Symmetric Hash Functions for Fingerprint Minutiae

Sergey Tulyakov, Faisal Farooq and Venu Govindaraju

SUNY at Buffalo, Buffalo NY 14228, USA

Abstract. The possibility that a database with biometric data is compromised is one of the main concerns in implementing biometric identification systems. In this paper we present a method of hashing fingerprint minutia information and performing fingerprint identification in a new space. Only hashed data is transmitted and stored in the server database, and it is not possible to restore fingerprint minutia locations using hashed data. We also present a performance analysis of the proposed algorithm.

1 Introduction

The problem we are dealing with is well described in section 9.7 of Handbook of Fingerprint recognition[1]. Plaintext passwords can be hashed, and only hash values are stored in the database and transmitted across networks. Password authentication requires comparison of the hashed values and not original passwords. If database with hash values is ever compromised, all persons can be reenrolled using different passwords or different hash function.

Situation is different when using biometric data for person authentication. Due to the difficulty of devising hashing functions for biometric data biometric templates are usually stored unprotected in a central database. Even if stored templates are encrypted, matching is still performed using decrypted templates, and decryption process can be compromised as well. If the biometric database is compromised and an intruder obtains a person's biometric template, using this biometric will be impossible for the rest of person's life.

In this work we want to devise a method for biometric data, in particular fingerprint data, to be hashed, and the biometric identification to be performed using hashed biometric data. Hashing functions are one-way functions, and given the hash values it is impossible to reconstruct original template. Only the hash values are transmitted over the network and stored in the biometric database. In case the hash values are compromised, person will be re-enrolled using new hash function. The original biometric(e.g. fingerprint) is safe and never compromised.

Figure 1 presents a diagram of a system using proposed hashing algorithm. Fingerprints are obtained by the scanner, minutia locations are found and hashes of minutia subsets are constructed. Finding minutiae and hashes can be incorporated into scanner. Only hashes are transmitted and stored in the database. During verification stage new hash values are produced by the scanner and are

Fingerprint enrollment



Fig. 1. Application of the proposed hashing algorithm.

matched with those stored in the database. Matching can be performed either on the client or on the server.

2 Related Work

The hashing functions for text passwords usually completely change hash values even if a single character in a password is changed. Is it possible to construct a person authentication algorithm if we allow the password to change slightly? Error correcting codes [2] are designed to deal with such situations of recovering changed data and their use might be appropriate here. Indeed, Davida et al.[3] 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[4] and some of its properties were derived.

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. A fuzzy vault algorithm (Juels and Sudan [5]) 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.[6]. 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.

To address the frequent impossibility to properly align fingerprint images, Uludag and Jain [7] proposed to use features independent of global rotation and translation. It is still unclear if their approach will work.

Soutar et al. [8] 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.

In our work we use ideas similar to [9] to combine results of localized matchings into the whole fingerprint recognition algorithm. In that work localized matching consists in matching minutia triplets using such features as angles and lengths between minutia points. For each minutia feature vector of length 3 (x,y,θ) and its two nearest neighbors, 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 neighbors. 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 can not 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[7] might also use this 2-stage technique. Unlike fingerprint vault algorithm[6] 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.

3 Hash Functions of Minutia Points

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.

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 ways to put the finger on scanner. Thus the transformation of one fingerprint to the other can be described by the complex function f(z) = rz + t. 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.

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

$$h_1(c_1, c_2, \dots, c_n) = c_1 + c_2 + \dots + c_n$$

$$h_2(c_1, c_2, \dots, c_n) = c_1^2 + c_2^2 + \dots + c_n^2$$

$$\dots$$

$$h_m(c_1, c_2, \dots, c_n) = c_1^m + c_2^m + \dots + c_n^m$$
(1)

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

$$h_{1}(c'_{1}, c'_{2}, \dots, c'_{n}) = c'_{1} + c'_{2} + \dots + c'_{n}$$

$$= (rc_{1} + t) + (rc_{2} + t) + \dots + (rc_{n} + t)$$

$$= r(c_{1} + c_{2} + \dots + c_{n}) + nt = rh_{1}(c_{1}, c_{2}, \dots, c_{n}) + nt$$

$$h_{2}(c'_{1}, c'_{2}, \dots, c'_{n}) = c'^{2}_{1} + c'^{2}_{2} + \dots + c'^{2}_{n}$$

$$= (rc_{1} + t)^{2} + (rc_{2} + t)^{2} + \dots + (rc_{n} + t)^{2}$$

$$= r^{2}(c^{2}_{1} + c^{2}_{2} + \dots + c^{2}_{n}) + 2rt(c_{1} + c_{2} + \dots + c_{n}) + nt^{2}$$

$$= r^{2}h_{2}(c_{1}, c_{2}, \dots, c_{n}) + 2rh_{1}(c_{1}, c_{2}, \dots, c_{n}) + nt^{2}$$
(2)

