Paper: ASAT-15-194-CE

15th International Conference on
AEROSPACE SCIENCES & AVIATION TECHNOLOGY,
ASAT - 15 – May 28 - 30, 2013, Email: asat@mtc.edu.eg ,
Military Technical College, Kobry Elkobbah, Cairo, Egypt,
Tel: +(202) 24025292 –24036138, Fax: +(202) 22621908

A Secure Fingerprint Authentication on Smart Card

{I. M. El-leithy*, G. I. Mohamed, T. A. Mahmoud}†

Abstract: We propose a small size fingerprint matching algorithm (4 KB) that can be executed on devices with a low computational power and a limited memory size. The algorithm is based on matching features that are invariant with respect to major transformations like translation and rotation. The matching algorithm has been implemented on a smartcard over the Java Card ™ platform. The algorithm has an asymmetric behavior, with respect to the execution time that varies between correct positive and negative matches. The performance in terms of authentication reliability was tested on some databases from the Fingerprint Verification Competition 2002 (FVC2002) .

Keywords: Biometric authentication, fingerprint matching, match on card, smartcard.

1. Introduction

Any human physiological and/or behavioral characteristic can be used as a biometric characteristic as long as it satisfies some requirements such as permanence (measurement should be invariant with time), uniqueness (different values for different persons), universality (everyone should have this trait), acceptability (if people are willing to accept this technology), performance (the recognition accuracy and system requirements) and circumvention (how it is easy to fool the system) [2].
The most common biometric techniques are facial analysis, fingerprint verification, hand geometry and voice verification. Fingerprint matching is one of the most common biometric techniques used in automatic personal identification, because of its strong reliability and its low implementation cost. Moreover, it is also the most explored technology of all the others.
Performing a biometric verification inside a smartcard is very difficult, since the processing capabilities of smartcard processors are limited for such a complex task. With Match-on-Card (MoC) technology, the original fingerprint template is stored inside the card, unavailable to the external applications and the outside world. In addition, the final matching decision is computed inside the smart card itself. Thus, the proposed MoC algorithm was developed to work in this bounded environment .

The algorithm is based on some minutiae characteristics and more precisely on their local structure information. So there is no need to pre-align the processing fingerprint templates, which would be a difficult task to implement inside a smartcard. Moreover, it shows an asymmetric execution time between correct positive matches (same fingerprint) and correct negative matches (two different fingers). This is because the match procedure stops immediately when few minutiae pairs result in a positive match. If this check does not succeed, for example if the two fingers are different, or if the two acquisitions of the same finger are very different, the procedure is executed till the end (which will take longer time) and the match decision is taken only at its end.

* Elleithy78@hotmail.com
† Egyptian Armed Forces, Egypt.

Any biometric verification system is being tested by measuring two types of errors: considering the biometric measurements from two different fingers to be from the same one (false acceptance), and considering two biometric measurements from the same finger to be from two different fingers (false rejection).The probability of these events is respectively defined as the False Acceptance Rate (FAR) and False Rejection Rate (FRR).

2. Related Work

Regarding scientific literature on MoC, in [3] the minutiae spatial positions and the associated ridge orientation angle (x, y, θ) are used to represent and match the minutiae in the two templates. The authors use an accumulator array to compute the approximate transformation, and this process is repeated to find a fine-grain resolution, discretizing every time the array cell corresponding to the best transformation. However, the resources needed by this algorithm are above the current availability of today's smartcard, since it has been tested on a 32 bit ARM-7 processor. The test image set is composed of 400 images taken from 100 individuals and the reported equal-error rate (EER) is about 6% .The equal error rate means that the false acceptance rate FAR and false rejection rate (FRR) are identical.

The authors of [4] use the same representation (x, y and θ). The transformation between the two templates is accomplished outside the smartcard using the average horizontal and vertical coordinate values and the average direction of all the minutiae. The match is then performed by transforming the coordinates in a polar form with respect to the previous average values; for this reason, problems could arise in case of partial overlapping between the two fingerprints. Here the database has been generated using only 10 different fingerprints, and about 20 minutiae in each template. One more paper using a first step concerning external registration is [5]. Here, the match is then accomplished by applying Gabor filters to the fingerprint image, as described in [6]. The reliability tests were executed on the Siemens Fingerprint Database, containing 100 images each of 36 distinct users. Performances are reported for several different system configurations and, in the best reliability case, the procedure achieves a FRR of about 4% for a FAR of 0.1%, using four different reference templates (storage occupation is 9-10 Kbyte) and only one "query template".

