

# Symmetric Hash Functions for Secure Fingerprint Biometric Systems

Sergey Tulyakov

Faisal Farooq \*

Praveer Mansukhani

Venu Govindaraju

*CUBS, SUNY at Buffalo, Amherst, NY 14228*

---

## Abstract

Securing biometrics databases from being compromised is one of the most important challenges that must be overcome in order to demonstrate the viability of biometrics based authentication. In this paper we present a novel method of hashing fingerprint minutia and performing fingerprint identification 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). In fact, most approaches of hashing minutia and developing a cancellable system described in the literature assume the location of the singular points. Others assume a pre-alignment between the test and the stored fingerprint templates. These assumptions 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 about 3%. We also present the performance analysis of a hybrid system that has an *EER* of about 2% which is very close to the performance of plain matching in the minutia space.

## 1 Introduction

Securing a biometric template is a critical step in the successful implementation of biometric 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. Password based authentication systems often come under attacks (e.g., man-in-the-middle) during transfer over a network or database hijack wherein the whole password database can be compromised [1]. To prevent such attacks, plaintext 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). Thus,  $h = H(m)$ . Hash function  $H$  is said to be a one-way function if it is hard to invert, that is, given a hash value  $h$ , it is computationally infeasible to find some input  $x$  such that  $H(x) = h$  [1].

Biometric based authentication systems face challenges similar to password systems. We have developed a method for biometric data which is similar to password encryption and hashing. Biometric identification is performed using hashed biometric data instead of the original template. Figure 1 illustrates the system for fingerprint biometrics. Fingerprints are obtained by a online

---

\* Corresponding author

*Email address:* `ffarooq2@cubs.buffalo.edu` (Faisal Farooq).

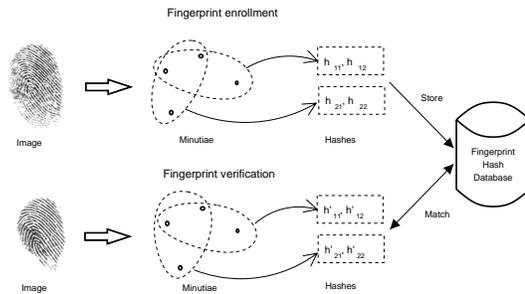


Fig. 1. Securing fingerprint information

scanner, the minutia locations are found and hashes of minutia subsets are constructed. The operations of finding minutiae and hashes can potentially be incorporated into scanner. Only the hashes then 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 or on the server. In this paper we extend our earlier work [2] by introducing additional methods of securing and personalizing the hash for fingerprint data.

## 2 Challenges

The hash value for text passwords completely changes even if a single character in a password is changed. Hashing is still feasible in case of passwords because the authentication is an *all-or-none* system and access is granted only if the entire password entered is correct. Also, in password protected systems, in case the password database is compromised, a new set of passwords can be generated or set up. Biometric systems though proven to be more secure and efficient than password protected systems, are probabilistic and not all-or-none like passwords. Authentication is based on scores that can vary anywhere between 0 – 100%. In case biometric data is hashed, even a slight change in the

acquisition of the biometric can lead to a totally different hash value which might not match the original within the same matching threshold as that of unhashed ones. Thus, the hash-based system should possess the following:

- similar fingerprints should have similar hash values
- different fingerprints should not have similar hashes
- rotation and translation of original template should not have a big impact on hash values
- possibly partial fingerprints should be matched

### 3 Previous Work

The situation we are facing here is analogous to a password based authentication system where we would like successful authentication even if the password provided is *almost same*. Is it possible to construct a person authentication algorithm if we allow the password to change slightly? Error correcting codes [3] have successfully been utilized in such situations of recovering changed data and their use might be appropriate here. Indeed, Davida et al.[4] presented an authentication algorithm based on error correcting codes. In this algorithm, error-correcting digits are generated from the biometric data and some other verifying data, and stored in the database. During authenticating stage, possibly changed biometric data is combined with stored error-correcting digits and error correction is performed. The amount of correction required serves as a measure of the authentication success. This algorithm was later modified as fuzzy commitment scheme in the work of Juels and Wattenberg[5] and some of its properties were derived. Kuan et al. [6] presented a method for extracting cryptographic keys from dynamic handwritten signatures. A similar approach

for face templates was presented by Kevenaar et al. [7] in which they generate binary feature vectors from biometric face data that can be protected by using helper data introduced into this bit sequence.

