Securing Fingerprint Template: Fuzzy Vault with Helper Data *

Umut Uludag
Michigan State University
uludagum@cse.msu.edu

Anil Jain
Michigan State University
jain@cse.msu.edu

Abstract

An important issue gaining attention in biometrics community is the security and privacy of biometric systems: How robust are these systems against attacks? What happens if the biometric template is lost or stolen? Can the privacy of the users be preserved even when a security breach occurs? Among the numerous attacks that can be launched against these systems, protecting the user template that is stored either locally (e.g., on a smart card) or centrally (e.g., on the server) is a major concern. As a possible solution to this problem, a new class of algorithms, termed biometric cryptosystems has been proposed. These systems do not store the original template but only a transformed version of the template within a cryptographic framework. An example of such systems is the fuzzy vault construct proposed by Juels and Sudan. In this construct, the biometric template is converted to a 2D point cloud, containing a secret such as a symmetric encryption key. The operation of the vault requires some "helper" data. In this paper, we present an implementation of the fuzzy fingerprint vault based on orientation field based helper data that is automatically extracted from the fingerprints. We further show that this helper data does not leak any information about fingerprint minutiae, hence complementing the increased user privacy afforded by the fuzzy fingerprint vault. We demonstrate the vault performance on a public domain fingerprint database.

1. Introduction

Security and privacy of biometric systems is becoming a major issue in biometrics community [12]: how robust are these systems against attacks? What happens if the biometric template is lost or stolen? Can the privacy of the users be preserved even when a security breach occurs? Among these issues, protecting the user template that is stored either locally or centrally is a major concern [16]. To increase the security of biometric systems, a new class of algorithms, termed biometric cryptosystems [20] has been proposed. These systems do not store the original template but only a transformed version of the template (with the aim of protecting them, hence, increased privacy) within a cryptographic framework.

An example of such a system is the fuzzy vault construct proposed by Juels and Sudan [9]. In their scheme, Alice can place a secret S (e.g., secret encryption key) in a vault and lock (secure) it using an unordered set A. Bob, using an unordered set B, can unlock the vault (access S) only if B substantially overlaps with A. The procedure for constructing the fuzzy vault is as follows: First, Alice selects a polynomial p of variable x that encodes S (e.g., by fixing the coefficients of p according to S). She computes the polynomial projections, $p(A)$, for the elements of A. She adds some randomly generated chaff points (to increase security) that do not lie on p, to arrive at the final point set R. When Bob tries to learn S (i.e., find p), he uses his own unordered set B. If B overlaps with A substantially, he will be able to locate many points in R that lie on p. Using error-correction coding (e.g., Reed-Solomon [10]), it is assumed that he can reconstruct p (and hence find S).

Another system for template protection is proposed by Linnartz and Tuyls [11]. They assume that a noise-free template X of a biometric identifier is available at the enrollment time and use this to encode a secret S to generate helper data W. They further assume that each dimension (of a multidimensional template) is quantized at q resolution levels. In each dimension, the process of obtaining W is equivalent to finding residuals that must be added to X to fit to an odd or even grid quantum depending upon whether the corresponding S bit is 0 or 1. The biometric query Y (a noisy version of X) is used to decrypt W to generate a message which is approximately the same as S. Their technique assumes that the biometric presentations (query and template) are aligned and that noise in each dimension is relatively small compared to the quantization.

Soutar et al. [18] proposed a key binding algorithm for an optical correlation-based fingerprint matching system. This algorithm binds a cryptographic key (typically 128 bits) with the user's fingerprint image at the time of en-

*This research was supported by ARO Grant number W911NF-05-1-0483.

