Hybrid Fingerprint Matching on Programmable Smart Cards *

Tommaso Cucinotta, Riccardo Brigo, Marco Di Natale

Scuola Superiore Sant'Anna
{cucinotta,rojelio,marco}@sssup.it

Abstract This paper presents a hybrid fingerprint matching algorithm combining two heterogeneous schemes, namely the texture-vector and minutiae-based methods. The proposed technique has been designed in order to run on a programmable smart card, with image processing and feature extraction performed on the host, and matching performed by the card device. The two matching algorithms have been carefully tuned in order to achieve an acceptable performance despite the computation and memory constraints. Given the high level of intrinsic security that smart cards already have, and the interactive nature of target applications, the complexity of the problem has been greatly reduced, making such an approach feasible. This is validated by the experimental results we show, gathered from an implementation onto a Java Card device, where acceptable false acceptance and rejection rates are achieved at the cost of a reasonable response time of the device.

1 Introduction

User authentication is one of the most important issues when designing a secure system. Traditional password based solutions, relying on the concept that a user is authenticated by proving knowledge of a secret information, usually offer an unacceptable security level. In fact, the secret information can easily be revealed to (or stolen by) unauthorised users. If the password is not strong, it can also be easily guessed by an attacker. Use of smart cards, along with cryptographic authentication protocols, increases security by requiring a user to prove both possession of a physical card, containing a cryptographic key, and knowledge of a secret information, usually a Personal Identification Number (PIN) protecting the card (two factor authentication). This raises the security level with respect to remote attackers, but still it is subject to the problem of voluntary delegation, or card stealing / PIN extortion. Biometrics based authentication techniques solve this problem, by requiring the user to prove possession of a unique, characteristic property of his own body, such as fingerprints ridges, hand shape, retina, etc... When such a technique is used in conjunction with smart card technology, a high security level is achieved since users are required to prove, at the same

* This work has been partially supported by the European Commission within the IST project 2001-34820 ARTIST.

time, knowledge of a secret information, possession of a physical token, and possession of their own physical body (three factor authentication), before access to a system is granted.

This work is focused on systems where the authentication mechanism relies on the cryptographic capabilities of the card, and *fingerprint verification* is used by the card, in addition or alternative to PIN code verification. An alternative target is a secure application running entirely or in part onto a smart card, where the card itself authenticates users. A typical target application is smart card based digital signature, where the non repudiation property, usually established only at a jurisdictional level by dictating card owner responsibilities, can be technically enforced by requiring a biometrics authentication by the card, before the signing operation takes place.

This paper is organised as follows. Section 2 briefly reviews works found in literature related to fingerprint verification. Section 3 features an overview of the proposed technique, with a detailed description of the matching mechanism that has been implemented on the card device. Evaluation results for the proposed algorithm are reported in Section 4. Specific notes about the algorithm implementation are reported in Section 5. Finally, Section 6 draws conclusions and presents possible areas of future investigation.

2 Related Work

In recent years, the problem of merging smart card technology and biometrics for the purpose of authenticating users has gained more and more attention from research and industry altogether. Smart card based authentication has been widely used whenever user authentication was required, though the result has always been the authentication of the plastic card itself, not the user. Biometrics promise a final solution to this problem, achieving an integrated authentication system in which not only a user is authenticated by proving possession of a physical token and knowledge of a secret information, but by showing to the system some unique biological characteristics of its own body.

Correct use of biometrics and smart cards is not as immediate as it could seem at a first glance. Recent works [1,2] focused on the possible attacks a system integrating such technologies could be subject to. In [2] eight types of attacks to a biometric authentication system are identified, targeted either to the components themselves, or to the communication protocols among them. Recently, the European Union has also focused attention on feasibility of matching-on-card technologies, as in [3], where it is underlined that, in the context of electronic signatures, the possibility of identifying people based on biometric characteristics is of fundamental importance due to the non-repudiation security requirement, and the need for on-card matching is also outlined.