None of these approaches can directly be extended to fingerprints. Fingerprint data with minutia positions as features presents additional challenges for designing hashes. Minutia sets of two fingerprints usually do not coincide, it is nearly impossible to introduce some order in minutia set, and global transformation parameters are usually present between corresponding minutiae. Error correcting codes require that the original sequence be in some ordered fashion in order to locate and then try to correct the errors in the modified sequence. A fuzzy vault algorithm (Juels and Sudan [8]) improves upon fuzzy commitment scheme in trying to solve first two challenges and also uses error-correcting codes. The security of the algorithm relies on the addition of chaff points, or, in the case of fingerprint vault, false minutia points. The attacker would try to find a subset of points well intersecting with non-chaff point set. Thus more chaff points provides better security, but arguably worse vault unlocking performance. The application of fuzzy vault to fingerprint identification appeared in the work of Clancy et al.[9]. That paper showed realistic expectations on the numbers of chaff points and associated attack complexity. The algorithm used the assumption that fingerprints are aligned, and corresponding minutiae had similar coordinates. Uludag and Jain [10] proposed a fuzzy vault scheme by adding extra chaff points and securing the template by a standard 128-bit AES algorithm. It still requires pre-aligning the test and stored fingerprint and the achieved FAR on a test set of 100 fingerprint is  $\sim 20\%$ . Moreover, Tuyls et al [11,12] 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. [13] took another approach to secure fingerprint biometrics. The algorithm operates on images by constructing special filter in Fourier space encoding key data. The data can be retrieved only by presenting similar fingerprint image to the decoder. The matching procedure is correlation based, thus translations of images are possible but not rotations. More recently, Uludag and Jain[14] presented an advancement of the earlier algorithm with a genuine accept rate of  $\sim 72\%$ . However, the alignment is highly prone to error, and does not work on poor quality or partial images.

## 4 Motivation

The main difficulty in producing hash functions for fingerprint minutiae is the inability to somehow normalize fingerprint data, for example, by finding specific fingerprint orientation and center. If fingerprint data is not normalized, then the values of any hashing functions are destined to be orientation/position-dependent. The way to overcome this difficulty is to have hash functions as well as matching algorithm deal with transformations of fingerprint data.

### 4.1 *Minutiae based matching*

In fingerprint based biometric authentication systems, minutiae based matching has become a *de-facto* standard. A fingerprint is made of a series of ridges and furrows on the surface of the finger. The uniqueness of a fingerprint can be determined by the pattern of ridges and furrows as well as the minutiae points. Minutiae points are local ridge characteristics that occur at either a ridge bifurcation or a ridge ending. Correlation based techniques have proven

to be inefficient and at times infeasible being highly sensitive to translation and rotation. The task of fingerprint matching requires that the two prints be aligned in the best possible alignment. After alignment, the number of matching minutiae points determine how good the match is. In our work we use ideas similar to [15] and [16] to combine results of localized matchings into the whole fingerprint recognition algorithm. Localized matching consists of matching minutia triplets using such features as angles and lengths between minutia points. For each minutia feature vector of length 3  $(x,y,\theta)$  and its two nearest neighbours, a secondary feature vector of length 5 is generated which is based on the Euclidean distances and orientation difference between the central minutia and its nearest neighbours. Matching is performed on these secondary features. In contrast, for localized matchings in this work we keep only limited information about matched neighborhoods, so that minutia positions cannot be restored. Global matching is essentially finding a cluster of localized matchings with similar rotation( $r$ ) and transformation( $t$ ) parameters. It seems that proposed algorithm of Uludag and Jain[?] might also use this 2-stage technique. Unlike fingerprint vault algorithm[9] our algorithm performs hashing of not only enrolled fingerprint, but of test fingerprint also. Thus hashing can be incorporated into scanner, and original fingerprint data will never be transmitted nor stored in the database.

