Taylor & Francis
Taylor & Francis Group

IETE Journal of Research

ISSN: 0377-2063 (Print) 0974-780X (Online) Journal homepage: http://www.tandfonline.com/loi/tijr20

Biometric Cryptosystem based on Keystroke Dynamics and K-medoids

Vicente Alarcon-Aquino, Hector Augusto Garcia-Baleon, Juan Manuel Ramirez-Cortes, Pilar Gomez-Gil & Oleg Starostenko

To cite this article: Vicente Alarcon-Aquino, Hector Augusto Garcia-Baleon, Juan Manuel Ramirez-Cortes, Pilar Gomez-Gil & Oleg Starostenko (2011) Biometric Cryptosystem based on Keystroke Dynamics and K-medoids, IETE Journal of Research, 57:4, 385-394

To link to this article: https://doi.org/10.4103/0377-2063.86341

Published online: 01 Sep 2014.

Biometric Cryptosystem based on Keystroke Dynamics and K-medoids

Vicente Alarcon-Aquino, Hector Augusto Garcia-Baleon, Juan Manuel Ramirez-Cortes[1], Pilar Gomez-Gil[2] and Oleg Starostenko

Departments of Computing, Electronics and Mechatronics, Universidad de las Americas, Puebla Cholula, Puebla CP 72820, [1]Departments of Electronics and [2]Computer Science, National Institute of Astrophysics, Optics and Electronics, Tonantzintla, Puebla 72840, Mexico

ABSTRACT

An approach for a biometric cryptosystem based on keystroke dynamics and the k-medoids algorithm is proposed. The stages that comprise the approach are training enrollment and user verification. The proposed approach is able to verify the identity of individuals offline avoiding the use of a centralized database. The approach as reported in this paper may be implemented in stand-alone terminals or embedded in password-based systems to increase the security. The performance of the proposed approach is assessed using 20 samples of keystroke dynamics from 20 different users. Simulation results show a false acceptance rate of 2.89% and a false rejection rate of 3.35%. The cryptographic key released by the proposed architecture may be used in several potential applications such as user login, file encryption or even portable authentication to gain access to virtual private networks.

Keywords:
Biometrics, Cryptography, Keystroke dynamics, K-medoids, Minkowski distance.

1. INTRODUCTION

The fusion of biometrics and cryptography offers to explode the advantages of these two potentially complementary security technologies to create highly-secure and flexible architectures. Biometrics is about measuring the physical or behavioral unique characteristics to identify individuals with a high degree of trust whilst cryptography mainly guaranties trusted transactions over non-secure networks. The idea of this fusion is not new; however, the biometric cryptosystems developed so far require a centralized database to store the biometric information. This fact has a negative impact in the social acceptance of the proposed biometric cryptosystems [1,2].

The short history of biometric cryptosystems may be summarized with the following detailed survey which focuses on researches that extract or derive *biometric keys* [2,3]. Monrose *et al*. proposed one of the first systems which is based on keystroke dynamics [4,5]. They combine password and short binary string derived from the keystroke characteristics of the individual to create a hardened password. However, their system relies on an encrypted table which can be stolen either on its way to the database or directly from the database. Hao and Chan used handwritten signatures [6]. They extracted 43 signature features, which are quantized into bits to form a binary string. They reported 40-bit key entropy with a 28% false acceptance rate (FAR) and 1.2% false rejection rate (FRR). Clancy *et al*. proposed a similar work based on fingerprints [7]. The fingerprint minutiae locations are recorded as real points to form a locking set. However, the secret key can be derived using polynomial reconstruction. Goh and Ngo combined some of the works presented before to build a system based on face biometrics [8]. Eigen projections are extracted then mixed with random strings and quantized into single bits to form a binary string. Also, error-correction capabilities are considered in the cryptosystem. Hao *et al*. reported the first practical biometric cryptosystem that integrates the iris biometrics into cryptographic applications [1]. The proposed architecture in this work also avoids the use of a centralized database. They explode the error-correction techniques to improve the performance of the system. Finally, Garcia-Baleon and Alarcon-Aquino proposed a bimodal biometric cryptosystem based on electrocardiogram and speech signals [9]. This work combines most of the techniques presented before to create a system that avoids the use of a centralized database. It considers a password to authenticate the user and a biometric sample to verify the identity. However, due to the complexity of the system, it is designed to work in devices with high computational capabilities. More recently, several works based on keystroke dynamics have reported an improvement of the FAR and FRR of less than 2% using Gaussian mixture models and classifier fusion techniques [10-12]. However, these works sometimes need specialized hardware, or the proposed algorithms are too complex and time consuming that an implementation is not worth. What is more, these works are not designed to release a cryptographic key. Nonetheless, a brief discussion is presented in the results section in order to compare and contrast our results with those reported recently.

Keystroke dynamics can be defined as the timing data that describes when a key is pressed or released as a user types at the keyboard. The recorded timing data can be processed through an algorithm to determine a primary timing pattern (PTP) for future verification. The PTP may be used to verify the identity of the individual. The work reported in this paper considers three security factors, namely, a user password, a behavioral biometric sample, and a token. It works using a 3D random distribution of the biometric data that assures also the randomness of the cryptographic key released. The 3D pattern is extracted from the 3D random biometric pattern using the k-medoids algorithm tested for different types of distances that measure similarity, namely, Manhattan, Euclidean, Chebyshev and Minkowski distance. The rest of the paper is organized as follows: In Section 2, the k-medoids algorithm is described. Section 3 presents the Minkowski distance for measuring similarity. Section 4 presents the keystroke dynamics and shows how the PTP is extracted to work with the proposed architecture. In Section 5, the design of the proposed architecture is explained, whereas in Section 6 simulation results are reported. Finally, conclusions are reported in Section 7.