Feature extraction from fingerprint images has been widely studied, as shown in [4], where a good overview is made on the general structure of automatic fingerprint identification systems (AFIS), emphasizing the main challenges such a system has to face with. In [5] the authors demonstrate how an image en-

hancement algorithm based on Gabor filters can significantly improve the performances of an AFIS thanks to the greater reliability and precision gained by the minutiae extraction process, which leads to a reduced False Rejection Rate (FRR) for a given False Acceptance Rate (FAR). In [6], Prabhakar proposes an innovative approach to fingerprint analysis & matching, based on the use of a Gabor filter bank to extract from the fingerprint image statistical information, which have been proven to degrade much more smoothly with image quality than classical minutiae-based algorithms. This approach has further been developed in [7] in order to achieve comparable performances even in conjunction with small sensors, which offer to the analysis system only a limited portion of the fingerprint. In the same work, the authors opened a relatively new investigation direction inspired from the so called *multi modal* biometric verification techniques, where multiple kinds of a person biometric characteristics are used at once for the purposes of authentication. Due to the independence between the different kind of biometric information that is matched in such techniques, a combination of them results in a higher performance, as shown in [8].

The specific problem of combining two fingerprint matching algorithms in order to improve performance is addressed in [9], where the authors compare three different ways of combining the scores obtained from distinct matching algorithms (a *linear* combination, a *multiplicative* combination and a combination based on the *logistic* function) and demonstrate that the best performance is achieved with the logistic function.

With respect to previous investigations on hybrid fingerprint matching, the approach which is being introduced in this paper is specifically focused on the problem of implementing such techniques onto programmable smart card devices. It does not aim at achieving the highest possible performance, but achieving an acceptable performance for the cited usage context, while keeping a sufficiently low complexity level so to allow implementation onto a programmable card device. We give an extensive description of the adopted algorithms, and a precise specification of how various parameters have been tuned in the implementation. Furthermore, we present an on-card architecture for the matching algorithm, realised as a consistent extension to the protocol and JavaCard Applet introduced in [10], and report experimental timings gathered from the execution of the proposed algorithm onto a JavaCard device.

On a related note, fingerprint matching on JavaCard devices is not novel, as industrial products already exist based on the same kind of technology, like the one from Precise Biometrics [11]. However, implementation details and extensive description of the experimental setup from which such measurements arise are not available, making it impossible to perform a comparison with other approaches.

3 Hybrid Matching

Our system uses both Prabhakar's *fingercode* and minutiae information in order to perform a multi modal biometric verification of the user. Both techniques

have been split into the two fundamental steps of feature extraction and feature matching. Thus, the live-scanned fingerprint image is first analysed on the host machine in order to extract the features using the two relatively complex feature extraction algorithms; such features are then transmitted to the smart card device, which performs the matching phases of both algorithms, comparing the received features with the templates previously stored into its internal memory during enrollment.

In the following, we report a description of the extraction and matching algorithms adopted for the two techniques.

3.1 Features Extraction and Representation

Fingercode. Fingercode extraction has been implemented following the method described in [6], where an exhaustive description of the algorithm can be found. Briefly, it consists of the following steps (see Fig. 1). First, the fingerprint image is normalised to a constant mean and variance, then a reference point (*core*) is determined, defined as the topmost point on the innermost upward ridge. A circular *region of interest* around the core is then tessellated and filtered using eight Gabor filters, tuned over eight different directions. For each of the filtered images, and for each tessel, the intensity absolute deviation from the mean is computed. The complete list of such absolute deviations, normalised in the range [0..255], constitutes the *fingercode* of the original image.

Figure 1. Example of Fingercode computation: a *region of interest* is determined around the core, then it is directionally filtered (only vertical filtering is showed), tessellated and intensity absolute deviations (represented in gray scale) are computed.

Minutiae. In order to extract the minutiae from the fingerprint image, we adopted the algorithm supplied by the National Institute of Standards and Technology (NIST) as implemented in the NIST Fingerprint Image Software (NFIS), a public domain software developed for the Federal Bureau of Investigation.