The algorithm in [7] uses both the binary fingerprint image and the minutiae position. First, the host PC transmits the core location of the new acquired image, which is used by the card to compute registration parameters; these parameters are sent back and used by the host to register its image. Then the smartcard chooses some coordinates (typically, near to the minutiae locations) and the host PC use them to cut rectangular pieces (called "chips") from the fingerprint image. Again, these chips are transmitted to the card and used for the final match step. Performances are obtained on a database with 576 fingers, with a FAR of 0.1% and 2% FRR.

A verification method implemented in a Personal Digital Assistant (PDA) with a 206 Mhz Strong Arm processor is presented In [8]. So for a less resources-constrained system in respect to smartcards. The minutiae information recorded during the extraction is the triplet x, y and θ. The main aim of the authors is to show that replacing floating-point with fixed point computation does not affect the verification reliability performance: actually, most of the embedded processors in embedded devices do not support floating-point arithmetic. Only a small subset of all the minutiae points (the ones near to the core point) is used during the match procedure, which simply applies a bounding box technique. The test databases are collected from 383 different fingers for a total of 1149 images and the average computation time is 0.9 seconds. The achieved EER is nearly 7%.

The authors in [9] introduce an algorithm developed for embedded devices. This algorithm is based on the minutia neighbor features, like the neighbor distance and orientation with respect to the central minutia. The minutiae neighborhood similarity is computed by finding the feature distances and successively controlling them with the aid of a delimiting bounding box; if the checks are positive, the corresponding neighbors are then matched. The compared minutiae are considered as matched if the total number of their matching neighbors is above a certain predefined value. The final decision regarding the two entire templates is taken from an estimation based on the total number of the minutiae matched in this way. The test database has 100 images and performance are claimed to be 0.01% FAR and 1% FRR.

In [10] a description is given of a matching algorithm expressly developed for the Java Card™ platform and using the same feature extractor software as the one adopted in our solution. It uses two distinct algorithms for different feature types (a hybrid matcher) and at the end, the overall score is computed as a linear combination of the two independent sub-scores. The first algorithm is based on the minutiae features and a graph structure is built starting from the core point position, then visiting the neighbor minutiae. The matching procedure is inspired from the point-pattern matching algorithm in [11] and its purpose is to find a spanning ordered tree touching as many nodes as possible in the two graphs. The second algorithm is ridge feature-based (texture) and is implemented exactly as described in [12].

3. Framework for a Secure Fingerprint Authentication on Smart Card

3.1 Background

In Sec. 2, the previous work related to the Match-on-Card problem has been presented. Some works have however been proposed for processor and devices with higher computational capabilities [3,8,9]. However, other algorithms are analyzed on databases with few images [4]. In other papers, good security performances are achieved only repeating the match on multiple templates [5], thus decreasing the average match time. Most of times, given solutions seem to need too much processor resources to be developed inside present-day smartcards.

In this paper, the challenge was to accomplish an algorithm that can be implemented in a current smartcard environment; at the same time, the algorithm had to have satisfying characteristics of speed and security. For this reason, the local minutiae matching technique based on minutiae neighborhood is used, with which pre-align the two fingerprints is avoided, and which generally supplies simplicity and low computational complexity. With local structures it is possible to deal with displacement, rotation and partial overlapping problems, while bounding boxes are employed on the feature differences to compensate for the small plastic deformations.

3.2 Framework Description

Figure 1 depicts the functional block diagram of a secure fingerprint authentication on smart card. It shows that fingerprint authentication consists of the enrolment stage and the verification stage. Moreover, it could be noticed that the feature extraction is done in the computer and the matching process is done on the smartcard itself.

It is useful to compare the matching algorithm performance with a common reference that is why Finger Verification Competition 2002 (FVC2002) is used. FVC2002 is a public benchmark allowing industrial, academic and independent developers to compare their

algorithms. In this way, we can compare and show the performance with respect to the algorithms in the competition, which are not restricted to the smartcard constraints.