rollment. The key is then retrieved only upon successful authentication using the so-called correlation filter functions. Davida et al. [6, 7] proposed an algorithm based on iris biometric. They consider binary representation of iris texture, called IrisCode [5] and used multiple IrisCodes to arrive at a canonical representation, which is combined with error correction data. Both these algorithms assume that the multiple acquisitions of the biometric (fingerprint and iris) are prealigned. Monrose et al. [14, 13] proposed a method to make passwords more secure by augmenting them with keystroke and voice biometric features. Their technique was inspired by password *salting*, where a user's password is salted by prepending it with a random number (the "salt"), resulting in a hardened password. The keystroke and voice templates were used for generating the associated salts, resulting in 60-bit cryptographic secrets. Dodis et al. [8] proposed theoretical foundations for generating keys from the *key material* that is not exactly reproducible (e.g., passphrases, biometric data that changes between enrollment and verification). Similar to the notion of helper data of Tuyls et al. [11], Dodis et al. [8] define *fuzzy extractors* (FE) to generate a variable R from the key material w, and public (helper) data P. Given the variable P, FE again generates R from w', if w' is "close" to w. For three distance metrics (Hamming distance, set difference and edit distance), Dodis et al. calculate the information revealed by P, and elaborate on the existence of possible algoritms for FE construction. They also propose a modification of the Juels and Sudan's fuzzy vault scheme [9]: instead of adding chaff points to the projections of the polynomial p, Dodis et al. [8] propose to use a polynomial p' (of degree higher than p) which overlaps with p only for the points from the genuine set A. This new polynomial p' replaces the final point set R of Juels and Sudan's scheme [9].

One of the main challenges in the above biometric cryptosystems is to align a query with the template that is available only in the transformed domain. Since we do not want to reveal any information about the stored template, the alignment must be done in the transformed domain. But the transformed version of the template does not carry sufficient information for alignment. Therefore, auxiliary (helper) data is needed to assist in alignment. A generic flowchart of the biometric cryptosystems is given in Fig. 1, where the secret S is transformed to a construct, $F(S, T)$, encompassing biometric template T. The secret is revealed using the helper data only when the query Q is sufficiently similar to T. Note that the helper data of the user which is potentially available in the public domain should not reveal any information that will enable an impostor to reconstruct the template. Hence, we are faced with two generally contradicting requirements, namely, (i) helper data should contain sufficient information to allow an alignment between query and template, and (ii) helper data should not leak critical information about the template. In this paper, we propose a scheme for generating helper data in the fuzzy fingerprint vault framework. The orientation-field based helper data proposed here does not leak minutiae information, yet it enables us to properly align the input query.

Previous work on such automatic alignment includes the work of Yang and Verbauwhede [21] who proposed to use *reference minutiae* that is extracted during vault encoding and decoding for alignment. If these two reference minutiae are the same, the origins of the coordinate frames used during locking and unlocking of the vault would be the same, and hence, the alignment could be established. Yang and Verbauwhede assumed this to be the case, but this assumption is not true in practice. Further, Chung et al. [2] proposed to use geometric hash tables (which code the transformation of genuine and chaff points with respect to a single reference minutia) for alignment. The authors did not provide any experimental results on alignment performance, further, the resulting hash tables can be prohibitively large.

2. Fuzzy Fingerprint Vault

The approach that forms the basis for our scheme is the fuzzy vault construct of Juels and Sudan [9]. Fig. 2 shows the block diagram of the proposed *fingerprint-based* fuzzy vault system. The system operates on the fingerprint minutiae characterized by either ending or bifurcation of the ridges. Minutiae are generally represented as (x, y, θ) triplets, denoting their row indices (x), column indices (y) and angle of the associated ridge, respectively (for a preliminary version of this research, please see [19]).

2.1. Encoding

The size of secret S that can be feasibly protected is limited by the capacity of the entity used for locking and unlocking the vault. Currently, we use the x and y coordinates of minutiae points for locking/unlocking the vault (similar to the algorithms of Clancy et al. [3], and Yang and Verbauwhede [21]). The encoding operation secures S with fingerprint minutiae data: if a query minutiae (that is aligned with respect to the template using the helper data as explained in Section 3) set that is similar to the template minutiae set is presented during decoding S can be reconstructed accurately. Note that the vault operation is decoupled from any backend application (e.g., encryption/decryption using S): vault is only responsible for securing S with fingerprint data. The fingerprint plays the role of a key. Note that this is not the key in traditional cryptosystems (e.g., AES) per se: rather, it has the role of a key for a new cryptographic construct, namely the fuzzy vault. In the current implementation, S is generated as a 128-bit random bit stream, like an AES symmetric encryption key.