This algorithm can be roughly subdivided into the following phases [1]. First, the gray-scale fingerprint image is reduced to a binary, black and white one, then analysed in order to find every singular point (bifurcations and terminations) which could be a potential minutia. This operation results in false positives (minutiae detected where none exists), due to low-quality image, cuts, bruises or other *noise*. Thus, various heuristics are applied to discover and delete such false positives (for example two facing and aligned minutiae at small distance are likely to be due to a cut determining two false terminations). Finally, for each of the remaining minutiae the algorithm outputs the position, direction (defined as the main direction of the surrounding ridge flow) and an index of reliability, determined considering multiple factors such as local image quality and proximity to image borders. Our system excludes from further analysis the minutiae with a reliability index under a given threshold. The others are ordered based on increasing distance from the core, where only the nearest num_{m} ones, up to a maximum number of *maxMinutiaeNumber*, are considered, so to limit computation requirements for the matching phase.

Let $\{m_i\}_{0 \leq i \leq num_{\mathrm{m}}}$ denote the set of found minutiae, where m_0 is the fingerprint core. The algorithm builds a graph representation of the minutiae, where each minutia m_i is associated a node n_i in the graph, and the set of outgoing arcs from a node n_i represents the set of minutiae which are considered *neighbours of* m_i for the purpose of matching. The following algorithm builds the graph:

1. determine the bounding-box of the minutiae set;
2. let ref_i be the number of references to m_i, initially 0;
3. let p be the list of pending minutiae; initially p contains m_0;
4. let c be the list, initially empty, of *consolidated* minutiae;
5. extract next minutia m_{curr} from p, and add m_{curr} to c;
6. enumerate m_{curr}'s nearest neighbours, given the following restrictions:
   (a) a maximum of max_{neigh} neighbours can be listed, where max_{neigh} is n_{c} if $m_{\mathrm{curr}} \equiv m_0$, n_{m} otherwise;
   (b) minutiae in c are ignored;
   (c) neighbour's distance from m_{curr} must be in the range $[dist_{\mathrm{min}}, dist_{\mathrm{max}}]$; we do not accept a neighbour too close because at small distances even light errors in position detection can determine large variations in direction when expressed in polar coordinates; on the other hand we can't accept too far neighbours because at large distances the elastic deformation of finger's skin couldn't be ignored;
7. for each neighbour m_j found
   (a) associate to m_{curr} the corresponding *vector* (i. e. distance and direction from m_{curr} to m_j and the index j);
   (b) increment ref_j; if $ref_j = max_{\mathrm{ref}}$, add m_j to c;
   (c) add m_j to p, if not already present;
8. if p is not empty, continue from step 5.

[1] For further details the reader is referred to NFIS official documentation, freely distributed by NIST (*http://www.nist.gov*).

If $max_{\text{ref}} = 1$, the graph becomes a *spanning tree* touching every minutia in the set; the choice to allow multiple references ($max_{\text{ref}} > 1$) to the same minutia is due to the necessity to give the graph enough redundancy, which (as discussed in Section 3.2) reduces the probability of erroneous early abort by the matching algorithm when comparing two corresponding fingerprints. On the other hand, max_{ref} has to be lower than max_{neigh}, otherwise the graph could result in a strongly connected, central cluster of nodes which does not reach outer minutiae[2].

The representation of a fingerprint, as output by the described process, is composed of: the bounding box coordinates; the found minutiae list $\{m_i\}$, including, for each minutia, its Cartesian coordinates (relative to the core), direction and list of vector-distances to its neighbours.

3.2 Matching

Fingercode. Fingercode matching has been implemented as described in [6]: given the two vectors, we compute the sum of absolute differences between corresponding elements and store the result as the score of the process $score_{\text{fc}}$.