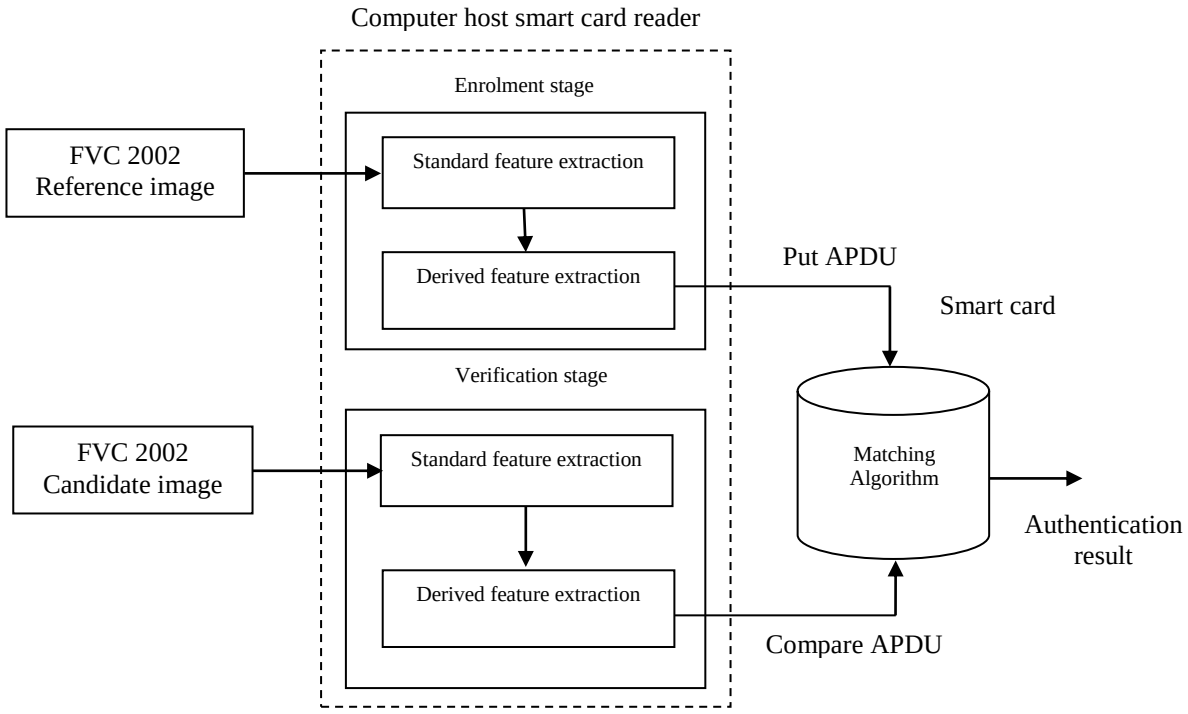

Figure (1): Functional block diagram of a secure fingerprint authentication on smart card

3.3 Features Extraction

In our implementation we have adopted the NIST Fingerprint Image Software (NFIS) [13], an open source toolkit which includes the MINDTCT minutiae data extractor used to extract the minutiae from a given fingerprint image. We have used this information to derive additional features directly used in our matching algorithm. These features are computed for each minutia with respect to its neighbors, and so each neighbor is described by four features (as shown in Figure 2):

- The Euclidean distance between the central minutia and its neighbor minutia (segment D); latterly referred to as Ed (Euclidean distance).
- The angle between segment D and the central minutia ridge direction (angle α); latterly referred to as Dra (Distance relative angle).
- The difference angle between central minutia and neighbor ridge orientation angle ($\theta_1 - \theta_2$); latterly referred to as Oda (Orientation difference angle).
- The ridge count between central and neighbor minutiae: given two minutiae A and B, the ridge count between them is the number of ridges intersected by the segment D (in Fig. 2 ridge count value is 1); latterly referred to as Rc (Ridge count).

