Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition Letters 28 (2007) 2427–2436
Pattern Recognition Letters
www.elsevier.com/locate/patrec

Symmetric hash functions for secure fingerprint biometric systems

Sergey Tulyakov, Faisal Farooq *, Praveer Mansukhani, Venu Govindaraju

Center for Unified Biometrics and Sensors, State University of New York at Buffalo, Amherst, NY 14228, USA

Received 26 June 2006; received in revised form 8 August 2007
Available online 19 August 2007

Communicated by J.A. Robinson

Abstract

Securing biometrics databases from being compromised is an important research challenge that must be overcome in order to support widespread use of biometrics based authentication. In this paper we present a novel method for securing fingerprints by hashing the fingerprint minutia and performing matching in the hash space. Our approach uses a family of symmetric hash functions and does not depend on the location of the (usually unstable) singular points (core and delta) as is the case with other methods described in the literature. It also does not assume a pre-alignment between the test and the stored fingerprint templates. We argue that these assumptions, which are often made, are unrealistic given that fingerprints are very often only partially captured by the commercially available sensors. The Equal Error Rate (EER) achieved by our system is 3%. We also present the performance analysis of a hybrid system that has an EER of 1.96% which reflects almost no drop in performance when compared to straight matching with no security enhancements. The hybrid system involves matching using our secure algorithm but the final scoring reverts to that used by a straight matching system.

Keywords: Biometrics; Fingerprints; Hashing; Security; Cancelable biometrics

1. Introduction

Securing a biometric template is a critical step in the successful implementation of biometrics based authentication systems. Typically, biometric templates are stored unprotected in a central database. Even if the stored templates are encrypted, matching continues to be performed using decrypted templates where the decryption process itself can be compromised. An analogy can be made with password based authentication systems which come under eavesdropping attacks (e.g., man-in-the-middle) during transfer over a network (Schneier, 1996). To prevent such attacks, plain-text passwords are hashed, and only the hash values are stored in the database and transmitted across networks.

A hash function H is a transformation that takes an input m and returns a value h (called the hash value); $h = H(m)$. Hash function H is said to be a one-way function if it is hard to invert (Schneier, 1996), that is, given a hash value h, it is computationally infeasible to find some input x such that $H(x) = h$.

We have developed a method for biometric data which is similar to password encryption and hashing and involves the following steps. Biometric matching is performed using hashed features instead of the original template (Fig. 1). Fingerprints are obtained using an online scanner. The minutia features are located and hashes of the minutia subsets are constructed. These operations of finding minutiae and hashes can potentially be incorporated into the scanner itself, so that only the hashes will need to be transmitted and stored in the database. During verification, new hash values are produced by the scanner and are matched with those stored in the database. Matching can be performed either on the client side or on the server side.

In this paper we extend our earlier work (Tulyakov et al., 2005) by introducing additional methods of securing and personalizing the hash for the fingerprint data.

* Corresponding author. Fax: +1 7166456176.
E-mail addresses: ffarooq2@cubs.buffalo.edu (F. Farooq).

0167-8655/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2007.08.008

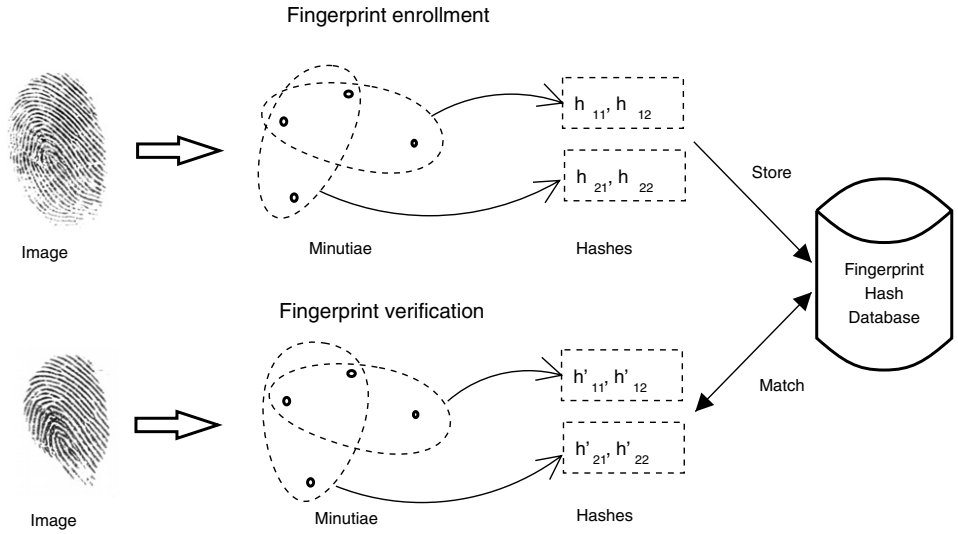

Fig. 1. Securing fingerprint information.

2. Challenges

The hash value for text passwords completely changes even if a single character in a password is changed. This is due to a desirable property of the hash algorithm known as the *avalanche effect*. Hashing is still feasible in case of passwords because the authentication is an *all-or-none* paradigm and access is granted only if the entire password that is offered matches correctly. Also, in password protected systems, in case the password database is compromised, a new set of passwords can be set up. Biometric systems, on the other hand, are probabilistic. Authentication is based on scores that can range between 0% and 100%. In case the biometric data is hashed, even a slight change in the acquisition of the biometric (a very likely scenario) can lead to a totally different hash value. Thus, it may not match the original within the same matching threshold as that for the straight unhashed scenario. Therefore, the hash-based system must adhere to the following additional properties:

- similar fingerprints should have similar hash values,
- different fingerprints should not have similar hashes,
- rotation and translation of the original template should not have a big impact on hash values,
- partial fingerprints (with missing core and delta) should be matched if sufficient minutiae are present.