Our current decoding implementation does not include

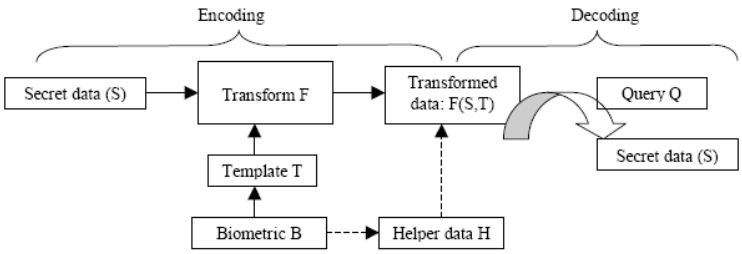
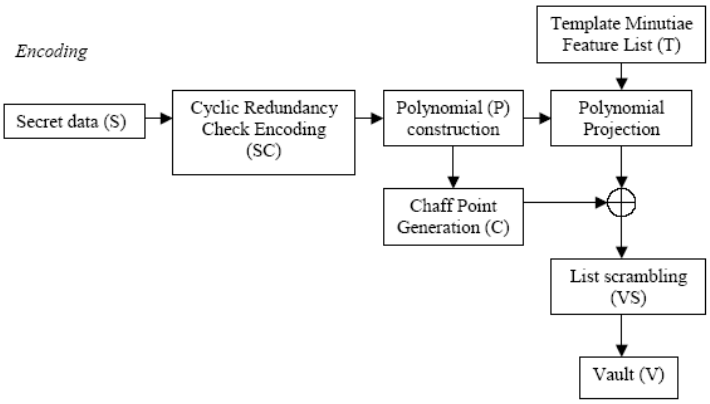
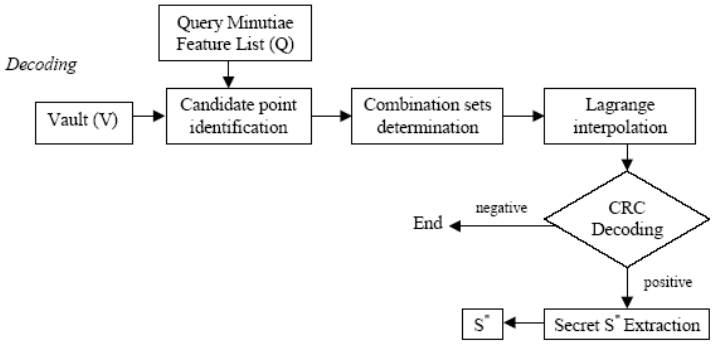


Figure 1. Using helper data in biometric cryptosystems.



(a)



(b)

Figure 2. Proposed fuzzy fingerprint vault: (a) vault encoding, (b) vault decoding.

any error-correction scheme, as proposed by Juels and Sudan [9], since realizing the necessary polynomial reconstruction via error-correction does not appear to be feasible. Instead, our algorithm decodes many candidate secrets and then identifies which one of these candidates is the actual secret. To enable this, we used Cyclic Redundancy Check (CRC). In our case, using incorrect minutiae points during decoding will cause an incorrect polynomial reconstruction, resulting in errors. In our implementation, we generate 16-bit CRC data from the secret S. The 16-bit primitive polynomial we use for CRC generation, $g_{CRC}(a) = a^{16} + a^{15} + a^2 + 1$, is called "CRC-16" used for EBCDIC messages by IBM [15].

Appending the 16 CRC bits to the original secret S (128-bits), we construct 144-bit data SC. From this point on, all operations take place in Galois field $GF(2^{16})$: we concatenate x and y coordinates of a minutia (8-bits each) as $[x|y]$ to arrive at the 16-bit locking/unlocking data unit u.