2. K-MEDOIDS ALGORITHM

The k-medoids algorithm is a clustering algorithm based on the k-means algorithm and the medoid-shift algorithm. Both, k-means and k-medoids, algorithms break the dataset up into k clusters [13,14]. Also, these algorithms attempt to minimize the squared error. The squared error can be defined as the distance between points labeled to be in a cluster and a point designated as the center of that cluster. The k-medoids algorithm chooses data points as centers instead of computing the centers as the k-means algorithm does. The k-medoids algorithm is a partitioning technique of clustering that clusters the dataset of n objects into k clusters known *a priori*. The k-medoids algorithm is more robust to outliers and noise compared to the k-means algorithm [14]. A medoid is defined as that object of a cluster whose average dissimilarity to the rest of the objects in that cluster is minimal. The partitioning around medoids (PAM) algorithm describes a common realization of the k-medoid clustering algorithm. The PAM algorithm is as follows:

1. Arbitrary selection of k objects as medoid points out of n data points ($n>k$)
2. Associate each data object in the given dataset to the most similar medoid to form clusters. The similarity in this step can be computed using distance measure. The distance measure is computed for Euclidean, Manhattan, Chebyshev, and Minkowski distance
3. Randomly select a non-medoid object named R' for each cluster
4. Compute the total cost S of swapping the initial medoid object to R'
5. If $S<0$, then swap the initial medoid with the new one. Otherwise, the initial medoid remains
6. Repeat Steps 2 to 5 until there is no change in the medoids.

The PAM algorithm is based on an iterative optimization process that evaluates the effect of swapping between the initial medoid object and the non-medoid object randomly selected. The principle of the PAM algorithm resides in Step 5. It can be seen that it may require trying all objects that are currently not medoids. Thus it represents an expensive computational cost, $Cost(k(n-k)^2)$, in each iteration. The PAM algorithm results in high quality clusters, as it may try every possible combination, working effectively for small datasets. However, due to its computational complexity, it is not practical for clustering large datasets [13,14].

3. MINKOWSKI DISTANCE

Formally, a similarity function aims at comparing two entities of a domain M based on their common characteristics. Similarity can be measured in several ways depending on the scale of measurement or data type. Based on the vector representation, the similarity can be calculated using the concept of distance. In this paper, we use the Minkowski distance to do so. The selection of the Minkowski distance is due to the fact that it is easy to implement in software and hardware, its computational cost is lower compared with more complex distances like Mahalanobis distance, and it fits well with the characteristics of the proposed approach considering the type of data used. In general, the distance d_{ij} between any two points, $P=(x_1,x_2,\ldots,x_n)$ and $Q=(y_1,y_2,\ldots,y_n) \in \Re^n$, in n-dimensional space may be calculated by the equation given by Minkowski as follows [15]:

$$d_{i,j}=\left(\sum_{i=1}^{n}\left|x_{ik}-x_{jk}\right|^{p}\right)^{\frac{1}{p}} \tag{1}$$

with k being the index of the coordinates and p determining the type of distance. There are three special cases of the Minkowski distance:

- p=1: this distance measure is often called city block distance, or Manhattan distance.
- p=2: the Minkowski distance is reduced to the well-known Euclidean distance.
- p=∞: the Minkowski distance is reduced to the Chebyshev distance. In the limiting case of p reaching infinity, the resultant equation is as follows:

$$d_{i,j}=\lim_{p\to\infty}\left(\sum_{i=1}^{n}\left|x_{ik}-x_{jk}\right|^{p}\right)^{\frac{1}{p}}=\max\left|x_{ik}-x_{jk}\right|_{i=1}^{n} \tag{2}$$

4. A BEHAVIORAL BIOMETRIC: KEYSTROKE DYNAMICS

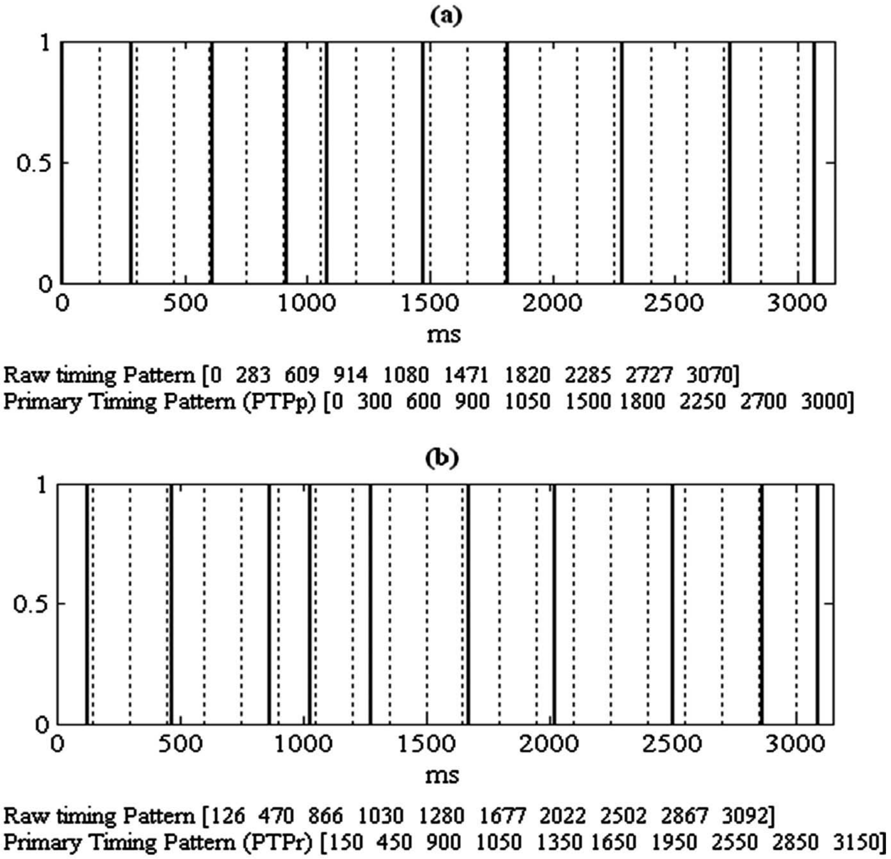
Keystroke dynamics is defined as the timing data that describes when a key is pressed or released as the user types at the keyboard. This behavioral biometric uses the manner and the rhythm in which a user types characters. The keystroke rhythms of a user are measured to develop a unique biometric pattern of the users typing for future verification. The recorded timing data can be processed through an algorithm to determine a PTP for future verification. The PTP is used to verify or even try to determine the identity of the individual who is producing those keystrokes. This is often possible because some characteristics of keystroke production are as individual as face, iris or handwritten signature [1,6,8].