3. Previous work

The situation we are facing here is analogous to a password based authentication system which authenticates even if the password provided is "*almost the same*". Error-correcting codes (Peterson and Weldon, 1972) have been successfully utilized in such situations. Davida et al. (1998) describe an algorithm where error-correcting digits are generated from the biometric data and stored in the database during registration. During the authenticating stage, biometric data is combined with the stored error-correcting digits and correction is performed. The amount of correction required is a measure of the authentication confidence. This algorithm was later modified as a fuzzy commitment scheme in the work of Juels and Wattenberg (1999). Kuan et al. (2005) presented a slightly different method for extracting cryptographic keys from dynamic handwritten signatures. An approach for face templates is presented by Kevenaar et al. (2005) in which they generate binary feature vectors from biometric face data and gain security by using helper data introduced into this bit sequence.

However, none of the above mentioned approaches can be directly extended to fingerprints where the minutia positions themselves are features. This presents additional challenges for designing hashes as minutia sets of two fingerprints, in most instances are not exactly the same. It is also impossible to introduce an order in the minutia set, and global transformation parameters are usually present between corresponding minutiae. Error-correcting codes require that the original sequence be ordered in some fashion, so as to locate and then correct the errors in a modified sequence. The fuzzy vault algorithm (Juels and Sudan, 2002) improves upon the fuzzy commitment scheme in addressing these challenges. The security of the algorithm relies on the introduction of chaff points (false minutiae). An attacker must find a subset of points intersecting with the non-chaff point set. Thus, more chaff points provide better security, but reduce the vault unlocking performance.

The application of the fuzzy vault to fingerprint identification appears in the work of Clancy et al. (2003). It shows realistic expectations on the numbers of chaff points and associated attack complexity. The algorithm uses the assumption that fingerprints are aligned, and corresponding minutiae have similar coordinates. Uludag et al.

(2005) propose a fuzzy vault scheme by adding extra chaff points and securing the template by a standard 128-bit AES algorithm. However, the method still requires pre-aligning the test and stored fingerprint and achieves FAR of about 20% on a test set of 100 fingerprints.

Linnartz and Tuyls (2003) and Tuyls et al. (2005) propose a technique that assumes complete alignment of template and test biometric data in addition to assuming minimal effect of noise on the securing functions. Soutar et al. (1999) construct a special filter in the Fourier space to encode key data. The data can be retrieved only by presenting a similar fingerprint image to the decoder. The matching procedure is based on correlation, thus translations of images are possible but not rotations. More recently, Uludag and Jain (2006) presented an advancement of the earlier algorithm with a genuine accept rate of about 72%. However, the alignment is highly prone to error, and does not work on poor quality or partial images. Ratha et al. (2007) in a recent work describe surface folding transformation functions that can work with existing point-based matchers. However, they require precise locations of the singular points (core and delta) for the alignment of the fingerprints in order to guarantee repeatability of the transformations. Teoh et al. (2004) present a two-factor authentication system with high accuracies, but the algorithm also requires the precise location of the core for feature extraction. This prevents compatibility of such systems with existing databases and fingerprint scanners.

A comprehensive survey of previous work in securing biometric data can be found in (Uludag et al., 2004). However all the reported methods are primarily limited by requirements of pre-alignment or location of singular points. Our objective is to overcome this limitation.

4. Motivation

The main difficulty in producing hash functions for fingerprint minutiae is the inability to normalize the fingerprint data. If the fingerprint data is not normalized, then values of the hashing functions are destined to be orientation/position- dependent. The way to overcome this difficulty is to have hash functions as well as the matching algorithm deal with transformations of the fingerprint data. This is precisely the approach we have developed in this paper.

4.1. Minutiae based matching

In fingerprint based biometric authentication systems, minutiae based matching has become a *de facto* standard. A fingerprint is comprised of a series of ridges and furrows on the surface of the finger. The uniqueness of a fingerprint is determined by analyzing the contours as well as the minutiae points which are local ridge characteristics that occur at either a ridge bifurcation or a ridge ending (ANSI/INCITS 378-2004,) (Fig. 2, left). Correlation based techniques are usually inefficient and at times infeasible because of their high sensitivity to translation and rotation.

The task of fingerprint matching requires that the two prints be aligned and the number of matching minutiae points determines the goodness of match. In our work we use ideas similar to Germain et al. (1997) and Jea et al. (2004) to combine the results of 'localized matchings' into

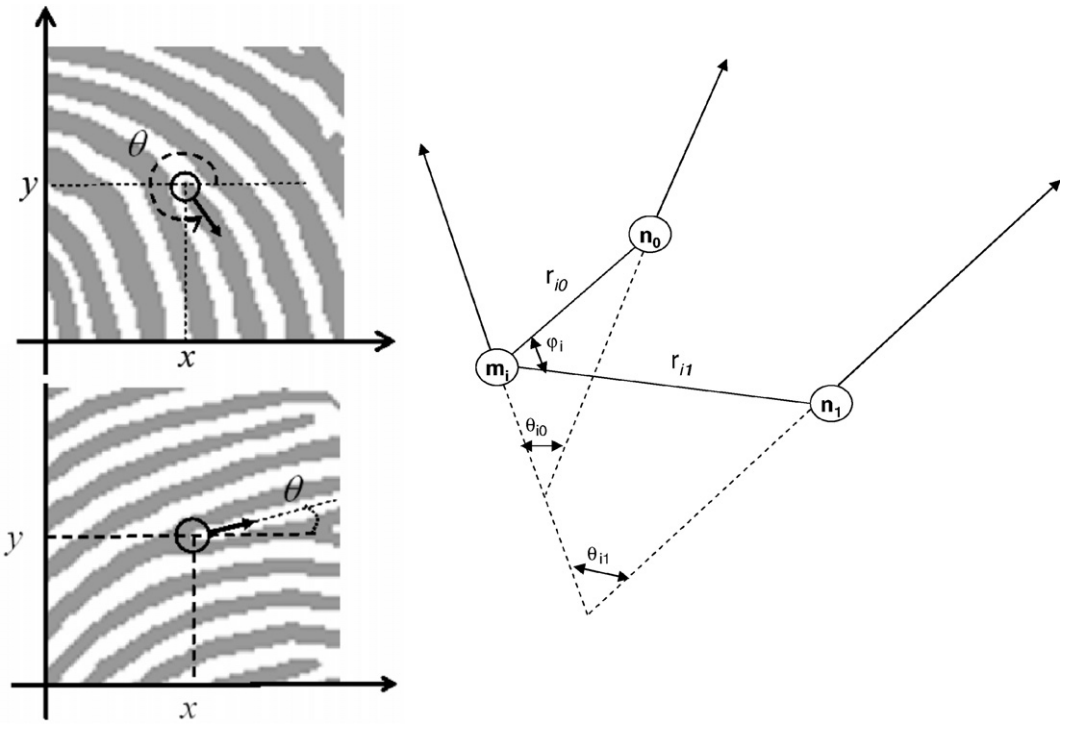

Fig. 2. Left: Minutia angles at a ridge and bifurcation. Right: Secondary features based on the minutia m_i and its nearest neighbors n_0 and n_1.