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 hash values of corresponding minutia set of another fingerprint as $h'_i = h_i(c'_1, c'_2, \ldots, c'_n)$. Equations 2 now become

$$h'_{1} = rh_{1} + nt$$

$$h'_{2} = r^{2}h_{2} + 2rth_{1} + nt^{2}$$

$$h'_{3} = r^{3}h_{3} + 3r^{2}th_{2} + 3rt^{2}h_{1} + nt^{3}$$
...
(3)

Equations 3 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 $\{h_1, \ldots, h_m\}$ and hash values of test fingerprint $\{h'_1, \ldots, h'_m\}$ can be represented as

$$h_i = f_i(r, t, h_1, \dots, h_n) + \epsilon_i \tag{4}$$

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

4 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[9] 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 $\{h_{i,1}, \ldots, h_{i,m}\}$, $i = 1, \ldots, k$, where k is the total number of localized minutia sets.

During matching of two hash sets we first perform a match of all 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.

5 Experiments

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 $h_m(c_1, c_2, \ldots, c_n) = c_1^m + c_2^m + \cdots + c_n^m$ where m = 1, 2.

We use similar formulae for directions.

We compared performance with fingerprint matching algorithm developed in [9] 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 [9] to perform matching. 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 $\frac{(100*99)}{2} = 4950$. Currently achieved equal error rate(EER) of proposed algorithm is $\sim 3\%$. The 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 2.



Fig. 2. ROC Curves for the baseline system[9] and the different experimental configurations.

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. It should be also noted 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. There can be additional reasons for observed performance hit, such as difficulty in matching localized hashed values and reduced number of matched localized neighborhoods. Determining exact cause of performance loss and correcting it is one of the future research topics.

Nevertheless, the benefits of securing fingerprint data can easily outweigh the performance loss in many applications. Performance loss would mean stricter 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.

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

The problem is illustrated in figure 3. 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[10] 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.

7 Future Work

In this paper we presented one method of constructing hash functions. 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 $m, \{s_1, \ldots, s_m\}$ 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 variables c_1, \ldots, c_n . Thus, hashes of transformed minutia could be expressed using 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.

In presented algorithm global matching relies heavily on first order hash functions, basically centers of minutia triplets. If we want to use arbitrary symmetric hash functions, then the global matching algorithm should be modified.

The ROC curves in figure 2 suggest that the algorithm has slightly lesser accuracy than the baseline system which could be attributed to the fact that by considering centers of minutia triplets as the features to match, we might lose some information that the original minutia possess. Currently we are working on improving the accuracy of the system by possibly learning the parameters automatically and also trying to possibly use different scoring techniques.

Additional possible area of research is the use of scalar functions. For example, it is easy to construct minutia triplet features which are rotation and translation invariant. But, since algorithm requires estimation of rotation and translation, these features will not suffice.

References

- Maltoni, D., Maio, D., Jain, A.K., Prabhakar, S.: Handbook of Fingerprint Recognition. Springer, New York, (2003)
- Peterson, W., Weldon, E.: Error-Correcting Codes. 2nd edn. MIT Press, Cambridge, USA (1972)

- Davida, G., Frankel, Y., Matt, B.: On enabling secure applications through online biometric identification. In: Proc. of the IEEE 1998 Symp. on Security and Privacy, Oakland, Ca. (1998)
- 4. Juels, A., Wattenberg, M.: A fuzzy commitment scheme. In: ACM Conference on Computer and Communications Security. (1999) 28–36
- 5. Juels, A., Sudan, M.: A fuzzy vault scheme. In: IEEE International Symposium on Information Theory. (2002)
- Clancy, T., Lin, D., Kiyavash, N.: Secure smartcard-based fingerprint authentication. In: ACM Workshop on Biometric Methods and Applications (WBMA 2003). (2003)
- 7. Uludag, U., Jain, A.: Fuzzy fingerprint vault. In: Proc. Workshop: Biometrics: Challenges Arising from Theory to Practice. (2004) 13–16
- Soutar, C., Roberge, D., Stoianov, A., Gilroy, R., Kumar, V.: Biometric encryption. In Nichols, R., ed.: ICSA Guide to Cryptography. McGraw-Hill (1999)
- Jea, T.Y., Chavan, V.S., Govindaraju, V., Schneider, J.K.: Security and matching of partial fingerprint recognition systems. In: SPIE Defense and Security Symposium. (2004)
- Uludag, U., Jain, A.: Attacks on biometric systems: a case study in fingerprints. In: SPIE-EI 2004, Security, Seganography and Watermarking of Multimedia Contents VI. (2004)