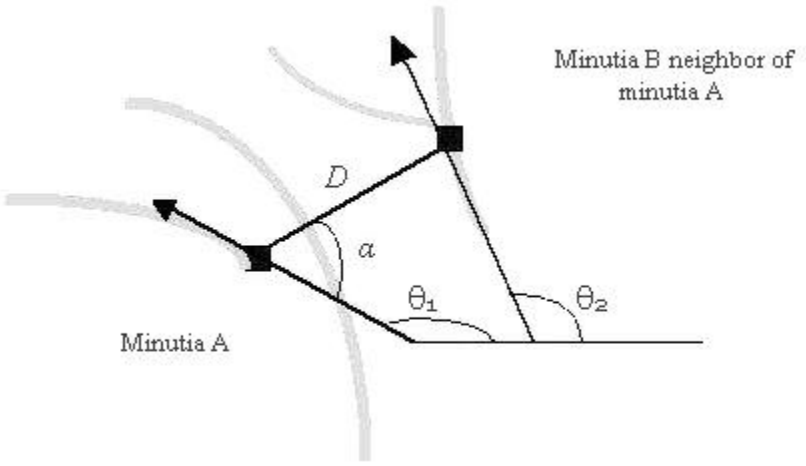

Figure (2): Graphical description of the used features

3.4 Matching Algorithm Description

The matching algorithm computes how much the neighborhood of a minutia in the candidate template is similar to the neighborhood of each minutia in the reference template. After this process, the two most similar minutiae are matched and then discarded from following scan phases concerning other different minutiae of the candidate template. All these similarity measures are summed together during the process, and at the end the algorithm can decide if the two templates match by applying a threshold on this global score.

Because the algorithm is run on the bounded environment of the smartcard , waiting for a complete minutiae match could lead to a waiting time too long for the user, that problem had been solved by stopping the algorithm as soon as one of the following cases is reached

a. The algorithm finds some minutiae pairs (4 in our case) are matched with a very good similarity measure.

b. Only the last examined minutiae pair has a matching value less than a very rigorous threshold.

c. If the two previous conditions are not fulfilled, the algorithm explores all the minutiae pairings space.

The delay for unsuccessful matches scanning all the minutiae list is not a problem, because it is more important to gain a high execution speed while verifying the true card-owner identity than quickly rejecting an impostor.

4. The Implementation of the Proposed Match-on-Card Algorithm

a. Our Implementation differs from the proposed algorithm in [14] in that we have reduced the number of neighbors of each minutia from 8 to 5 and this will lead to the following:

1) MINDTCT C source code does not have to be modified to get 8 neighbor for each minutia

2) The size of memory required to represent each minutia is reduced by about 62.5%.

3) Since an increase in the size of the templates to be compared, means a greater increase in the execution time of the algorithm, so we have earned a great enhancement in execution time.

b. Converted applet file (cap file) is generated for the matching algorithm with size 4 KB (which is suitable for loading into the java card) using eclipse-SDK-3.7.2.

c. The cap file is loaded into the java card using GPShell-1.4.4.

d. The two fingerprints are matched using the uploaded applet by sending to the java card three APDU (application protocol data unit)

1) Select APDU to select the matching applet (cap file).

2) Put APDU to upload the array of bytes of the reference fingerprint.

3) Compare APDU which contain the array of byte of the candidate fingerprint.

e. The matching result is obtained by showing the response APDU which show whether the result is true or false.

The fingerprint matching algorithm has been fully developed in JCOP V2.4.1 Revision, Product type J2A080, java card 2.2.2, global platform 2.1.1. The chosen smartcard has 77968 bytes of EEPROM, 68112 bytes free ROM for applets, about 2014 bytes of RAM memory, 1462 bytes APDU buffer, the transmission protocol used is the T=0.

The card support extended length of APDUs with a maximum total length of 32767 bytes.
Due to the environmental constraints like the EEPROM space, we have limited the maximum number of minutiae forming the template to the 20 most reliable, and the neighbor feature values have been sampled to be then stored in the low capacity Java Card™ data types as byte type.

5. Experimental Results

In order to evaluate our proposed implementation of the matching algorithm, FVC2002 DB1_A (FVC2002) fingerprint databases have been used to measure the performance.

After choosing the best values of the established thresholds to improve the results, a FAR100 (the lowest achievable FRR for a FAR <= 1%) of 17% have been finally obtained with the FVC2002 fingerprint database. This is a very good result considering that, first, these databases include a lot of low quality images, and second, it is an implementation optimized to achieve both memory efficiency and faster matching time. A comparison of the performance and memory required for fingerprint templates between (Bistarelli 2006) algorithm in [14] and our implementation is shown in Table 1.