To account for slight variations in minutiae data (due to fingerprint distortions), raw minutiae data are first quantized. Namely, each minutia is translated to originate from a square tessellation of the 2D image plane. As an example, if the block size used in the tessellation is 11x11, any minutiae that has an x coordinate in the range [1, 11] is assumed to originate from the x coordinate 6. This allows for ± 5 pixel variations in the x and y coordinates of template and query minutiae. Note that this quantization scheme may occasionally put the same minutiae in the template and query prints into different bins due to quantization boundaries.

The bit string SC is used to find the coefficients of the polynomial p: 144-bit SC can be represented as a polynomial with 9 (144/16) coefficients, with degree $D = 8$:

$$p(u) = c_8 u^8 + c_7 u^7 + \ldots + c_1 u + c_0. \quad (1)$$

In other words, SC is divided into non-overlapping 16-bit segments, and each segment is declared as a specific coefficient, c_i, $i = 0, 1, 2, \ldots, 8$. Note that this mapping method (from SC to c_i) should be known during decoding, where the inverse operation takes place: decoded coefficients (c_i^*) are mapped back to decoded secret SC^*.

Two sets composed of point pairs are generated. The first one, called genuine set G, is found by evaluating $p(u)$ on the template minutiae (T). Assuming that we have N template minutiae (if we have more than N minutia, we choose the first N sorted according to ascending u values), $u_1, u_2, \ldots, u_N$, we construct

$$G = \left\{ \begin{array}{c} (u_1, p(u_1)) \\ (u_2, p(u_2)) \\ \vdots \\ (u_N, p(u_N)) \end{array} \right\} \quad (2)$$

Note that the template minutiae, $u_1, u_2, \ldots, u_N$, are selected to be unique, namely, $u_i \neq u_k$, if $i \neq k$, where $i = 1, 2, \ldots, N$, $k = 1, 2, \ldots, N$.

The second set, called the chaff set C, ensures the security of the system. To add M chaff points, we first generate M unique random points, $c_1, c_2, \ldots, c_M$ in the field $GF(2^{16})$, with the constraint that they do not overlap with $u_1, u_2, \ldots, u_N$. Then, we generate another set of M random points, $d_1, d_2, \ldots, d_M$, with the constraint that the pairs (c_j, d_j), $j = 1, 2, \ldots, M$ do not fall onto the polynomial $p(u)$. The chaff set C is defined as

$$C = \left\{ \begin{array}{c} (c_1, d_1) \\ (c_2, d_2) \\ \vdots \\ (c_M, d_M) \end{array} \right\} \quad (3)$$

The union of the genuine and chaff sets, $G \cup C$, is finally passed through a list scrambler that randomizes the list, with the aim of removing any stray information that can be used to separate chaff points from genuine points. This results in the vault set VS

$$VS = \left\{ \begin{array}{c} (v_1, w_1) \\ (v_2, w_2) \\ \vdots \\ (v_{N+M}, w_{N+M}) \end{array} \right\}. \quad (4)$$

2.2. Decoding

A user tries to unlock the vault V using N query minutiae $Q = \{u_1^*, u_2^*, \ldots, u_N^*\}$. The points to be used in polynomial reconstruction are found by comparing u_i^*, $i = 1, 2, \ldots, N$ with the abscissa values of the vault V, namely v_l, $l = 1, 2, \ldots, (N + M)$: if any u_i^*, $i = 1, 2, \ldots, N$ is equal to v_l, $l = 1, 2, \ldots, (N + M)$, the corresponding vault point (v_l, w_l) is added to the list of points to be used during decoding. Assume that this list has K points, where $K \leq N$. For decoding a degree D polynomial, $(D + 1)$ unique projections are necessary. We find all possible combinations of $(D + 1)$ points, among the list with size K, resulting in $\left(\begin{array}{c} K \\ D+1 \end{array} \right)$ combinations. For each of these combinations, we construct the Lagrange interpolating polynomial. For a specific combination set given as