#### *4.2 Symmetric Hash Functions*

As described earlier, a small change in the input to a hash function changes the hash value considerably. This change could be information missing from the original input, added noise to the input or a change in the order of the

input. 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  $X = x_1x_2x_3 \dots x_n$ . We can have two hash functions

$$H(X) = k_1x_1 + k_2x_2 \dots + k_nx_n, \quad k_1 \neq k_2 \dots \neq k_n \quad (1)$$

$$H_{sym}^m(X) = x_1^m + x_2^m \dots + x_n^m \quad (2)$$

As we observe, if the order of the input is changed to  $X = x_2x_3x_n \dots x_1$ , 1 yields a different hash value where as 2 remains unchanged. We can generate similar hash functions that are symmetric. Moreover, arbitrary combinations of more than one hash function yield other hash functions. Thus, we can have a whole family of symmetric hash functions by combining together elementary symmetric functions of 2:  $H_{sym,f}(X)' = f(H_{sym}^1(X), \dots, H_{sym}^n(X))$ . This property of the symmetric hash functions can be exploited to our purpose in 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 orientation of finger on scanner. Thus the transformation of one fingerprint to the other can be described by the complex function  $f(z) = rz + t$  (Figure 2). In our approach we construct hash functions and corresponding matching algorithm, so that this

transformation function is taken into account. Additionally we cannot set specific order of minutiae, so we want our hash functions be independent of this order. Thus we consider symmetric complex functions as our hash functions.

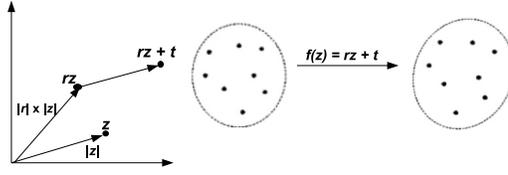


Fig. 2. Transformation of minutiae as represented in the complex plane.

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

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

If the number of hash functions  $m$  is less than the number of minutia points  $n$  participating in the construction of hash function, it is not possible to restore original minutia positions given hash values.

Suppose that the another image of the fingerprint is obtained through above described transformation  $f(z) = rz + t$ , thus locations of corresponding minutia points are  $c'_i = f(c_i) = rc_i + t$ . Hash functions of the transformed minutiae can be rewritten as

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

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

$$\begin{aligned}
h'_1 &= rh_1 + nt \\
h'_2 &= r^2h_2 + 2rth_1 + nt^2 \\
h'_3 &= r^3h_3 + 3r^2th_2 + 3rt^2h_1 + nt^3 \\
&\dots
\end{aligned} \tag{5}$$

Equations 5 have two unknown variables  $r$  and  $t$ . If we take into account errors introduced during fingerprint scanning and minutia search, the relation between hash values of enrolled fingerprint  $\{h_1, \dots, h_m\}$  and hash values of

test fingerprint  $\{h'_1, \dots, h'_m\}$  can be represented as

$$h'_i = f_i(r, t, h_1, \dots, h_n) + \epsilon_i \quad (6)$$

The matching between hash values of enrolled fingerprint  $\{h_1, \dots, h_m\}$  and hash values of test fingerprint  $\{h'_1, \dots, h'_m\}$  consists in finding  $r$  and  $t$  that minimize errors  $\epsilon_i$ . During algorithm implementation we considered minimization of error functions  $\epsilon = \sum \alpha_i |\epsilon_i|$ , where weights  $\alpha_i$  were chosen empirically.

## 6 Global Fingerprint Matching Using Hash Functions

It turns out that trying to use hash functions with respect to the minutia set of whole fingerprint is impractical. Even the small difference in minutia sets of two prints of the same finger will produce significant difference in hash values. Additionally, the higher order hash values tend to change greatly with the small change in positions of minutia points.

To overcome these difficulties we considered using hash functions for matching localized sets of minutia, and global matching of two fingerprints as a collection of localized matchings with similar transformation parameters  $r$  and  $t$ . As in base fingerprint matcher[16] the localized set is determined by a particular minutia and few of its neighbors. The hashes are calculated for each localized set. Total hash data extracted from fingerprint is a set of hashes  $\{h_{i,1}, \dots, h_{i,m}\}$ ,  $i = 1, \dots, k$ , where  $k$  is the total number of localized minutia sets.

During matching of two hash sets we first perform a match of all localized sets in one fingerprint to all localized sets in another fingerprint. The matches

with highest confidences are retained. Then, assuming in turn that a particular match is a correct match, we find how many other matches have similar transformation parameters. The match score is composed from the number of close matches and confidences of those matches.