the fingerprint recognition algorithm to avoid global alignment. Localized matching consists of matching minutia triplets using such features as angles and distances between minutia points. For each minutia feature vector of length 3 (x, y, θ) and its two nearest neighbors, a secondary feature vector $s_1 = \{r_{10}, r_{11}, \theta_{10}, \theta_{11}, \phi_1\}$ is generated based on the Euclidean distances and orientation difference between the central minutia and its nearest neighbors (Fig. 2, right). For localized matching, we only keep track of limited information about the matched neighborhoods, so that the minutia positions cannot be restored from the transformed data.

Global matching is essentially about finding a cluster of localized matchings with similar rotation (r) and transformation (t) parameters. Unlike the fingerprint vault algorithm (Clancy et al., 2003) our algorithm performs hashing of not only the enrolled fingerprint, but of the test fingerprint as well. Thus hashing can be incorporated into the scanner itself, and the original fingerprint data never needs to be transmitted or stored in a database.

4.2. Symmetric hash functions

A small change in the input (missing information, noise or a change in the order of the input etc.) can cause a significant change in the hash value. A certain class of hash functions can, however, be formulated that are invariant to the order in which the input pattern is presented to the hash function. Such hash functions are known as order-independent or *symmetric* hash functions. Consider an input sequence $X = x_1x_2x_3\ldots x_n$ and the following two hash functions (examples)

$$H(X) = k_1x_1 + k_2x_2 + \cdots + k_nx_n, \quad k_1 \neq k_2 \cdots \neq k_n \tag{1}$$

$$H^m_{\text{sym}}(X) = x_1^m + x_2^m + \cdots + x_n^m \tag{2}$$

If the order of the input is changed to $X = x_2x_3x_n\ldots x_1$, the first function yields a different hash value where as the second remains unchanged. We can generate similar hash functions (like (2)) that are symmetric. Moreover, arbitrary combinations of more than one hash function yield new hash functions. Thus, we can have a whole family of symmetric hash functions by combining the elementary symmetric functions of (2): $H_{\text{sym},f}(X)' = f(H^1_{\text{sym}}(X), \ldots, H^n_{\text{sym}}(X))$. This property of symmetric hash functions can be used for hashing the fingerprint minutiae (or any set of unordered points).

5. Hash functions of minutia points

We represent minutia points as complex numbers $\{c_i\}$. We assume that two fingerprints of the same finger can have different position, rotation and scale, coming from (possibly) different scanners and different positioning of the finger on the scanner. The transformation of one fingerprint to another can be described by the complex function $f(z) = rz + t$ (Fig. 3). z represents the minutia point c_i as $x_i + y_i$ located at coordinates (x_i, y_i). r and t represent the scalar rotation and translation parameters of the accidental shift of points under the registration and authentication scans. In our approach we construct hash functions and the corresponding matching algorithm so that the accidental shifting is taken into account. Additionally, we do not rely on a specific order of minutiae because we want our hash functions to be independent of this order. We consider symmetric complex functions as our hash functions.

Specifically, given n minutia points $\{c_1, c_2, \ldots, c_n\}$ we can construct the following m symmetric hash functions

$$\begin{aligned} &h_1(c_1, c_2, \ldots, c_n) = c_1 + c_2 + \cdots + c_n \\ &h_2(c_1, c_2, \ldots, c_n) = c_1^2 + c_2^2 + \cdots + c_n^2 \\ &\cdots \\ &h_m(c_1, c_2, \ldots, c_n) = c_1^m + c_2^m + \cdots + c_n^m \end{aligned} \tag{3}$$

If the number of hash functions (m) is less than the number of minutia points (n) participating in the construction of the hash function, then it is not possible to restore the original minutia positions given the hash values. Suppose another image of the fingerprint is obtained using the above described transformation $f(z) = rz + t$. Locations of the corresponding minutia points are $c'_i = f(c_i) = rc_i + t$. Hash functions of the transformed minutiae can be rewritten as

$$\begin{aligned} &h_1(c'_1, c'_2, \ldots, c'_n) \\ &\quad = c'_1 + c'_2 + \cdots + c'_n \\ &\quad = (rc_1 + t) + (rc_2 + t) + \cdots + (rc_n + t) \\ &\quad = r(c_1 + c_2 + \cdots + c_n) + nt \\ &\quad = rh_1(c_1, c_2, \ldots, c_n) + nt \\ &h_2(c'_1, c'_2, \ldots, c'_n) \\ &\quad = c'^2_1 + c'^2_2 + \cdots + c'^2_n \\ &\quad = (rc_1 + t)^2 + (rc_2 + t)^2 + \cdots + (rc_n + t)^2 \\ &\quad = r^2(c_1^2 + c_2^2 + \cdots + c_n^2) + 2rt(c_1 + c_2 + \cdots + c_n) + nt^2 \\ &\quad = r^2h_2(c_1, c_2, \ldots, c_n) + 2rh_1(c_1, c_2, \ldots, c_n) + nt^2 \\ &\ldots \end{aligned} \tag{4}$$

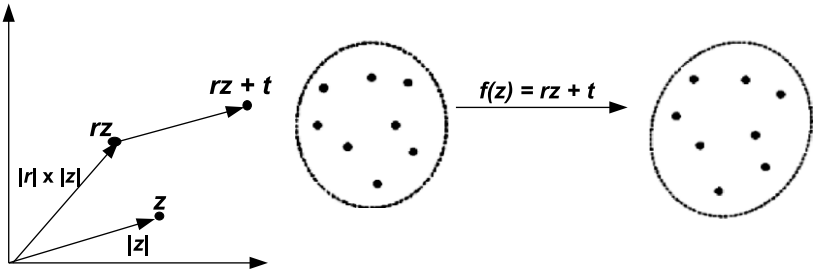

Fig. 3. Minutiae as represented in the complex plane. f represents the accidental shifting of minutia points.