The proposed technique used to extract the PTP considers partitioning the acquisition time in time slots. The size of the time slot affects directly the FAR and FRR metrics. Several experiments performed showed that a size of 150 ms for the time slot is enough to minimize the FAR and FRR metrics as it is shown in Section 6. Figure 1 shows an example of the timing data of an individual. In the top part, the timing data from the key pressing events is shown. The bottom part shows the timing data from the key releasing events. As can be seen, the key pressing process produces 10 events represented by the bold lines. The key releasing process produces 10 events also represented by the bold lines. It is important to notice that the first key pressed launches the acquisition stage and also the timer.

The rest of events are located in the time scale according to the value that the timer has when the events take place. Figure 1 also depicts that the events can occur at any time within a determined time slot; however, the time value is rounded to the closest time slot value given in *ms*. This fact assures that the extracted PTP is only comprised by a combination of the possible discrete time values otherwise the possible time values that the event could take are infinite. From now on, the extracted PTP from the key pressing events will be referred to as PTP_p and the extracted PTP from the key releasing events will be referred to as PTP_r.

5. PROPOSED APPROACH

The successful recovering of the random biometric key depends upon a correct combination of the user password, the behavioral biometric sample and the



Raw timing Pattern [0 283 609 914 1080 1471 1820 2285 2727 3070]
Primary Timing Pattern (PTPp) [0 300 600 900 1050 1500 1800 2250 2700 3000]

Raw timing Pattern [126 470 866 1030 1280 1677 2022 2502 2867 3092]
Primary Timing Pattern (PTPr) [150 450 900 1050 1350 1650 1950 2550 2850 3150]

Figure 1: Acquisition stage (a) Key pressing events pattern; (b) Key releasing events pattern.

token that stores the user password hash $H(pwd)$, the encrypted random distribution vectors, $RDV_{encrypted}$, used to reconstruct the 3D random biometric pattern, and the 3D pattern hash $H(P)$. The architecture presented here ensures that compromising two factors at most will not let the attacker reveal the random biometric key. The proposed architecture comprises two stages, namely, *training-enrollment* and *user verification*. Figure 2 shows a detailed representation of the architecture. The first stage is executed when an individual is enrolled for the first time to the biometric cryptosystem. This stage produces through a simple training process the information needed to verify the user in the second stage. The training process uses the keystroke dynamics biometric information obtained from the user at his enrollment.

The first stage can also be executed each time the random biometric key needs to be revoked or renewed for any security concern. The second stage, user verification, is executed each time the user needs to be identified before the biometric cryptosystem. The training–enrollment stage consists of the following steps:

1. A 10-character password is required to the user. The user password *pwd* is hashed using any hash function. The selection of the hash function depends on the capabilities and resources of the system where the proposed architecture may be implemented. The hash result *H(pwd)* is then directly stored in the token.
2. The raw timing data is recorded as the user types the password. Then, the PTPs are extracted from the raw timing data as explained in the previous section [see Figure 1]. As a result of the PTP extraction, two dataset are obtained, namely, PTP_p and PTP_r. A third dataset is created taking the ASCII values of the password characters. Notice that the PTPs may vary even when the biometric information comes from the same individual. This is due to the fact that the user is not allowed to input the chosen password or external factors affect his typing. Then, a training process is needed to overcome these difficulties. The purpose of the training process is to converge and generalize the PTPs. The training process is as follows:
   - The user is required to input ten times the 10-character password chosen as shown in (3). Each time the PTPs are extracted as explained previously.

$$\left\{\begin{matrix} p_1(1,\ldots,m), p_2(1,\ldots,m),\ldots, \\ p_n(1,\ldots,m) \end{matrix}\right\} \rightarrow \left\{\begin{matrix} PTP_{p_1}(1,\ldots,a), PTP_{p_1}(1,\ldots,a), \\ \ldots, PTP_{p_n}(1,\ldots,a) \\ PTP_{r_1}(1,\ldots,b), PTP_{r_2}(1,\ldots,b) \\ ,\ldots, PTP_{r_n}(1,\ldots,b) \end{matrix}\right\} \tag{3}$$

   where n=10, number of times that the password is required to the user for training purposes; m=10, number of characters of the password; a=10, number of key pressing events; b=a, number of key releasing events.
   - The respective 10 PTPs are compared each other point by point. It is clear that a PTP_p should not be compared with a PTP_r If the two compared points are separated each other for more than four time slots when the comparison takes place, that timing pattern is automatically discarded, see (4).

$$\text{Comparison}\left(PTP_{p_i}(t), PTP_{p_j}(t)\right)\Big|_{t=1}^{a} = \left\{\begin{matrix} \text{if}\min\left(PTP_{p_i}(t), PTP_{p_j}(t)\right) < \max\left(PTP_{p_i}(t), PTP_{p_j}(t)\right) \\ -4\lambda; \textit{fail} \text{ otherwise}; p_{pass}++, PTP_{p_{pass}}(\&t) \end{matrix}\right. \tag{4}$$