$$L = \left\{ \begin{array}{c} (v_1, w_1) \\ (v_2, w_2) \\ \vdots \\ (v_{D+1}, w_{D+1}) \end{array} \right\}, \quad (5)$$

the corresponding polynomial is

$$p^*(u) = \frac{(u - v_2)(u - v_3)\ldots(u - v_{D+1})}{(v_1 - v_2)(v_1 - v_3)\ldots(v_1 - v_{D+1})} w_1 \ldots + \frac{(u - v_1)(u - v_2)\ldots(u - v_D)}{(v_{D+1} - v_1)(v_{D+1} - v_2)\ldots(v_{D+1} - v_D)} w_{D+1}. \quad (6)$$

This calculation is carried out in GF(2^{16}), and yields

$$p^*(u) = c_8^* u^8 + c_7^* u^7 + \ldots + c_1^* u + c_0^*. \quad (7)$$

The coefficients are mapped back to the decoded secret SC^*. For checking whether there are errors in this secret, we divide the polynomial corresponding to SC^* with the CRC primitive polynomial, $g_{CRC}(a) = a^{16} + a^{15} + a^2 + 1$. Due to the definition of CRC, if the remainder is not zero, we are certain that there are errors. If the remainder is zero, with very high probability, there are no errors. For the latter case, SC^* is segmented into two parts: the first 128-bits denote S^* while the remaining 16-bits are CRC data. Finally, the system outputs S^*. If the query minutiae Q overlaps with template minutiae T in at least $(D+1)$ points

for some combinations, the correct secret will be decoded, namely, $S^* = S$ will be obtained. This denotes the desired outcome when query and template fingerprints are from the same finger.

3. Constructing Helper Data

The Orientation Field Flow Curves (OFFC) are sets of piecewise linear segments that represent the underlying flow of fingerprint ridges [4]. They are robust to noise arising from minutiae, islands, smudges, and cuts. These curves are obtained as follows: firstly, the orientation field that shows the dominant orientation (o) in each 8x8 fingerprint block is found. A flow curve with a starting point s_0 is then found iteratively as

$$s_j = s_{j-1} + d_j * l_j * o_{s_{j-1}} \tag{8}$$

where j denotes the index of points on the curve, $d_j \in \{-1, 1\}$ is the flow direction between s_j and s_{j-1}, l_j is the length of line segment (e.g., 5 pixels) between these two points, and $o_{s_{j-1}}$ is the orientation value at location s_{j-1}. By tracing points in the direction of respective orientation values, the full flow curve is obtained. Repeating this procedure for multiple starting points s_0, the set of flow curves is generated. The maximum curvature points on these curves and the corresponding curvature values constitute our helper data. Fig. 3 shows the steps in generating this helper data (shown as blue points). Note that the helper data is constructed first for the template (during vault encoding), and then for the query (during vault decoding).

3.1. Curvature Estimation for OFFC

The set of points on the piece-wise linear OFF curves are used to find the maximum curvature point locations and the actual curvature values. Namely, given a curve, (i) the curvature value C is calculated for every point on the curve, (ii) point with the maximum curvature is identified, (iii) its curvature value and location are added to the corresponding candidate helper data CH (Fig. 3).

The underlying curvature estimation is based on finding the angles subtended by 5 nearest neighbors of a point p, and a linearly weighted cosine of these angles. More weight is given to points that are far from point p in the curvature calculation. Note that $0 \leq C \leq 1$ for all points in the OFF curve. The point with the maximum curvature is added to candidate helper data CH for all the OFF curves.

3.2. Helper Data Filtering

The candidate helper data CH that is output by the curvature estimation routine given above can contain outliers (due to inexact localization of maximum curvature point). Furthermore, points with very low (corresponding to points on curves that are nearly flat) and very high curvature (corresponding to points that are too spiky) values should be filtered. This filtering operation results in the final helper data, H.