Let us denote the hash values of the minutia set of one fingerprint as $h_i = h_i(c_1, c_2, \ldots, c_n)$ and the hash values of the corresponding minutia set of another fingerprint as $h'_i = h_i(c'_1, c'_2, \ldots, c'_n)$. Eq. (4) now becomes

$$\begin{aligned} h'_1 &= rh_1 + nt \\ h'_2 &= r^2 h_2 + 2rth_1 + nt^2 \\ h'_3 &= r^3 h_3 + 3r^2 t h_2 + 3rt^2 h_1 + nt^3 \\ &\ldots \end{aligned} \tag{5}$$

Eq. (5) has two unknown variables r and t. If we take into account the errors introduced during the fingerprint scanning and minutia search, the relation between the hash values of the enrolled fingerprint $\{h_1, \ldots, h_m\}$ and the hash values of the test fingerprint $\{h'_1, \ldots, h'_m\}$ can be represented as

$$h'_i = f_i(r, t, h_1, \ldots, h_n) + \epsilon_i \tag{6}$$

Matching between the hash values of the enrolled fingerprint $\{h_1, \ldots, h_m\}$ and the hash values of the test fingerprint $\{h'_1, \ldots, h'_m\}$ amounts to finding r and t that minimize the errors (ϵ_i). During implementation we have considered minimization of error functions $\epsilon = \sum \alpha_i |\epsilon_i|$, where weights α_i are chosen empirically.

6. Global fingerprint matching using hash functions

It turns out that using hash functions with respect to the minutia set of the whole fingerprint is impractical. Even the small difference in minutia sets of two prints of the same finger produce significant differences in hash values. Further, the higher order hash values tend to change in a large measure with even small change in positions of the minutia points. Thus, this is contrary to the desirable properties we motivated in Section 2.

To overcome these difficulties we perform matching only on localized sets of minutia. Global matching of two fingerprints is taken as a collection of the localized matchings with similar transformation parameters r and t. As in the base fingerprint matcher (Jea et al., 2004), the localized set is determined by a particular minutia and a few of its neighbors. The hashes are calculated for each localized set. The total hash data extracted consists of a set of hashes

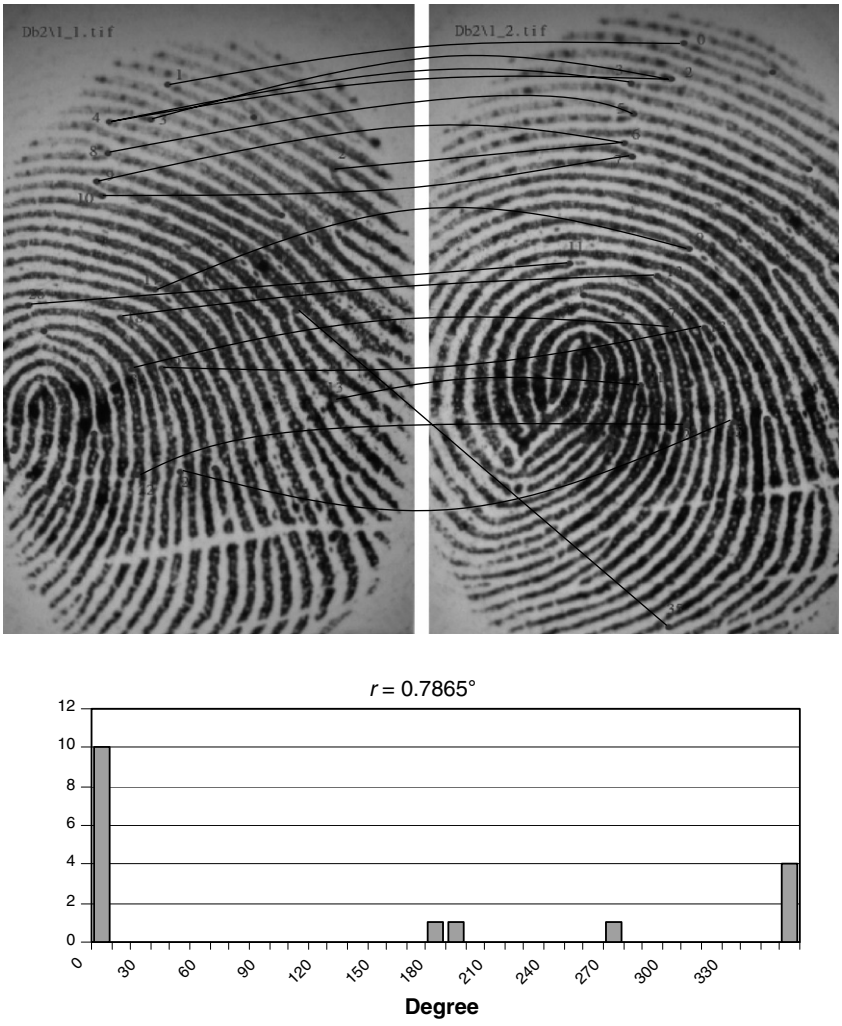

Fig. 4. Voting based recovery of the rotation between prints.

$\{h_{i,1},\ldots,h_{i,m}\}$, $i=1,\ldots,k$, where k is the total number of localized minutia sets.

During matching of two hash sets we first perform a match of all the localized sets in one fingerprint with all the localized sets in another fingerprint. The matches with the highest confidences are retained. Assuming that a particular match is correct (iteratively), we find how many other matches have similar transformation parameters. Fig. 4 describes the exhaustive voting procedure for a pair of fingerprints. We observe that most hypotheses vote for the rotation of 0.7865° and this is selected as the rotation parameter r. Similarly, the translation parameter t is established. The match score is a composite of the number of close matches and the confidences of those matches.