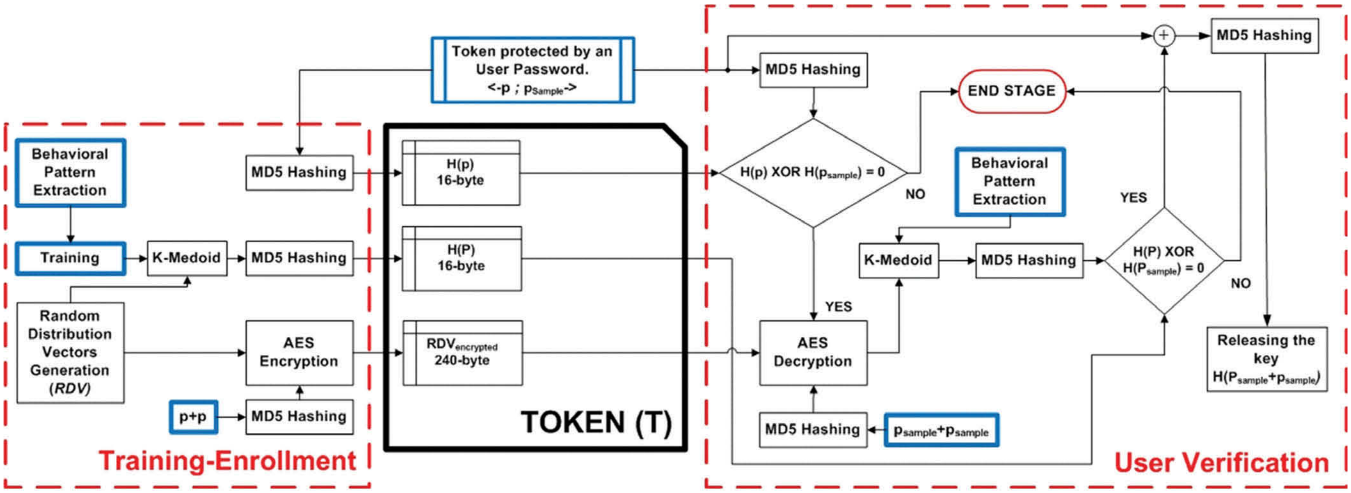


Figure 2: Proposed architecture based on the three security factors scheme.

where t refers to current test and a denotes the total number of the tests, i refers to a PTP_p selected as base for the comparisons; j refers to a PTP_p other than i that is selected to test whether it is four time slots apart or not; λ denotes the size of the time slot; ++ indicates that a counter called p_{pass}, initially equal to zero, is incremented. This counter indicates how many PTP_p have passed the test; $PTP_{p_{pass}}$ is a matrix that saves the PTP_p that has passed the test and is a concatenation operator. The training for the PTP_r is analogous to (4).

- If at least six timing patterns survive this comparison process, the mean is calculated for each point and the result is rounded to the nearest time slot value (see (5)). Otherwise, the training process must be restarted. Practical experiments showed that a user used to type a password generates the same PTPs at least 6 out of 10 tries.
- $PTP_{P_{global}}(1..a)$

$$= \frac{PTP_{P_{pass}}(1,:) + PTP_{P_{pass}}(2,:) + \cdots + PTP_{P_{pass}}(P_{pass},:)}{P_{pass}} \tag{5}$$

The global PTP_r is computed analogously to (5).

- The resultant PTPs obtained from this training process are considered as the global PTP to be used with the proposed approach.

3. Three random vectors are generated of 160 values each one (see (6)). The formed datasets by the key pressing pattern, the key releasing pattern, and the ASCII password values are distributed according to the generated Random Distribution Vector (RDV), which contain pseudorandom values drawn from the standard uniform distribution on the open interval (0, 10).

$$RDV_p(1,\ldots,s) = rand(0.10)\Big|_1^s$$

$$RDV_r(1,\ldots,s) = rand(0.10)\Big|_1^s$$

$$RDV_v(1,\ldots,s) = rand(0.10)\Big|_1^s$$

$$RDV = [RDV_p^{'}; RDV_r^{'}; RDV_v^{'}] \tag{6}$$

where s denotes the number of position the RDVs must have. In Section 6 it is shown the choice of s=160; *rand* is a function that returns a vector of s random values in the desired interval with a normal distribution; ′ is a transpose operation.

Each of the three random vectors corresponds to a coordinate in a 3D plane. Figure 3 shows a 3D random biometric distribution generated using a specific behavioral biometric with a determined random distribution vector. The 3D pattern computed is formed for the resultant eight points obtained by performing the k-medoids algorithm over the random distribution of datasets according to (7).

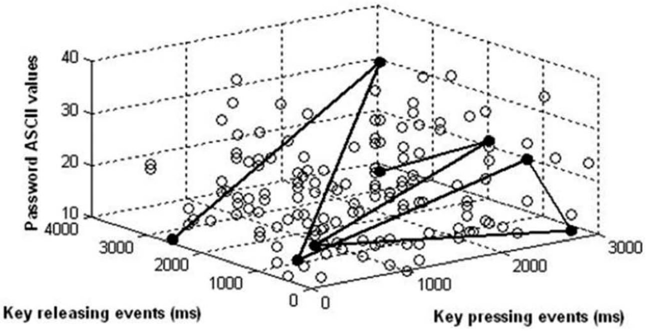


Figure 3: A random distribution of the biometric information is shown. The bold line shows the convergence points, 3D pattern, after performing the k-medoids algorithm.

$$\text{Randon Distribution}(1,\ldots,s:3)$$

$$= \begin{bmatrix} PTP_{p_{global}}(RDV_p(1,...,s)); PTP_{r_{global}} \\ (RDV_r(1,...,s)); ASCII(p(RDV_v(1,...,s))) \end{bmatrix} \tag{7}$$

where ASCII is a function that returns the ASCII value of each character of the password.

4. Once the k-medoids algorithm converges and the 3D pattern, P, is extracted, the pattern, P, is hashed using any hash function and the hash result $H(P)$ is also saved into the token.

$$P = \text{k-medoids(Random Distribution)} \tag{8}$$

5. The RDVs used to construct the 3D random biometric pattern are encrypted using a suitable encryption algorithm (e.g., AES) depending on the capabilities and resources of the system where the architecture may be implemented and stored in the token (see (9)). The hash result H(*pwd*+*pwd*) of the concatenation of the user password *pwd* is used as the cryptographic key that the encryption algorithm needs to work (see (10)).

$$RDV_{\text{encrypted}} = \text{encryption Algorithm} \\ (RDV, H(\text{pwd+pwd})) \tag{9}$$

The training–enrollment stage can thus be defined as follows:

$$\langle pwd, P, RDV \rangle \xrightarrow{\text{training-enrollment}} T \begin{cases} H(pwd) \\ H(P) \\ RDV_{encrypted} \end{cases} \tag{10}$$

Now, we proceed to a detailed description of the user verification stage. It must be assumed that the user has the token with the three parameters stored in it. The verification stage comprises the following steps:

1. The user password, pwd_{sample}, is required to the user who is claiming the identity. Then the password provided for the user is hashed using any hash function which must be equal to the one used in the training–enrollment stage, $H(pwd_{sample})$, and compared with the hash stored in the token $H(pwd)$. If both hashes do not match, the stage ends. Otherwise, the stage continues to Step 2.
2. To perform decryption over the encrypted RDVs stored in the token using as a key the hash result of the concatenation of the user password already authenticated.