Regarding matching time, it is found that an on card matching time of about (0.9-50) seconds is obtained for nearly all of the matches. Maximum time is performed only when the two acquisitions belong to different fingerprints. In this case, the algorithm explores all the minutiae pairs. Actually the maximum time is not a problem, because it is more important to gain a high execution speed while verifying the true card-owner identity (true acceptance) than quickly rejecting an impostor (true rejection).

Table 1: Overall performance results of both algorithms applied on FVC2002 fingerprint databases

Algorithm	FAR	FRR	Required memory
(Bistarelli) algorithm in [14]	1%	10.6%	2×20×8×4 = 1280 byte
Proposed algorithm	1%	17%	2×20×5×4 = 800 byte

6. Conclusions

In this paper a new fingerprint matching algorithm is proposed. This algorithm is tolerant to typical match problems such as rotation, translation. Our procedure achieves a small size algorithm suitable for the Java Card™ environment. The high reliability, as determined from our analysis, can be further greatly improved using a good enrollment image, that produces a good quality template. The hypothesis of having a good quality template is not too restrictive and it is easily applicable, since the enrollment phase is accomplished only one time at the expedition of the java card. Scoring a FAR100 result of about 17% makes the algorithm implementation feasible in the live-scan applications for identity verification (like a MOC system). Our procedure is stopped as soon as the two templates are considered to belong to the same finger. Moreover algorithm shows a different execution time between correct positive and negative matches.

References

[1] FVC2002, http://bias.csr.unibo.it/fvc2002/

[2] A. Jain, A. Ross and S. Prabhakar, An Introduction to Biometric Recognition, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 14, No. 1, January 2004.

[3] S. B. Pan, D. Moon, Y. Gil, D. Ahn, and Y. Chung. An ultra-low memory fingerprint matching algorithm and its implementation on a 32-bit smart card. IEEE Transactions on Consumer Electronics, Vol. 49, pages 453–459, May 2003.

[4] Y. S. Moon, H. C. Ho, K. L. Ng, S. F. Wan, and S. T. Wong. Collaborative fingerprint authentication by smart card and a trusted host. Canadian Conference on Electrical and Computer Engineering, Vol. 1, pages 108–112, Washington, DC, USA, March 2000.

[5] J. Reisman, U. Uludag, and A. Ross. Secure fingerprint matching with external registration. Audio and video based biometric person authentication, 5th International Conference, pages 720– 729, NY, USA, 2005.

[6] A. Ross, A. Jain, and J. Reisman. A hybrid fingerprint matcher. Pattern Recognition, Vol. 36, No.7, pages 1661-1673, July 2003.

[7] S. Ishida, M. Mimura, and Y. Seto. Development of personal authentication techniques using fingerprint matching embedded in smart cards. EICE Transactions on Information and Systems, Vol. E84-D, pages 812–818, July 2001.

[8] T. Y. Tang, Y. S. Moon, and K. C. Chan. Efficient implementation of fingerprint verification for mobile embedded systems using fixed-point arithmetic. Proceedings of the 2004 ACM Symposium on Applied Computing, pages 821–825, New York, USA, 2004.

[9] S. Yang and I. Verbauwhede. A secure fingerprint matching technique. Proceedings of the 2003 ACM SIGMM Workshop on Biometrics Methods and Applications, pages 89–94, New York, USA, 2003.

[10] T. Cucinotta, R. Brigo, and M. Natale. Hybrid fingerprint matching on programmable smart cards. International Conference on Trust and Privacy in Digital Business, pages 232–241, Spain, 2004.

[11] P. van Wamelen, Z. Li, and S. Iyengar. A fast algorithm for the point pattern matching problem. Technical Report 1999-28, Louisiana State University, Dept. of Mathematics, Baton Rouge, USA, 2000.

[12] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Filterbank-based fingerprint matching, IEEE Trans. Image Processing, vol. 9, no. 5, pages 846-859, May 2000.

[13] User's Guide to NIST Fingerprint Image Software (NFIS), NISTIR 6813, National Insitute of Standards and Technology.

[14] S. Bistarelli, F. Santini, and A. Vaccarelli. An asymmetric fingerprint matching algorithm for Java Card. Pattern Analysis & Applications, Springer, Vol. 9, No. 4, pages 359-376, 2006.