## 7 Experimental Analysis

### 7.1 Dataset

We tested our system on *FVC2002*'s DB1 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 various image quality. We followed the protocols of *FVC2002* to evaluate the FAR(False Accept Rate) and FRR(False Reject Rate). For FRR the total number of genuine tests is  $\frac{(8*7)}{2} * 100 = 2800$ . For FAR, the total number of impostor tests is  $\frac{(100*99)}{2} = 4950$ .

### 7.2 Experimental Setup

We carried out experiments with different configurations, using different number of minutia points( $n$ ) and hashing functions( $m$ ). We tried out the configurations as follows

- (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 find three nearest neighbors, and for each minutia triplet including original minutia point construct two hash functions using the formula  $h_m(c_1, c_2, \dots, c_n) = c_1^m + c_2^m + \dots + c_n^m$  where  $m = 1, 2$ .

We use similar formulae for directions.

Configuration 3 for the experimental setup can be explained as follows: 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 for all minutia triplets are now used for hashing, performing the alignment between the template and the test fingerprint and also to calculate the matching scores. Thus, if a fingerprint is represented in the minutia space by a set of minutia points  $\{m_1, m_2, \dots, m_n\}$ , this operation maps it into a new space where it is now represented as a set of triangle centers  $\{T_1, T_2, \dots, T_k\}$ . The task of reversing this hash function would be to find out the actual minutia point locations given these triangle centers. We compared performance with fingerprint matching algorithm developed in [16] and using same set of fingerprints with identically extracted minutiae points. Also, since in configurations 1 and 2 we simply get another set of minutia points, we used matching algorithm of [16] to perform matching.

Currently achieved equal error rate (point where  $FAR = FRR$ ) of proposed algorithm is  $\sim 3\%$ . The equal error rate ( $EER$ ) for plain matching is  $\sim 1.7\%$ . The ROC characteristics of the baseline system and the different configurations of our system are shown in figure 3.

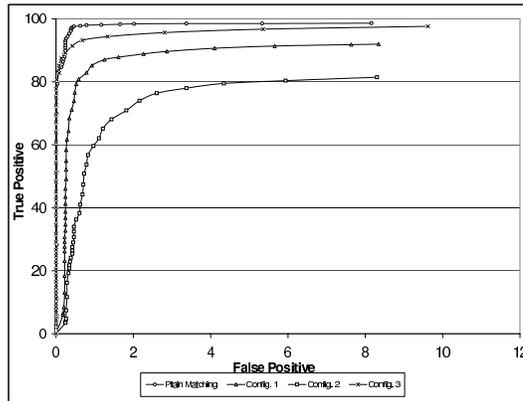


Fig. 3. ROC Curves for the baseline system[16] and the different experimental configurations.

As noted the accuracy of the secure system is slightly lesser than the baseline system. Nevertheless, the benefits of securing fingerprint data can easily outweigh the performance loss in many applications. Performance loss would mean more strict decisions on matching, and more frequent repeat matching attempts. Arguably many people will trade off the assurance on their fingerprint template privacy for the inconvenience of performing repeat fingerprint scan.

## 8 Security of Proposed Algorithm

The main purpose of the proposed algorithm is to conceal original fingerprint and minutiae locations from an attacker. Is it possible to reconstruct minutia positions given stored hash values? Since the number of hash values for each local minutia set is less than number of these minutiae, it is not possible to get locations using only information of one local set. On the other hand, it seems possible to construct a big system of equations involving all hashes (hashes of only first order might be considered for linearity). The biggest problem in constructing such system is that it is not known which minutia participated in the creation of particular hash value.

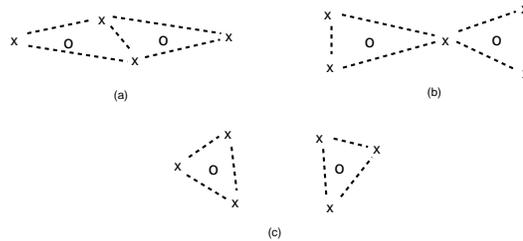


Fig. 4. Different number of minutiae(crosses) can participate in the creation of two triplet centers(circles).

The problem is illustrated in figure 4. Two triplet centers are formed from 4, 5 and 6 minutia points. Thus during constructing an equation system for finding minutia positions, we have a problem of deciding how many minutiae should be, in addition to matching minutia to triplet centers.