7. Experimental analysis

We tested our system on the 2002 Fingerprint Verification Competition DB1 (Second International Fingerprint, 2002) database. The dataset consists of 110 different fingers and 8 impressions for each finger. There are a total of 880 fingerprints (388 pixels by 374 pixels) at 500 dpi with varying image quality. In accordance to the protocols of *FVC*2002, we used only the first 100 individuals to evaluate the FAR (False Accept Rate) and FRR (False Reject Rate). FRR is computed based on the total number of genuine tests (compare each impression with the seven others for all individuals) and equals $\frac{(8*7)}{2}*100=2800$. FAR is computed based on comparing the first impression of each individual with the first of the others and thus the total number of impostor tests equals $\frac{(100*99)}{2}=4950$.

We have conducted experiments with different configurations, using different number of minutia points (n) and hashing functions (m). The following configurations were considered:

(1) $n=2$, $m=1$. For each minutia point we find its nearest neighbor, and the hash function $h(c_1,c_2)=\frac{c_1+c_2}{2}$.
(2) $n=3$, $m=1$. For each minutia point we find two nearest neighbors and the hash function $h(c_1,c_2,c_3)=\frac{c_1+c_2+c_3}{3}$.
(3) $n=3$, $m=2$. For each minutia point we find the three nearest neighbors, and for each minutia triplet including the original minutia point we construct two hash functions $h_m(c_1,c_2,\ldots,c_n)=c_1^m+c_2^m+\cdots+c_n^m$, where $m=1,2$.

Let us consider configuration 3 in detail. Given a minutia triplet represented by complex numbers (c_1,c_2,c_3), we find the center of the triangle formed by this triplet. The center is represented by the complex number $T=\frac{c_1+c_2+c_3}{3}$. Such triangle centers of all minutia triplets are used for hashing. The template and the test fingerprint are aligned to calculate the matching scores. Thus, if a fingerprint is represented in the minutia space by a set of minutia points $\{m_1,m_2,\ldots,m_n\}$, this operation maps it onto a new space where it is now represented by a set of triangle centers $\{T_1,T_2,\ldots,T_k\}$. The task of reversing this hash function would involve finding the actual minutia point locations given these triangle centers. We compared the FAR and FRR performance with the fingerprint matching algorithm developed in (Jea et al., 2004) and using the same set of fingerprints with identically extracted minutiae points. Also, since in configurations (1) and (2) we simply deal with a new set of minutia points, we used the matching algorithm of Jea et al. (2004).

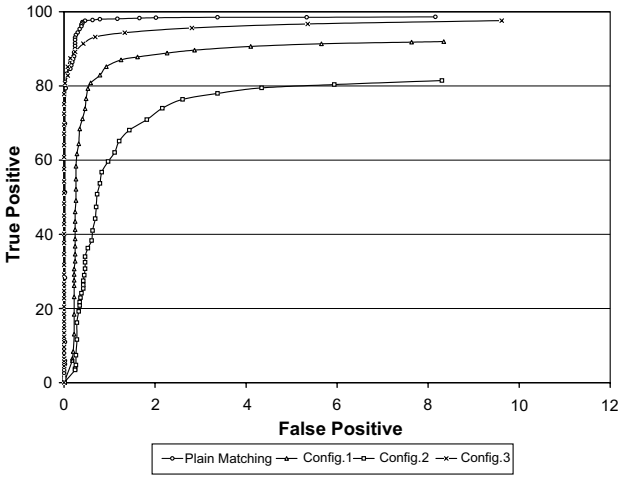

Fig. 5. ROC curves for the baseline system (Jea et al., 2004) and the different experimental configurations.

We achieved an equal error rate (EER, point where FAR = FRR) of 3% compared to 1.7% for straight matching in the minutia space. The ROC characteristics of the straight matching and the different configurations of our algorithm are shown in Fig. 5. The accuracy is slightly lower than the baseline system with the trade-off benefit of securing the fingerprint data against (hacker) inversion.

8. Security of the algorithm

The main purpose of the proposed algorithm is to protect the original fingerprint and minutiae locations from an attacker. Let us see if it is possible to reconstruct the minutia positions given the stored hash values. Since the number of hash values for each local minutia set is less than the number of these minutiae, it is not possible to get the locations using only the information of any one local set. Although, it may seem possible to construct a (big) system of equations involving all the hashes. It is not known which minutia participated in the creation of a particular hash value. Thus the non-invertibility is still maintained.

The problem is illustrated in Fig. 6. Two triplet centers are formed from 4, 5 and 6 minutia points. Thus during construction of an equation system for reverse engineering the minutia positions, the attacker would face the problem of deciding the number of participating minutiae, in addition to matching the minutia to the triplet centers.

Hill-climbing type attacks (Uludag and Jain, 2004) will have a difficult time to make a match since the varying

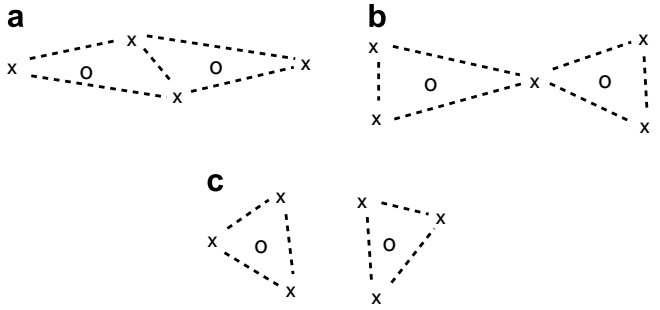

Fig. 6. Different number of minutiae (crosses) can participate in the creation of two triplet centers (circles).

minutia position would effect some triplets, thus influencing the matching score in a more complex way. Furthermore, even if an attack succeeds and a match is found, the resulting minutiae locations will be different from the original. In such a situation, change of hashing algorithm will make the reconstructed fingerprint unmatchable. Brute-force search on all the nearest neighbors of a triangle center for the above method may be computationally feasible, however, using higher order hashes instead of simply neighboring minutiae can render such attacks ineffective. If the minutia positions are floating point and not integers, then once again the brute-force method becomes computationally intractable. The float positions of minutia could be estimated by the minutia extraction algorithm, or in case of integer positions, these positions might be randomly perturbed. Another method might be to reduce the number of information bits in the hash value with respect to the actual fingerprint template thus making it infeasible to do a brute-force attack even on the whole fingerprint image. Whereas these methods only utilize the fingerprint minutiae, in the following sections we present methods for using additional information (keys, personal hashes etc.) that can harden the fingerprint hash.

