Ph.D. degree in electrical and computer engineering from the University of Victoria, Victoria, BC, Canada, in 2008.

He is a Senior Scientist at the Electrical and Computer Engineering Department, University of Victoria. He has been a Software Design Engineer, Project Manager, and Quality Assurance/Security Consultant in a number of leading firms.

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

He has been with the faculty of the Department of Electrical and Computer Engineering of the University of Victoria since 1999. He is currently an Associate Professor and the Coordinator of the Information Security and Object Technology (ISOT) Lab (http://www.isot.ece.uvic.ca) at the University of Victoria.