Minutiae. The minutiae matching has been inspired by the point-pattern matching algorithm described in [12], with the simplification obtained by computing a common reference point: the core. In the following, we consider two vectors (as defined in Section 3.1, step 7a) $v_1(dist_1, dir_1, i_1)$ and $v_2(dist_2, dir_2, i_2)$ to match given the rotation rot and the tolerance parameters th_{dist}, th_{dir} and th_{angle} when:

$$\begin{aligned} |dist_1 - dist_2| &< th_{\text{dist}} \\ |dir_1 - dir_2 + rot|_{360} &< th_{\text{dir}} \\ |m_{i_1}.dir - m_{i_2}.dir + rot|_{180} &< th_{\text{angle}} \end{aligned}$$

where $m_i.dir$ denotes the direction of the i^{th} minutia. These three tolerances have been chosen by performing a statistical analysis of pairs of corresponding fingerprints.

The basic task of the algorithm is to find, given the template and the minutiae graphs, a *spanning ordered tree* touching as many nodes as possible, starting from the two cores (which are assumed to be corresponding by hypothesis) and visiting the graphs only via common arches, i. e. the ones corresponding to vectors matching within accepted tolerance.

The algorithm proceeds as follows:

1. Let T_i and S_i indicate respectively the i^{th} minutia of the template and of the proposed set;

[2] It should be noted that we *do not* guarantee to reach every minutia, but only that the probability of a minutia to be excluded from the graph is sufficiently low.

2. let m be the list of matches found, composed of couples of indexes, where the presence into m of the couple (i_1, i_2) means that a match has been detected between T_{i_1} and S_{i_2};
3. let p be a list, initially empty, of *pending* minutiae;
4. look for the rotation *bestRot* which gives the maximum number of matches among the two cores' neighbours under tolerances th_{distCore}, th_{dirCore} and th_{angle} (th_{distCore} and th_{dirCore} are less restrictive than their general counterpart to take in account the possible imprecision in core detection); the search has two limitations:
   (a) $|bestRot| < max_{\text{rot}}$ (we assume that the user puts his finger approximately vertically);
   (b) given two rotations rot_1 and rot_2 which give the same number of matches, the lower one (in absolute value) is preferred;
5. for each minutia S_i for which a corresponding T_j was found during previous step, insert (j, i) into m and S_i into p;
6. extract next pending minutia S_{curr} from p and find into m the corresponding matching template minutia T_{corr};
7. for each vector $v_1(dist_1, dir_1, i_1)$ associated to S_{curr} look for a matching vector $v_2(dist_2, dir_2, i_2)$ associated to T_{corr} with rotation *bestRot* and tolerances th_{dist}, th_{dir} and th_{angle}; for each match found for which (i_2, i_1) is not already into m, add (i_2, i_1) to m and add m_{i_1} to p;
8. if p is not empty, continue with step 5.

Defined $numMin_{\text{t}}$ as the number of minutiae of T lying inside the bounding box of S, $numMin_{\text{s}}$ as the number of minutiae of S lying inside the bounding-box of T, and $numMatches$ as the number of matches found by the algorithm, the score is evaluated as:

$$score_{\text{min}} = 100 \frac{numMatches^2}{numMin_{\text{t}} * numMin_{\text{s}}}$$

Matchers' Fusion. Given the two scores $score_{\text{fc}}$ and $score_{\text{min}}$, the overall score is calculated as a linear combination of them:

$$score = \alpha score_{\text{fc}} + \beta score_{\text{min}} \ .$$

If *score* exceeds a given threshold th_{score} the system considers the proposed fingerprint to be sufficiently similar to the enrolled one and the match succeeds, otherwise the match fails. Coefficients α and β have been determined with an *a posteriori* analysis as the ones which minimise the overall Equal Error Rate (EER) of the system.

4 Results

Effectiveness of our verification algorithm has been tested by submitting to it pairs of fingerprint images and by measuring its ability to discriminate between corresponding and non-corresponding ones in terms of FAR/FRR curves.