Hill-climbing type attacks[17] will probably have more difficult time to make a match since varying minutia position might have effect on few triplets, thus influencing matching score in a more complex way. Also, we think, that even if attack succeeded and match is found, the resulting minutiae locations will

be different from original. In this situation, change of hashing algorithm will make reconstructed fingerprint unmatchable. Brute force search on all nearest neighbors of a triangle center for the above method could be computationally feasible, however, using higher order hashes instead of simply neighboring minutiae can render these attacks ineffective. If the minutia positions are floating point and not integers, then brute force is 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 as compared 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 to use additional information (keys, personal hashes etc.) that actually harden the fingerprint hash.

## 9 Cancellable Biometric

The proposed hashing of fingerprint templates eliminates the possibility of an intruder learning original minutia positions. Though we consider it as an extremely difficult task, an intruder might construct an artificial template producing similar hash values, but having different minutia positions. Thus we need to expand our algorithm to make fingerprint hashes cancelable. This can be achieved by reenrolling persons using 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 in a card or by entering on a keypad. Also, this key can be reissued in

case of a potential compromise. 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 may be based on a token that the user carries or a password that the user remembers. It may even be based on another biometric, thus making the key personal. To achieve a cancellable biometric algorithm we need to provide a way to automatically construct and use randomly generated hash functions. Presented set of hash functions is an algebraic basis in the set of polynomial symmetric functions. Thus, we were able to express hash functions of transformed minutia set through original set of symmetric functions. This is a clue to constructing other similar hash functions. Essentially we can take arbitrary algebraic basis of symmetric polynomials of degree less than or equal to  $m$ ,  $\{s_1, \dots, s_m\}$  as our hash functions. Then the hash functions of the transformed minutiae,  $s_i(rc_1+t, \dots, rc_n+t)$ , will still be symmetric functions of the same degree with respect to variables  $c_1, \dots, c_n$ . Thus, hashes of transformed minutia could be expressed using original hashes,  $s'_i = s_i(rc_1+t, \dots, rc_n+t) = F_i(r, t, s_1, \dots, s_m)$  for some polynomial functions  $F_i$ . These equations will allow matching localized minutia sets, and finding corresponding transformation parameters.

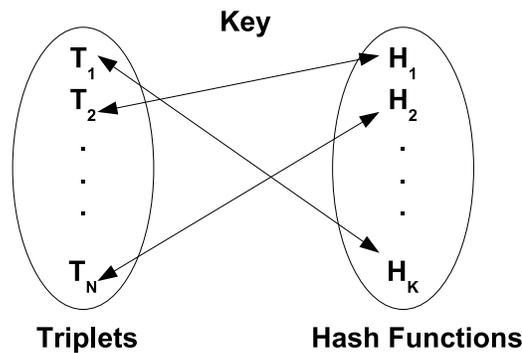


Fig. 5. Associating the minutiae triplets with hash functions.

## 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

$$h_1(c_1, c_2, c_3) = (c_1 + c_2 + c_3)$$

$$h_2(c_1, c_2, c_3) = (c_1c_2 + c_2c_3 + c_1c_3)$$

$$h_3(c_1, c_2, c_3) = c_1c_2c_3$$

$$h_4(c_1, c_2, c_3) = (c_1 - c_2)^2 + (c_2 - c_3)^2 + (c_1 - c_3)^2 \text{ etc.}$$

Any linear combination of these functions will also yield a symmetric hash function. Thus for any triplet, we have several functions  $h_1, h_2 \dots h_k$  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 multiplied if we could choose several of these hash function simultaneously and in some random order. Thus for each triplet  $T_1, T_2 \dots T_N$  we associate a corresponding hash function  $H_1, H_2 \dots H_N$ . 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. 5.

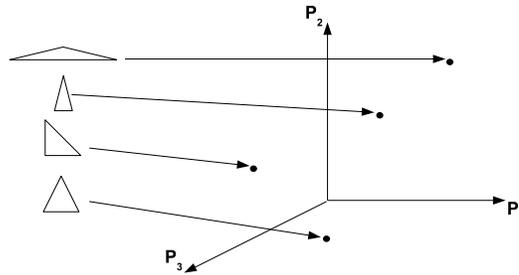


Fig. 6. Triangles as points in the parameter space.