Ratha et al. (2001) describe how the strength of a fingerprint with K discrete minutia positions and d associated directions can be associated with $\log_2(Kd)$ bits of information and have a brute-force strength of about 70–80 bits. Our approach actually increases the bit-strength as the number of points increases using the triplet centers. Thus, it has a security of the same order as that of a plain fingerprint to a brute-force attack. However, the invertibility of the hash of the actual fingerprint in our algorithm is intractable.

9. Cancelable biometric

Our algorithm for the hashing of fingerprint templates eliminates the possibility of an attacker learning the original minutia positions. Although we consider it an extremely difficult task, an adversary might construct an artificial template producing similar hash values, but having different minutia positions. Thus we need to expand our algorithm to make the fingerprint hashes cancelable. This can be achieved by re-enrolling people using a different set of hash functions.

In order to enhance the security, systems often implement a two-level authentication where a user in addition to the biometric provides a key which is stored on a card or enters it on a keypad. Also, this key can be reissued in case of a successful attack. In this section we present ways to increase the security of the hashing method by an exponential factor. This can be done by embedding a secret key into the hashing process. The key can be based on a token that the user carries or a password that the user remembers. It may even be based on another biometric modality, thus making the key personal. To achieve a cancelable biometric algorithm we need to provide a way to automatically construct and use randomly generated hash functions. The presented set of hash functions is an 'algebraic basis' in the set of polynomial symmetric functions. Thus, we are able to express hash functions of transformed minutia set through the original set of symmetric functions. This is a clue to constructing new hash functions of the same type. We can essentially take an arbitrary algebraic basis of symmetric polynomials of degree less than or equal to m, $\{s_1,\ldots,s_m\}$ as our hash functions. Then the hash functions of the transformed minutiae, $s_i(rc_1+t,\ldots,rc_n+t)$, will still be symmetric functions of the same degree with respect to the variables $c_1,\ldots,c_n$. Thus, hashes of transformed minutia can be expressed using the original hashes, $s'_i = s_i(rc_1+t,\ldots,rc_n+t) = F_i(r,t,s_1,\ldots,s_m)$ for some polynomial functions F_i. These equations will allow matching localized minutia sets, and finding corresponding transformation parameters.

9.1. Two-factor authentication

Let us assume that we compute a hash value for each triplet of minutiae (c_1,c_2,c_3). For each such triplet, we can choose from one of several symmetric hash functions such as

$$
\begin{aligned}
h_1(c_1,c_2,c_3) &= (c_1+c_2+c_3)\\
h_2(c_1,c_2,c_3) &= (c_1c_2+c_2c_3+c_1c_3)\\
h_3(c_1,c_2,c_3) &= c_1c_2c_3\\
h_4(c_1,c_2,c_3) &= (c_1-c_2)^2+(c_2-c_3)^2+(c_1-c_3)^2, \text{ etc.}
\end{aligned}
$$

Any linear combination of these functions will also yield a symmetric hash function. Thus for any triplet, we have several functions $h_1,h_2\ldots h_k$ from which we can derive the transformation. Instead of choosing the hash function in a deterministic way, the complexity of the transformation and hence the resulting security can be enhanced if we could choose several of these hash functions simultaneously in a random order. Thus, for each triplet $T_1,T_2\ldots T_N$ we can associate a corresponding hash function $H_1,H_2\ldots H_N$. The association can be based on a secret key K. The key specifies the association between the triplet T and the corresponding hash H as shown in Fig. 7.

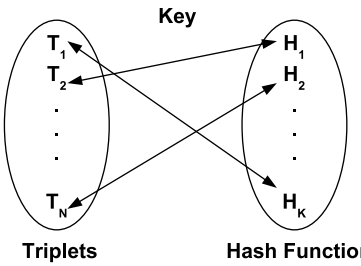

Fig. 7. Associating the minutiae triplets with hash functions.

However, in order to successfully verify the individual during authentication, the resulting triplets T'_1, T'_2 must also be associated with identical hash functions. The problem occurs because we do not know the association between T_1, T'_1 before hand. To overcome this problem, each triangle or triplet T can be represented parametrically by specifying three parameters such as – two sides and one angle, or one side and two angles etc. Let us represent these by p_1, p_2, p_3 in general. Thus each possible triangle now exists as a point in the parametric space (Fig. 8).

All triangles with similar geometries lie close together in this parametric space. Thus, given any triplet T we determine the point P where it lies in the parametric space. Any triplet T' that is geometrically similar will lie in close proximity of P as shown by the circles in Fig. 8. Further we divide the parameter space into non-overlapping cells (Fig. 9). The cells are shown in $2D$ for simplicity. Each cell is assigned a specific hash function.

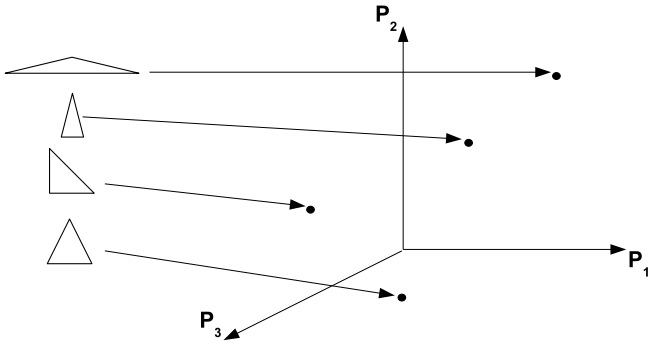

Fig. 8. Triangles as points in the parameter space.

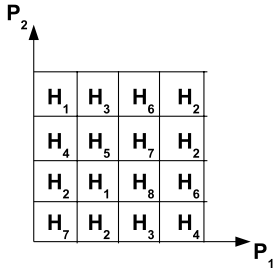

Fig. 9. Associating the hash functions with cells in the parameter space.