Tests have been conducted on a database of 55 live-scan fingerprints taken on a group of volunteers, with each fingerprint scan repeated ten times for a total of 550 images. The volunteers were completely unaware of biometrics related technology and scanner use, so they have been subject to a training phase of one minute with visual feedback, so to allow them to understand what was the right position and pressure of the finger for a good scan. Then, they have been asked to pose ten times the finger on the scanner in a natural way.

The obtained images have been analysed and matched in pairs using our algorithm, distinguishing matches between corresponding fingerprints (different images of the same finger) from matches between non-corresponding fingerprints. Figures 2(a) and (b) show the obtained joint (based on minutiae and on fingercode) scores' distributions in the two cases. In the two graphs, the X axis reports the score obtained with minutiae matching, which is higher when a higher number of matching minutiae is detected, while the Y axis reports the fingercode score, which is lower when the live-scan fingerprint is more similar to the on-card template. The same distributions are reported in the 3D plot of Figure 2(c) for convenience.

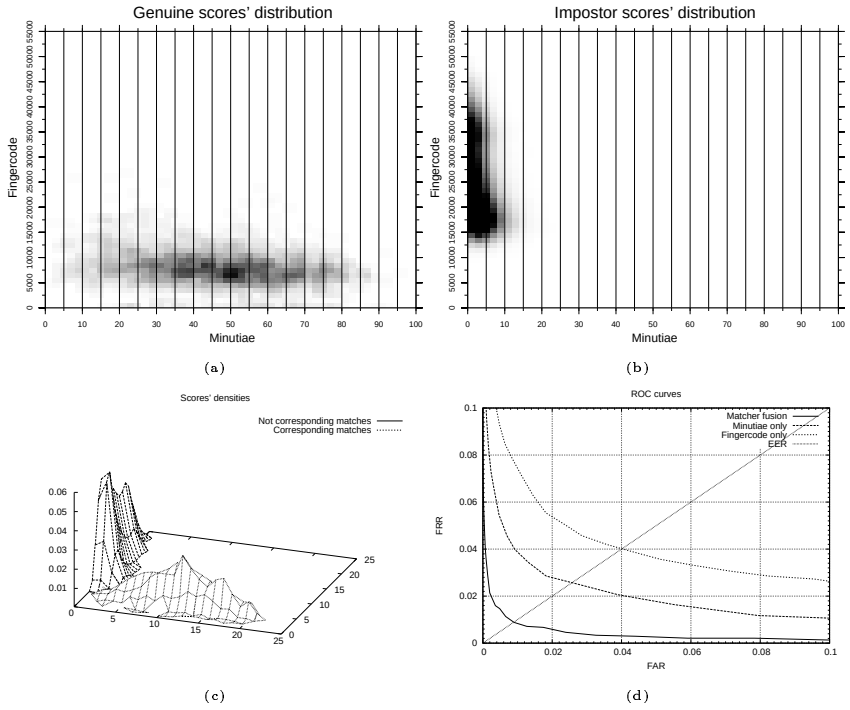

Figure 2. (a)-(c) Joint score distributions for genuine and impostor matches. (d) ROC curves

In Fig. 2(d) we compare the Receiver Operating Curves (ROCs) relative to each matcher separately and to their combination. These curves represent the FAR/FRR pairs that are obtained by continuously varying the score threshold of the matching algorithms. As the picture highlights, the hybrid technique results in a considerable increment of performance when compared to the results achieved singularly by the two matchers. In fact, in the hybrid ROC curve, the FRR value, for each possible FAR, is consistently lower than those obtained singularly by the two matchers. Furthermore, while the minutiae based and fingercode matchers obtain, respectively, an EER of about 2.3% and 4%, the combined matcher obtains an EER of about 0.8%. The combined matching algorithm requires an on-board computation time of about 11–12 seconds.

5 Implementation Notes

The described biometric authentication system has been developed, on the host side, as an extension to the MUSCLE Card [13] middleware, and on the card side as an extension to the MUSCLE Card JavaCard Applet. This framework defines a high level API that smart card aware applications can use to access smart card storage, cryptographic and PIN management services in a unified, card independent way. The framework also includes a JavaCard Applet allowing the middleware to use the on-card services by means of the protocol described in [10]. Briefly, the framework allows applications to manage on-board data containers (objects), cryptographic keys, and PIN codes. A security model allows to protect, on a per object and a per operation basis, objects and keys, by means of Access Control Lists (ACLs).