However, in order to successfully verify the individual at a later instance, 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 each triangle or triplet  $T$  can be represented parametrically by specifying three parameters such as - two sides and one angle, or one sides 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 this parametric space as in Fig. 6.

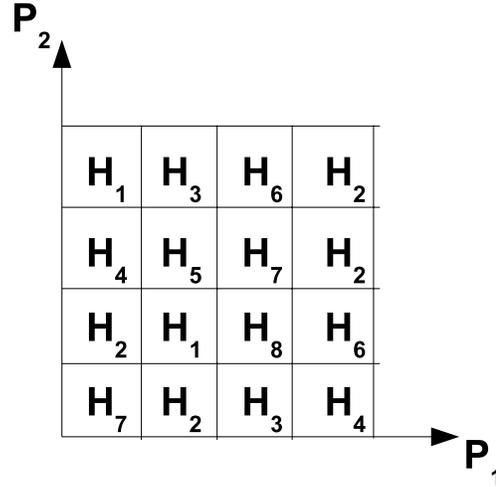


Fig. 7. Associating the hash functions with cells in the parameter space.

All triangles with similar geometries will 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. 6. Further we divide the parameter space into non-overlapping cells as in Fig. 7 (the cells are shown in  $2D$  for simplicity). To each cell we assign a specific hash function. The association between the hash function and the cell are now contained in the secret key. Assume two instances of the key are  $H_2H_4H_8H_1H_3H_1$  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 currently assume that there are  $c$  such cells in all. This arrangement solves the original problem of triplet association. If a triplet  $T$

	Plain	Secure	Hybrid
Avg. points matched	25.90	57.50	24.55
<i>EER%</i>	1.7	3.0	2.0

Table 1

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 them immune to brute force attack.

## 9.2 Personalizing and Reissuing

While the number of symmetric functions possible for each triplet is clearly infinite, it is not clear at this point of time as to how many symmetric functions can be chosen such that the transformation is still meaningful, but it can be assumed to be 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  $N \times N \times N \dots (c \text{ times}) = N^c$ . Thus by introducing the secret key  $K$ , we are exponentially multiplying the total possibilities of hash functions and increasing the computational com-

plexity of a brute force attack by the same amount. This key can be based on a biometric such as face or iris or its convolution by some signal. In case of compromise of the database the keys can be reissued and different set of hash functions chosen as shown earlier, thus rendering the biometric system cancellable.

## 10 Performance Analysis

The loss in the accuracy of the secure system as compared to the plain version could be attributed to various factors such as reduction in the number of points being matched. It should be noted, however, that 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 described in figure 3. Thus the total size of stored hash values can be even bigger than the size of original fingerprint template. The decrease in the accuracy might be caused by the loss in information when keeping reduced number of variables based on minutia triplets. For every three neighboring minutia points we have reduced the number of variables to 4 (2 complex numbers) instead of original 6. For example, the average number of minutia matched for a genuine match in the baseline version was observed to be  $\sim 25.9$ . In the secure version the average number of triplet centers matched for genuine tests were  $\sim 57.5$ . There can be additional reasons for observed performance hit, such as difficulty in matching localized hashed values

In order to evaluate the performance of the secure matching algorithm vis-a-vis the plain matching, we carried out experiments where the transformation parameters were acquired from our algorithm. These  $r$  and  $t$  parameters were

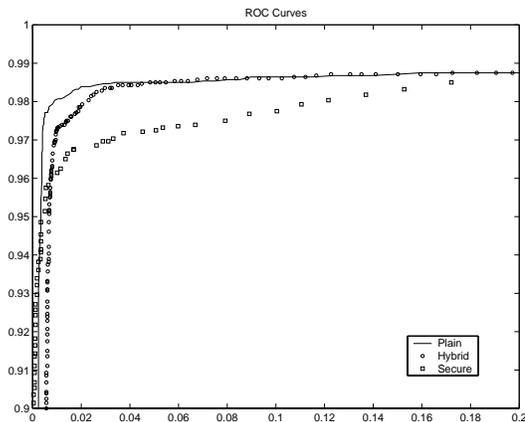


Fig. 8. Comparing the ROC curves of the plain, secure and hybrid systems.

then used as the transformation parameters for the plain version. For this setup, an EER of  $\sim 2.0\%$  was achieved. This suggests that the scoring formulae for the secure version requires improvement based on the current techniques. Figure 8 shows the three curves. As we see the hybrid system performs better than our baseline secure system, however, a slightly worse than the baseline plain system. Table 1 gives a comparison between the three systems. The comparable number of minutia matched in the plain version and the hybrid system suggest that indeed the secure system performs as good in terms of finding the tranformation parameters and matching the minutia. This suggests that the performance decrease is in the scoring methodology.