$$RDV\text{=encryption Algorithm}^{-1}(RDV_{\text{encrypted}}, H\,(pwd_{\text{sample}} + pwd_{\text{ sample}})) \quad (11)$$

3. To extract the PTPs of the keystroke dynamics sample presented by the user as described previously.

$$\left\{pwd_{sample_1}(1,\ldots,m),\ldots,pwd_{sample_1}(1,\ldots,m)\right\} \rightarrow \left\{\begin{matrix} PTP_{p_1}(1,\ldots,a), PTP_{p_1}(1,\ldots,a),\ldots,PTP_{p_n}(1,\ldots,a) \\ PTP_{r_1}(1,\ldots,b), PTP_{r_2}(1,\ldots,b),\ldots,PTP_{r_n}(1,\ldots,b) \end{matrix}\right\} \quad (12)$$

4. To build the 3D random biometric pattern using as datasets the biometric information obtained in Step 3 and password in Step 1 and as distribution the decrypted RDVs obtained in Step 2.

$$\text{Randon Distribution}_{\text{sample}}(1,\ldots,s:3) = \begin{bmatrix} PTP_p(RDV(1,...,s:1)); \\ PTP_r(RDV(1,...,s:2)); \\ ASCII(p_{sample}(RDV(1,...,s:3))) \end{bmatrix} \quad (13)$$

5. To apply the k-medoids algorithm over the 3D random biometric pattern built in the previous step to extract the 3D pattern.

$$P_{sample} = \text{kmedoids(Random Distribution}_{\text{sample}}) \quad (14)$$

6. The 3D pattern recovered, P_{sample}, in the previous step is hashed using the selected hash function, $H(P_{sample})$ and compared to the hash stored in the token $H(P)$. If both hashes do not match, the stage ends. Otherwise, the stage continues to Step 7.
7. The 3D pattern is added to the ASCII values of the password of the user. The result is hashed again to obtain a random biometric key, k. This is the cryptographic key that is released and belongs to the verified user.

$$k_{\text{temp}} = P_{\text{sample}}(0,0,1..8)+ASCII\,(pwd_{\text{sample}}(1,...,m))\ k = H(k_{\text{temp}}) \quad (15)$$

In Section 6, it is explained why the 3D pattern recovered, P_{sample}, does not have the same size of the password and its length need to be adjusted as (15) shows.

The user verification stage can thus be defined as follows:

$$\left\langle pwd_{\text{sample}}, PTP_{\text{sample}}, T\right\rangle \xrightarrow{\text{user verification}} k \quad (16)$$

Notice that in both the stages (training–enrollment and user verification) it is crucial that PTPs, 3D random biometric pattern, 3D pattern, and not encrypted RDVs, used along the stages must be securely crashed and not retained in memory. Also, it is important to remark that the architecture gives the flexibility of selecting any hash function and encryption algorithm depending on the type of system where the architecture may be deployed. Nevertheless, the selection of the hash function determines the encryption algorithm to be used due to the fact that the hash result acts as the key of the encryption algorithm. We suggest then to use a hash function of 128-bits at least and the appropriate encryption algorithm if the system has limited resources. However, if more robust and secure architecture is required a 1024-bit hash function and encryption algorithm is suggested. Notice that both hash function and encryption algorithm determine the final length of the key released k by the proposed architecture.

6. SIMULATION RESULTS

In this section, the performance results of the architecture presented in the previous sections are reported. To illustrate the performance of the three security factors architecture, a Keystroke Dynamics Database was created. This database contains the timing data of 20 different users. It was collected 20 raw timing patterns total per user without any discretization process. Then, the database contains a global total of 400 timing data to be used to compute the FAR and FRR metrics and the computational cost. Although the k-medoids algorithm presents several advantages as resistance to noise and outliers compared with other clustering algorithm, it also represents a high computational cost, $Cost(k(n - k)^2)$, due to the fact that it may try every point in the dataset before converging. Table 1 summarizes the maximum, minimum and the mean number of iterations needed to converge using different types of distances. As can be seen, a minimum of two iterations are needed to make converge the 3D pattern for all distances. However, the Manhattan distance is the most effective distance because it needs at most

Table 1: Iteration comparison of the k-medoids algorithm working with different types of distance

Distance *p*	Maximum	Minimum	Mean
1	4	2	2.39
2	6	2	2.44
3	5	2	2.41
4	5	2	2.43
∞	6	2	2.49

four iterations to converge compared to the six iterations that the Euclidean and the Chebyshev distance may need or with the five iterations that the Minkowski distance evaluated in 3 and 4 may need. Also, the computed mean using the Manhattan distance is the closest value to the minimum number of iterations which assures that the frequency of convergence with two iterations is higher compared to the rest of the distances. Then, Manhattan distance is the best choice for the architecture proposed because it needs less iterations to converge and its computational cost, with *k*=8 and *n*=160, is considerably lower compared to the rest of the tested distances.

The size of the time slot must be carefully chosen to minimize both FAR and FRR. The reason of choosing the 150 ms time slot size is due to the fact that this time slot minimizes the error metrics when several raw timing patterns are compared. It must be clear that there are some raw timing patterns associated to an individual which do not fulfill with the training criterion and affects directly the metrics. Later a new computation of the metrics considering the training process shows an improvement of the metrics. Table 2 shows that an increment in the size of the time slot affects positively the FRR but affects negatively the FAR. An improvement in the FAR is produced when the size of the time slot remains small; however, under these conditions the FRR is affected in a negative way. As can be seen, there is compromise between these two metrics.