The association between the hash function and the cell are now contained in the secret key. Let two instances of the key be $H_2H_4H_8H_1H_3H_1$ and $H_3H_2H_7H_3H_1H_6$. The length of the key is determined by how we subdivide the triangle space into cells. Let us assume that there are c such cells in all. This arrangement solves the original problem of triplet association. If a triplet T exists in the reference fingerprint and appears at T' (T with slight distortion) in another instance of the print, it falls in close proximity of the original triplet in the triangle space. Due to the spatial proximity it also falls in the same cell as the original triplet T and hence gets assigned the same hash function as before due to quantization of the triangle space. The proposed solution increases the security of the hashing function by rendering brute-force attacks infeasible.

9.2. Personalizing and reissuing

While the number of symmetric functions possible for each triplet is clearly infinite, it is not clear as to how many of these functions can be chosen such that the transformation is still meaningful. Let us assume it is some finite (perhaps large) number N. For somebody who has the original biometric, the task of circumventing the system reduces to trying out all of the N hash functions. By introducing the key K, there are N possible hash functions for each cell in the triangle space. Thus the total number of possible hash combinations is now $N \times N \times N \ldots (c \text{ times}) = N^c$. Thus, by introducing the secret key K, we are exponentially multiplying the total number of possibilities of hash functions and increasing the computational complexity of a brute-force attack by the same amount. This key can be based on another biometric modality such as face or iris or its convolution by some signal. In case of compromise of the database, the keys can be reissued and a different set of hash functions can be chosen as shown earlier, thus ensuring that the biometric system is cancelable.

10. Performance analysis

The slight loss in the accuracy of the secure system as compared to the straight version can be attributed to factors such as reduction in the number of points being matched. However, the total number of hashed values is not reduced in the same proportion since the same minutia can participate in the production of more than one triplet as shown in Fig. 6. Thus, total size of the stored hash values can be even larger than the size of the original fingerprint template. The decrease in the accuracy might also have been caused by the loss in information when keeping a reduced number of variables based on the minutia triplets. For every three neighboring minutia points we have reduced the number of variables to 4 (2 complex numbers) instead of the original 6. For example, the average number of minutia matched for a genuine match in the baseline version was observed to be about 25.9. In the secure version the average number of triplet centers matched for genuine

tests was 57.5. There can be additional reasons for the slight performance drop, such as difficulty in matching localized hashed values.

In order to evaluate the performance of the secure matching algorithm vis-a-vis the straight matching, we performed experiments where the transformation parameters r and t were determined by our algorithm. These parameters were then used as the transformation parameters for the straight version. For this hybrid setup, an EER of 1.96% was achieved (Fig. 10).

In the region of interest (i.e. FAR $< 10^{-2}$), the secure system has a lower (better) FRR. Hence, although the secure system is doing better in terms of FAR near the point of equal error, it is doing slightly worse in terms of FRR. Fig. 11 shows an example where the secure system falsely rejects a user that the straight system correctly grants access. As we have observed, the difference in the number of overlapping minutia between the two prints is large and this is magnified when the fingerprint is mapped on to the hashed space (e.g. the triangle centers). Thus, the straight matching system is able to match more minutia but our algorithm matches fewer triangle centers as many of them overlap. Thus, we are more conservative in the matching leading to a lower FAR but incur a slightly higher FRR. Table 1 gives a comparison between the three modes. The comparable number of minutia matched in the straight version and the hybrid system suggest that indeed the secure system performs as well in terms of finding the transformation parameters and matching the minutia.

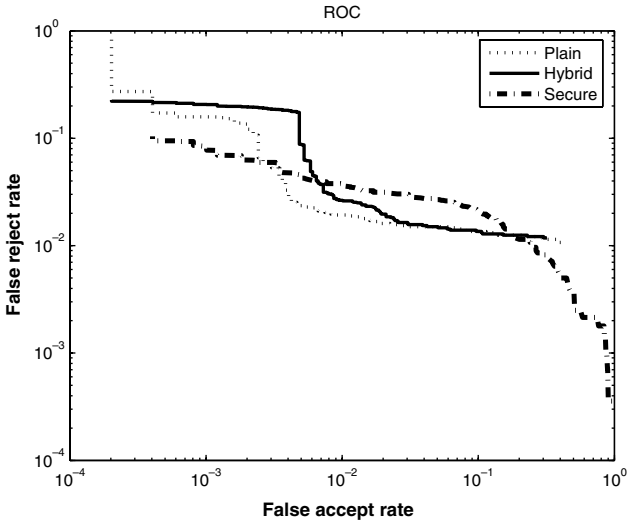

Fig. 10. Comparing the ROC curves of the plain, secure and hybrid systems.

Table 1
Comparison of the plain, secure and hybrid verification systems

	Plain	Secure	Hybrid
Average points matched	25.90	57.50	24.55
EER%	1.7	3.0	1.96

11. Conclusions

We have presented a method to secure fingerprint templates by using innovative symmetric hash functions. Such symmetric functions can be utilized for any biometric modality where the features are unordered as is the case with fingerprint minutia.

We have described the successful implementation of a secure authentication system with performance comparable to straight matching systems. We have also presented methods to cancel and reissue the biometric and to personalize the hash values based on keys that could be potentially derived from other biometric modalities. Our method does not make any assumptions regarding the pre-alignment of fingerprints or about the ordering of minutiae points based on locations of core and delta points. Thus, in contrast to all the previous methods described in the literature, our method is the most general,

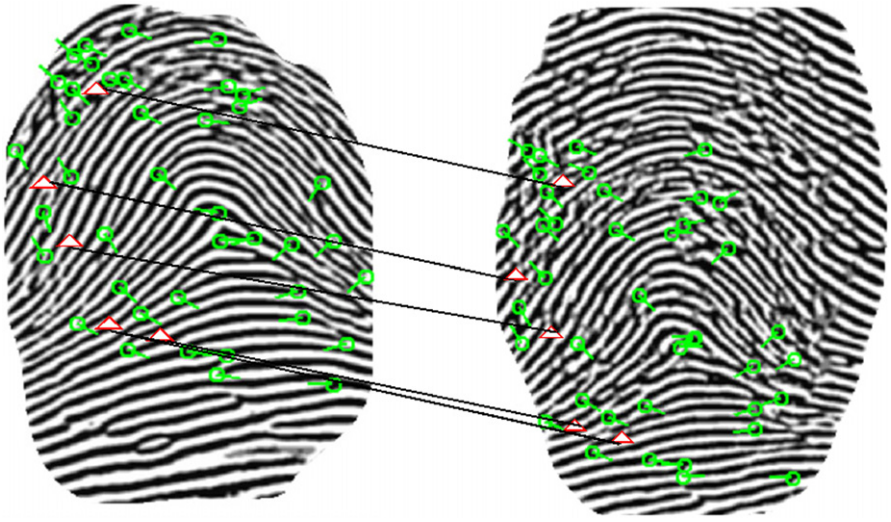

Fig. 11. An example genuine matching pair falsely rejected by our system.

and in case of partial fingerprints where locations of core and delta points are often unavailable, it is the only practical method.