## 11 Conclusions

In this paper 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 information is unordered as in the case of minutia on fingerprints.

We have successfully implemented a secure authentication system with performance comparable to plain matching systems. We have also presented methods to cancel and reissue the biometric and to personalize the hash values based on keys that could potentially be derived from other biometric traits.

## References

- [1] B. Schneier, *Applied Cryptography*, John Wiley, New York, 1996.
- [2] S. Tulyakov, F. Farooq, V. Govindaraju, Symmetric hash functions for fingerprint minutiae, in: *International Workshop on Pattern Recognition for Crime Prevention, Security and Surveillance*, Bath, UK, 2005, pp. 30–38.
- [3] W. Peterson, E. Weldon, *Error-Correcting Codes*, 2nd Edition, MIT Press, Cambridge, USA, 1972.
- [4] G. Davida, Y. Frankel, B. Matt, On enabling secure applications through on-line biometric identification, in: *Proc. of the IEEE 1998 Symp. on Security and Privacy*, Oakland, Ca., 1998.
- [5] A. Juels, M. Wattenberg, A fuzzy commitment scheme, in: *ACM Conference on Computer and Communications Security*, 1999, pp. 28–36.
- [6] Y. Kuan, A. Goh, D. Ngo, A. Teoh, Cryptographic keys from dynamic hand-signatures with biometric secrecy preservation and replaceability, in: *Auto ID 2005, Fourth IEEE Workshop on Automatic Identification Advanced Technologies*, 2005, pp. 27–32.
- [7] T. Kevenaar, G. Schrijen, M. Veen, A. Akkermans, F. Zuo, Face recognition with renewable and privacy preserving binary templates, in: *Auto ID 2005, Fourth IEEE Workshop on Automatic Identification Advanced Technologies*, 2005, pp. 21–26.

- [8] A. Juels, M. Sudan, A fuzzy vault scheme, in: IEEE International Symposium on Information Theory, 2002.
- [9] T. Clancy, D. Lin, N. Kiyavash, Secure smartcard-based fingerprint authentication, in: ACM Workshop on Biometric Methods and Applications (WBMA 2003), 2003.
- [10] U. Uludag, S. Pankanthi, A. Jain, Fuzzy vault for fingerprints, in: Proc. of the 5th International Conference on Audio and Video-based Biometric Person Authentication, Rye Town, NY, 2005, pp. 310–319.
- [11] J. Linnartz, P. Tuyls, New shielding functions to enhance privacy and prevent misuse of biometric templates, in: Proc. of the 4th International Conference on Audio and Video-based Biometric Person Authentication, Guildford, UK, 2003, pp. 393–402.
- [12] P. Tuyls, A. H. M. Akkermans, T. A. M. Kevenaar, G. J. Schrijen, A. M. Bazen, R. N. J. Veldhuis, Practical biometric authentication with template protection, in: Proc. of the 5th International Conference on Audio and Video-based Biometric Person Authentication, Rye Town, NY, 2005, pp. 436–446.
- [13] C. Soutar, D. Roberge, A. Stoianov, R. Gilroy, V. Kumar, Biometric encryption, in: R. Nichols (Ed.), ICSA Guide to Cryptography, McGraw-Hill, 1999.
- [14] U. Uludag, A. Jain, Securing fingerprint template: Fuzzy vault with helper data, in: Proc. IEEE Workshop on Privacy Research In Vision, New York, 2006.
- [15] R. Germain, A. Califano, S. Colville, Fingerprint matching using transformation parameter clustering, IEEE Computational Science and Engineering 4 (4) (1997) 42–49.
- [16] T.-Y. Jea, V. S. Chavan, V. Govindaraju, J. K. Schneider, Security and matching of partial fingerprint recognition systems, in: SPIE Defense and Security Symposium, 2004.

- [17] U. Uludag, A. Jain, Attacks on biometric systems: a case study in fingerprints, in: SPIE-EI 2004, Security, Seganography and Watermarking of Multimedia Contents VI, 2004.