Table 2: Performance of FAR and FRR metrics for different time slots

Time slot *(ms)*	FAR (%)	FRR (%)
25	2.63	30
50	4.74	15
100	5.26	10
200	10.52	5
300	17.10	5

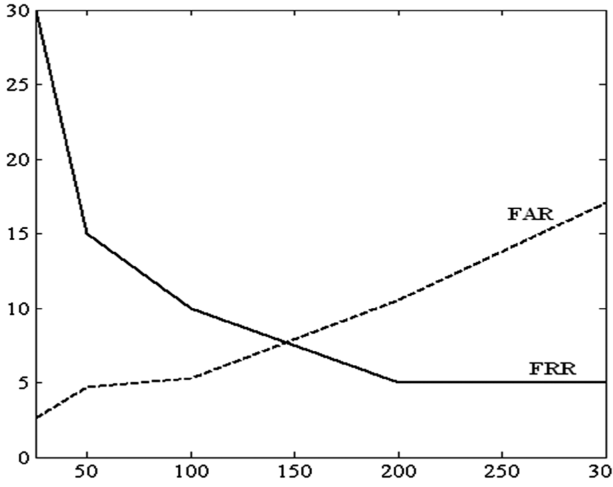


Figure 4: DET curve with no training process; x-axis, time in ms; y-axis, percentage of error in %.

According to [16], the best way to choose the value of λ which in our case is the size of the time slot is using a detection error trade-off (DET) curve. DET curves typically plot decision error rates (false reject rate vs. false accept rate) against λ values. DET curve is preferred over the Receiver Operating Characteristic (ROC) curve when it is needed to determine the best suitable value of λ known as the decision threshold. Figure 4 shows the DET curve build from the data shown in Table 2. The DET curve shows that the best compromise between FAR and FRR takes place around 150 ms.

It is also considered necessary to explain why the k-medoids algorithm uses a universe of 160 points in 3D and converge only to 8 points. First, it is fair to say that the crypto purists are familiarized to think that a 54-bit string should not be used as key. This number is a very conservative theoretical bound; however, no design should be below this bound to be considered as secure. Then if we think in a hardware implementation of the architecture and assume that the 3D pattern really acts as *salting* factor, an attacker only could be interested in knowing only the salt value. If each value of the 3D pattern *P* is represented using a byte, then 8 convergence points of the 3D pattern, 64-bit string, are good enough to accomplish the minimum of security required. Figure 5 shows the computational cost of performing the k-medoids algorithm considering different sizes of the universe with different number of groups. Given that we designed the architecture with the minimum requirements, the number of groups or convergence points chosen is 8. Figure 5 shows that the computational cost increases due to increments in the number of groups and increments in the number of total points or universe. Experimental results show that the best quality groups

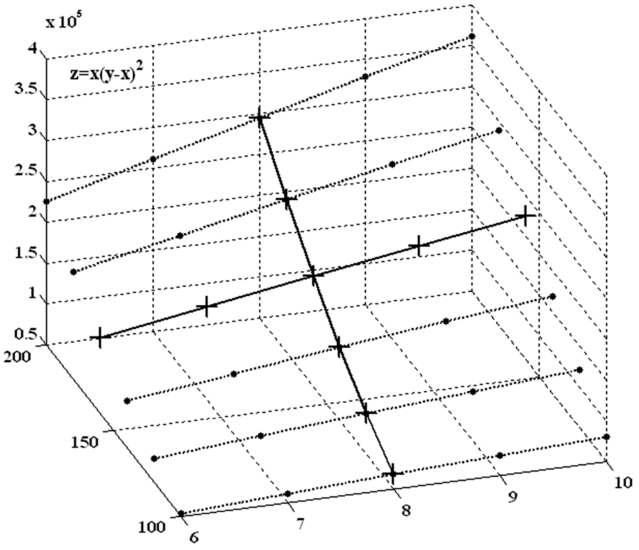


Figure 5: Computational cost of k-medoids algorithm; *x*-axis, *k* groups; *y*-axis, *n* total points or universe; *z*-axis, computational cost at the current *n* and *k*.

considering only 8 groups and the Manhattan distance were given when the size of the universe was 160. This selection assures a low computational cost and best quality grouping. Maintaining the complexity low is important because the architecture as presented here can be implemented in both software and hardware. The devices in which this architecture could be implemented do not need to have high computational capabilities. Also, it is important to mention that Figure 5 shows the maximum number of needed operations to make a 3D pattern converge. However, Table 1 shows that the number of needed iterations to make the 3D pattern converge is quite low.

Once the architecture is completely designed and the training enrollment process has been designed, the calculation of FAR and FRR is performed again. In training–enrollment stage, it is selected randomly a user as a base case, then the 20 timing data samples of that user are used in the training process to extract the PTPs. Once the PTPs have been extracted as explained previously, the proposed architecture generates and stores in the token the information needed in the user verification stage. The FAR and FRR metrics were obtained testing extracted PTPs against the 400 timing patterns stored in the Keystroke Dynamics database. Given that the database contains 20 timing data per user, it may be expected that only the 20 timing data samples that corresponds to the user who is claiming the identity should be accepted as legitimate in that iteration. The rest, 380 timing data of other users, should be rejected by the proposed architecture. However, the FAR computed after testing iteratively the architecture is 2.89%, which shows that this percentage of timing data that do not belong to the user who claims the identity respectively before the proposed architecture were accepted as authentic when they were not. Also, the FRR obtained is 3.35%. This percentage of timing data that in fact belong to the respective user who claims the identity were rejected even when they represented accurately a timing data sample used to generate the user verification data stored in the token. Most of the timing data that affected the metrics is related to PTPs that do not fulfill in the training criterion. It is important to mention that once the user is accustomed to input his password, the probabilities of generating a PTP that does not fit in the training criterion diminish considerably and thus the FAR and FRR metrics will be also less affected. Also, the FAR and FRR reported above shown that our proposed training proposed diminish considerably the metrics reported in Figure 4 where no training process was applied.

Table 3 shows a summary of some biometric cryptosystem's implementations. References [10-12] are not included despite of the good metrics because the reported works are not biometric cryptosystems. Even though, the works cited above [10-12] and others have reported EER of less than 1.5%, it should be clearly noted that it is quite hard to make a meaningful comparison between our proposed architecture and those works that may not be considered as biometric cryptosystem's implementations because conceptually these works and our proposed architecture are different. However, references [10-12] are a good reference about how our metrics must be improved. According to the data shown in the table, the FRR of our architecture performs well compared with [6,7,9]. In contrast, the FRR compared with [1] is still poor. However, it is fair to recall that our implementation is not using any error-correction technique because it is desired to keep the computational cost low. The FAR metric is poor compared with any implementation. However, once again it must be considered that the architecture was designed to perform at the best possible error metrics but with the less resource consumption.