An extension mechanism has been embedded in the framework so to allow applications to enhance the basic protocol and Applet in order to support application specific extensions. Biometrics based authentication has been embedded in this context by allowing the access to on-card resources (e.g. reading an object or using a cryptographic key) only after a successful on-board fingerprint verification. Furthermore, the existing access control mechanism allows, by using ACLs, the possibility to combine the new authentication mechanism with traditional PIN based or challenge-response cryptographic based authentication. The used fingerprint scanner is FX2000 USB, by Biometrika s.r.l. (http://www.biometrika.it), providing a portable development kit and API for access to the acquired biometric data. The development platform has been a RedHat 7.3 Linux system.

6 Conclusions and Future Work

In this paper, a hybrid fingerprint matching mechanism has been introduced, designed with the aim of running onto a programmable smart card. Experimental results showed that, by taking advantage of the simplifications inherent to the application context and using ad-hoc designed data representations, it is possible to realise an on-board hybrid fingerprint matcher with an acceptable performance,

even into such scarce-resource devices as programmable smart cards, maintaining reasonable execution times. In a short future, it is scheduled to undertake investigations related to the feasibility of on-board multi modal authentication based on alternative means of biometrics.

References

1. Hachez, G., Koeune, F., Quisquater, J.J.: Biometrics, access control, smart cards: A not so simple combination. In: Proc. of CARDIS 2000. IFIP (2000)
2. Ratha, N., Connell, J., Bolle, R.: Enhancing security and privacy in biometrics-based authentication systems. IBM Systems Journal **40** (2001)
3. Scheuermann, D., Schwiderski-Grosche, S., Struif, B.: Usability of biometrics in relation to electronic signatures. Technical Report GMD-Report-118, GMD - Forschungszentrum Informationstechnik GmbH (2000)
4. Jain, A.K., Pankanti, S.: Automated fingerprint identification and imaging systems. In: Advances in Fingerprint Technology, 2^{nd} Edition. H. c. lee and r. e. gaensslen edn. Elsevier Science (2001)
5. Hong, L., Jain, A., Pankanti, S., Bolle, R.: Fingerprint enhancement. In Sarasota, F., ed.: Proc. 1st IEEE WACV. (1996) 202–207
6. Prabhakar, S.: Fingerprint classification and matching using a filterbank. PhD thesis, Michigan State University (2001)
7. Ross, A., Prabhakar, S., Jain, A.: Fingerprint matching using minutiae and texture features. In: Proceeding of the International Conference on Image Processing (ICIP). (2001) 282–285
8. Ross, A., Jain, A.K., Qian, J.Z.: Information fusion in biometrics. Lecture Notes in Computer Science **2091** (2001) 354–359
9. Marcialis, G., Roli, F., Loddo, P.: Fusion of multiple matchers for fingerprint verification. In: Proc. of Workshop su Percezione e Visione delle macchine, 8^{vo} Convegno dell'Associazione Italiana per l'Intelligenza Artificiale AI*IA02, Siena, Italy (2002)
10. Cucinotta, T., Natale, M.D., Corcoran, D.: A protocol for programmable smart cards. In: Proceedings of the 14^{th} International Workshop on Database and Expert Systems Applications (DEXA'03), Prague, Czech Republic, IEEE Computer Society Press (2003) 369–374
11. Pettersson, M., Harris, M.: Whitepaper: Match-on-card for java cards. Precise Biometrics (2002)
12. Wamelen, P.V., Li, Z., Iyengar, S.: A fast algorithm for the point pattern matching problem. Technical Report 1999-28, Louisiana State University, Dept. of Mathematics (1999)
13. Corcoran, D., Cucinotta, T.: MUSCLE card framework – application programming interface, v1.3.0 (2001)