A conceptual model that addresses variability of feature representations, ordering of features, and the need for localization must be systematically investigated for all biometric modalities to achieve security, cancellability, and privacy. Whereas in the case of fingerprint modality, minutiae based feature representation is a (NIST) standard which is widely accepted, other biometric modalities do not enjoy a single standard representation of features. Thus the method described in this paper does not readily scale to other modalities.

Our method is generalizable to biometric modalities that use locations of certain image artefacts as primary features. For example, online signature matching algorithms often use the sequence of (x, y) coordinate locations as a function of time as features. Hash functions similar to ones constructed with minutia locations can be constructed for pen trajectory locations as well. The challenge will be the spatial localization of groups of features given the wide variability in the dimensions of signatures across people. Also, the order of the points within a local area in a signature carry vital information which may need to be encoded.

Voice print matching, like signature, offers the temporal dimension by providing a natural sequencing of features. However, the features themselves are more complex than simple location of certain attributes. We will investigate approaches suitable for non-behavioral biometrics as well where the temporal dimension does not exist.

References

ANSI/INCITS 378-2004, 2004. Information Technology – Finger Minutiae Format for Data. Interchange, InterNational Committee for Information Technology Standards.

Clancy, T., Lin, D., Kiyavash, N., 2003. Secure smartcard-based fingerprint authentication. In: ACM Workshop on Biometric Methods and Applications (WBMA 2003).

Davida, G., Frankel, Y., Matt, B., 1998. On enabling secure applications through on-line biometric identification. In: Proc. IEEE 1998 Symp. on Security and Privacy, Oakland, CA.

Germain, R., Califano, A., Colville, S., 1997. Fingerprint matching using transformation parameter clustering. IEEE Comput. Sci. Eng. 4 (4), 42–49.

Jea, T.-Y., Chavan, V.S., Govindaraju, V., Schneider, J.K., 2004. Security and matching of partial fingerprint recognition systems. In: SPIE Defense and Security Symposium.

Juels, A., Sudan, M., 2002. A fuzzy vault scheme. In: IEEE Internat. Symposium on Information Theory.

Juels, A., Wattenberg, M., 1999. A fuzzy commitment scheme. In: ACM Conf. on Computer and Communications Security.

Kevenaar, T., Schrijen, G., Veen, M., Akkermans, A., Zuo, F., 2005. Face recognition with renewable and privacy preserving binary templates. In: Auto ID 2005, Fourth IEEE Workshop on Automatic Identification Advanced Technologies.

Kuan, Y., Goh, A., Ngo, D., Teoh, A., 2005. Cryptogrpahic keys from dynamic hand-signatures with biometric secrecy preservation and replaceability. In: Auto ID 2005, Fourth IEEE Workshop on Automatic Identification Advanced Technologies.

Linnartz, J., Tuyls, P., 2003. New shielding functions to enhance privacy and prevent misuse of biometric templates. In: Proc. of the 4th Internat. Conference on Audio and Video-based Biometric Person Authentication, Guildford, UK.

Peterson, W.W., Weldon, E., 1972. Error-Correcting Codes, second ed. MIT Press, Cambridge, USA.

Ratha, N.K., Connell, J.H., Bolle, R., 2001. Enhancing security and privacy in biometrics-based authentication system. IBM Systems J. 40 (3), 614–634.

Ratha, N.K., Chikkerur, S., Connell, J.H., Bolle, R.M., 2007. Generating cancelable fingerprint templates. IEEE Trans. Pattern Anal. Machine Intell. 29 (4), 561–572.

Schneier, B., 1996. Applied Cryptography. John Wiley, New York.

Second International Fingerprint Verification Competition. <http://bias.csr.unibo.it/fvc2002/>.

Soutar, C., Roberge, D., Stoianov, A., Gilroy, R., Kumar, V., 1999. Biometric encryption. In: Nichols, R. (Ed.), ICSA Guide to Cryptography. McGraw-Hill.

Teoh, A., Ngo, D., Goh, A., 2004. Biohashing: Two factor authentication featuring fingerprint data and tokenised random number. Pattern Recognition 37 (11), 2245–2255.

Tulyakov, S., Farooq, F., Govindaraju, V., 2005. Symmetric hash functions for fingerprint minutiae. In: Internat. Workshop on Pattern Recognition for Crime Prevention, Security and Surveillance, Bath, UK.

Tuyls, P., Akkermans, A.H.M., Kevenaar, T.A.M., Schrijen, G.J., Bazen, A.M., Veldhuis, R.N.J., 2005. Practical biometric authentication with template protection. In: Proc. 5th Internat. Conf. on Audio and Video-based Biometric Person Authentication, Rye Town, NY.

Uludag, U., Jain, A., 2004. Attacks on biometric systems: A case study in fingerprints. In: SPIE-EI 2004, Security, Steganography and Watermarking of Multimedia Contents VI.

Uludag, U., Jain, A., 2006. Securing fingerprint template: Fuzzy vault with helper data. In: Proc. IEEE Workshop on Privacy Research In Vision, New York.

Uludag, U., Pankanti, S., Prabhakar, S., Jain, A.K., 2004. Biometric cryptosystems: Issues and challenges. Proc. IEEE 92 (6), 948–960.

Uludag, U., Pankanthi, S., Jain, A., 2005. Fuzzy vault for fingerprints. In: Proc. 5th Internat. Conf. on Audio and Video-based Biometric Person Authentication, Rye Town, NY.