Regarding the key length, the flexibility that provides our architecture lets to select the length of the key released whilst the other reported works have a defined length for the key released. The advantage of our proposed architecture is reflected in the capabilities and resources of the systems where it may be deployed. If the system has low resources and computational capabilities, a 128-bit combination of hash function–encryption algorithm is preferred. However, if the system is quite powerful and a high security is required a 1024-bit configuration is preferred.

The best form of performing a security analysis of the proposed architecture is assuming that the attacker has full access to the information contained in the token including the token, perfect knowledge about how the information stored in the token is generated, unlimited resources not beyond the laws of physics, and a few corresponding plaintext and ciphertext pairs where the biometric key k may be used. It is quite hard to think

Table 3: Summary of biometric cryptosystems implementations

Biometrics	Author	Features	Error handling	Key length	FRR (%)	FAR (%)
Signature	Hao (2002) [6]	43 dynamics	Feature coding	40	28	1.2
Fingerprint	Clancy (2003) [7]	Minutiae points	Reed-Solomon code	69	30	-
Iris	Hao (2005) [1]	Iris code	Concatenated coding	140	0.47	0
ECG and speech	García (2009) [9]	Wavelet transform	Hadamard code	240	10.62	0.127
Proposed approach		Time slots, k-medoids	Time slot size	128–1024	3.35	2.89

that an attacker has all the resources and information mentioned before; however, an analysis of the possible scenarios follows.

(1) The attacker has access to a few corresponding plaintext and ciphertext pairs where the biometric key, *k*, is used. In this scenario, the attacker could ignore the whole biometric cryptosystem and focus on deriving the biometric key only using the corresponding plaintext and ciphertext pairs. The complexity of deriving *k* depends upon the symmetric encryption algorithm used by the application and the cryptanalysis technique used by the attacker. For example, the most recent and first key recovery attack against AES-256 that works for all the keys has a complexity of 2^{119}. A similar cryptanalysis has been reported against AES-192. However, even though these cryptanalysis techniques claims to be able to break AES, the authors state that their attacks are still mainly of theoretical interest and do not present a threat to practical applications using AES [17]. We believe that it is easier for an attacker to compromise the user password using social engineering, to steal the token or even to get the biometrics signal samples than breaking an encryption algorithm using the most sophisticated cryptanalysis technique.

(2) Given that the user password hash $H(pwd)$ is stored in the token, the attacker may use the birthday attack to find a collision as follows: Given a function H, it can be found two different inputs $pwd1$, $pwd2$ such that $H(pwd_1) = H(pwd_2)$. Such a pair $pwd1; pwd2$ is called a collision. However, finding a collision is useless to the attacker because $H(pwd_1+pwd_1) \neq H(pwd_2+pwd_2)$. Then, it is required that the attacker knows exactly the user password pwd to be able to compute $H(pwd+pwd)$. If the attacker is able to get pwd, he is able to decrypt the $RDV_{encrypted}$. However, the attacker still needs to compromise the biometric keystroke dynamics of the individual to be verified. The information provided for the unencrypted RDV is useless due to the fact that this vector is generated randomly and does not save any information regarding the biometric behavioral sample.

There are other possible scenarios not considered here; however, the scenarios presented above can be used to determine the computational cost needed to derive the biometric random key under other scenarios. Notice that the attacker could compromise the three factors in which the architecture bases its security. However, no system can resist an attack when all the factors in which the system security is founded have been compromised.

7. CONCLUSIONS

In this paper, we have proposed an approach based on the keystroke dynamics and the k-medoids algorithm. The proposed approach comprises three security factors, namely, user password, behavioral biometric sample, and token. It assures that if an attacker compromises at most two factors, he is not going to be able to derive the random biometric key. The idea behind the three security factor architecture reported in this paper is not limited to work with the PTP as it is extracted here. The extraction technique may be more sophisticated to improve the FAR and FRR whilst the rest of the architecture remains unchanged. Instead of only considering the key pressing pattern and key releasing pattern, it could be added other parameters as the total typing time or the tendency of using certain keys by the user to make even more personal the biometric data. Furthermore, one of the most notable advantages of the proposed approach is that it is not necessary to maintain a centralized database with the biometric information. This fact impacts positively in the social acceptance of the biometric cryptosystem. The proposed three security factor approach is a very secure system because the distribution of the 3D random biometric pattern is randomly generated. Also, if an attacker could compromise the all three factors, the cryptographic key can be easily revoked and renewed by executing the training–enrollment stage again with a new password. If the user inputs a new password, a new timing pattern is then generated. In the case of that the attacker could somehow derive the cryptographic key, he could compromise the key of that specific user but not the keys of a group or a corporation that could happen in the case of maintaining a centralized database with the biometric information of all users.

The architecture reported in this paper has several potential applications such as user login, file encryption or even portable authentication to gain access to virtual private networks. This is possible due to the low complexity of the architecture. The architecture without considering the token can also be easily embedded in the current password-based systems. The flexibility that provides the block design of the architecture gives the chance of changing the hash function if there are complains about its use or changing the encryption algorithm block if there are security issues about its use. This flexibility also reflects how the architecture may be deployed in any system independently of the available resources. The complexity of the proposed identity verification algorithms is minimized without considering the hash function–encryption algorithm configuration. Then, the designer may choose a light configuration or a robust but highly secure configuration selecting 1024-bit hash function–encryption algorithm depending on the resources available.

REFERENCES

1. F Hao, R Anderson, and J Daugman, "Combining Cryptography with Biometrics Effectively", Technical Report UCAM-CL-TR-640, Computer Laboratory, University of Cambridge, 2005.

2 J Daugman, "Biometric decision landscapes", Technical Report UCAMCL- TR-482, Computer Laboratory, University of Cambridge,

2000.

3. C Soutar, D Roberge, A Stoianov, R Gilroy, and B V Vijaya Kumar, “Biometric Encryption,” ICSA Guide to Cryptography, New York, USA: McGraw Hill, 1999.

4. F Monrose, M K Reiter, and Wetzel, “Password hardening based on keystroke dynamics”, in Proceedings of 6th ACM Conference on Computer and Communications, CCCS, 1999.

5. F Monrose, M K Reiter, Q li, and R Wetzel, “Cryptographic Key generation from voice,” in Proceedings of the 2001 IEEE Symposium oc Security and Privacy, May 2001.

6. F Hao, and C W Chan, “Private key generation from on-line handwritten signatures”, Information Management and Computer Society, Issue. 10, No. 2, pp., 159-164, 2002.

7. T C Clancy, N Kiyavash, and D J Lin, “Secure smart card-based fingerprint authentication,” Proceedings of the 2003 ACM SIGMM Workshop of Biometrics Methods and Application, 2003.

8. A Goh, and D C Ngo, “Computation of cryptographic keys from face biometrics”, International Federation for Information Processing 2003, Torino, Italy: Springer-Verlag, LNCS 2828, 2003.

9. H A Garcia-Baleon, V Alarcon-Aquino, and J F Ramirez-Cruz, “Bimodal Biometric System for Cryptographic Key Generation Using Wavelet Transforms”, in Proceedings of the IEEE Mexican International Conference on Computer Science, ENC 2009, Sep. 2009.

10. L Hai-Rong, and W Wen-Yuan, “Biologic verification based on pressure sensor keyboards and classifier fusion techniques”, IEEE Transactions on Consumer Electronics, vol. 52, no. 3, pp. 1057-63, 2006.

11. P S Teh, A B Jin-Teoh, C Tee, and T Song-Ong, “A multiple layer fusion approach on keystroke dynamics”, Pattern Analysis and Applications, Theoretical Advances, London: Springer, 2009.

12. D Hosseinzadeh, S Krishnan, and A Khademi, “Keystroke Identification based on gaussian mixture models”, ICASSP 2006 Proceedings. 2006 IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 3, pp. III – III, 2006.

13. Q Zhang, and I Couloigner, “A new and efficient k-medoid algorithm for spatial clustering”, Singapore: Springer Berlin, LNCS 3482/2005, ICCSA 2005, 2005.

14. N M Barioni, L H Razente, J M Traina, and C Jr Traina, “Accelerating k-medoid-based algorithms through metric access methods”, Journal of Systems and Software, vol. 81, Issue. 3, 343-55, 2008.

15. T Kardi, “Minkowski Distance of order λ, (2009)”, Available from: http://people.revoledu.com/kardi/tutorial/Similarity/Minkowski Distance.html [Last cited on 2010 March 26].

16. A K Jain, and P Flynn, “Handbook of Biometrics”, New York, NY USA: Springer, 2008.

17. A Biryukov, and D Khovratovich, “Related-key Cryptanalysis of the full AES-192 and AES-256”, Cryptology ePrint Archive, Report 2009/317, 2009, Available from: http://eprint.iacr.org/ [Last accessed on 2009 July 20].

AUTHORS

Vicente Alarcon-Aquino received the Ph.D. and D.I.C. degrees in performance monitoring of communication networks from Imperial College London, London, U.K., in 2003. He is currently full-time professor and graduate coordinator in the department of computing, electronics and mechatronics at the Universidad de las Americas Puebla, Mexico. He has authored over 100 research articles in several refereed journals and conference proceedings, has written a research monograph on MPLS networks, and has over 100 citations to his research articles. His current research interests include security in communication networks, wavelet theory applied to performance monitoring of communication networks, intrusion detection, wavelet-based image processing, and path restoration in MPLS networks. Dr. Alarcon-Aquino has been active in program committees for international conferences and has also served as manuscript reviewer for numerous refereed journals. He is a member of IEEE and belongs to the Mexican National System of Researchers (Level I).

Email: vicente.alarcon@udlap.mx

Hector Augusto Garcia-Baleon received the B.Sc. degree in Electronics and Computer Engineering from Universidad de las Americas Puebla in 2008, and the M.Sc. degree in Electronics from the same university in 2009. In 2008, he was Software Developer in the Core Network Division at the Centre of Excellence NORTEL, Mexico City, Mexico. From 2008 to 2009, he was Research Assistant in the Department of Computing, Electronics and Mechatronics at Universidad de las Americas Puebla, Puebla, Mexico. Currently, he is a PhD student at Imperial College London in London, UK. He is working with the Intelligent Systems and Networks (ISN) Group within the Department of Electrical and Electronic Engineering. His research interests include biometrics, cryptography, security in communication networks, signal processing and wavelet theory applied to anomaly detection and data modelling.

Email: h.garcia-baleon10@imperial.ac.uk

Juan Manuel Ramirez-Cortes was born in Puebla, Mexico. He received the B.Sc. degree from the National Polytechnic Institute, Mexico, the M.Sc. degree from the National Institute of Astrophysics, Optics, and Electronics (INAOE), Mexico, and the Ph.D. degree from Texas Tech University, all in electrical engineering. He is currently a Titular Researcher at the Electronics Department, INAOE, in Mexico. He is member of the mexican national research system (SNI), level 1. His research interests include signal and image processing, biometry, neural networks, fuzzy logic, and digital systems.

Email: jmram@inaoep.mx

Pilar Gomez-Gil was born in Puebla, Mexico. She received the B.Sc. degree from the Universidad de las Americas A.C, Mexico, the M.Sc. and Ph.D. degrees from Texas Tech University, USA, all in computer science. She is currently a Titular Researcher in computer science at INAOE, Mexico. She is member of the mexican national research system (SNI), level 1. Her research interests include neural networks, image processing, pattern recognition, and software engineering.

Email: pgomezexttu@gmail.com

Oleg Starostenko received his BSc and MC degrees in Computer science from Lviv State University of Ukraine in 1982 and Ph.D. degree in Mathematics and Physics from the University Autónoma in Mexico in 1996. He is currently full-time professor in the Department of Computing, Electronics and Mechatronics at the Universidad de las Americas Puebla, Mexico. He has authored more than 150 research articles in several refereed journals, books, and conference proceedings. His current research fields are the access, retrieval, transmitting, and processing of multimedial information in distributed environments. He belongs to the Mexican National System of Researchers (Level I).

Email: oleg.starostenko@udlap.mx

DOI: 10.4103/0377-2063.86341; Paper No JR 163_10; Copyright © 2011 by the IETE