4. ICP based Alignment

The helper data is used in the fuzzy fingerprint vault system as follows: (i) Vault encoding: the helper data, H_E, associated with the enrollment fingerprint E (that is used for vault encoding) is saved along with the vault, V. (ii) Vault decoding: the helper data, H_Q, extracted from the query fingerprint is used for aligning the query and the associated template via template's helper data. Fig. 4 shows the helper data extraction for a template and query fingerprint pair.

Helper data H_E and H_Q are used as inputs to the Iterative Closest Point (ICP) based alignment routine summarized below. $T_{Q|E}$, which is a representation of the query minutiae template T_Q aligned with respect to the enrollment template T_E, is generated as the output and is used for vault decoding. The feature-weighted ICP algorithm [17] employed in this work is a modification of the generic ICP method of Besl and McKay [1]:

Step 1: Estimate the initial transformation between H_Q and H_E: The center of mass of maximum curvature points in H_Q is translated so that it coincides with the center of mass of points in H_E. Translate T_Q accordingly.

Step 2: Iterate until convergence (or maximum number of iterations is reached)

- Compute the set of correspondences between points in H_Q and H_E: Find the distance between a point in H_E (i.e., $p_E^i = (C_E^i, r_E^i, c_E^i)$, denoting the curvature value, row index, and the column index for a maximum curvature point, respectively) and a point in H_Q (i.e., $p_Q^i = (C_Q^i, r_Q^i, c_Q^i)$) as

$$d(p_E^i, p_Q^i) = \sqrt{(r_E^i - r_Q^i)^2 + (c_E^i - c_Q^i)^2} + \alpha |C_E^i - C_Q^i| \tag{9}$$

  where the parameter α determines the contribution of curvature based distance (second term) with respect to the Euclidean distance (first term).

- Compute the transformation that minimizes the mean square error between the paired points. Apply the transformation to H_Q and T_Q.

We have used the mean distance between corresponding points as the metric for convergence. The transformed query template, $T_{Q|E}$, is used during vault decoding.

5. Multiple ICP Applications and Helper Data Clustering

As explained above, the parameter α determines the contribution of curvature-based distance with respect to the Eu-

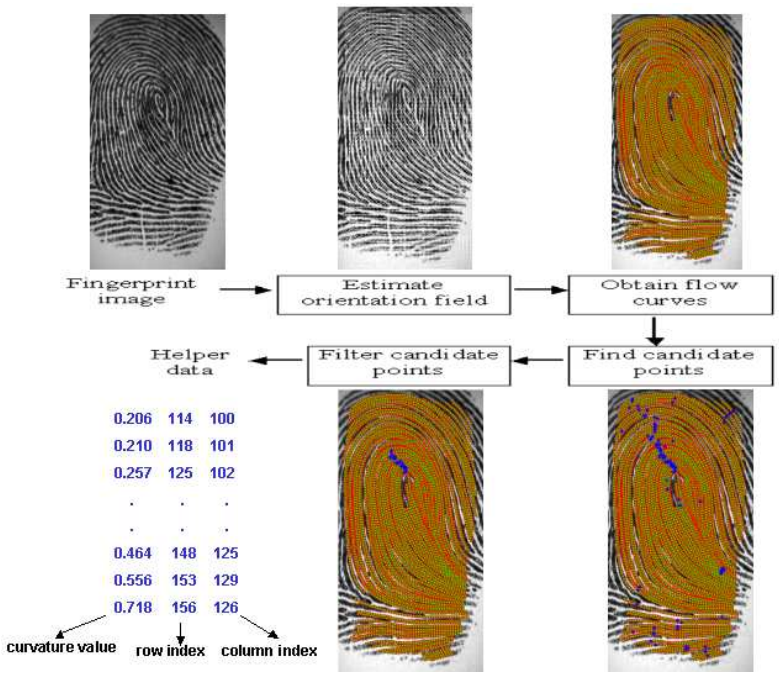


Figure 3. Generating helper data.

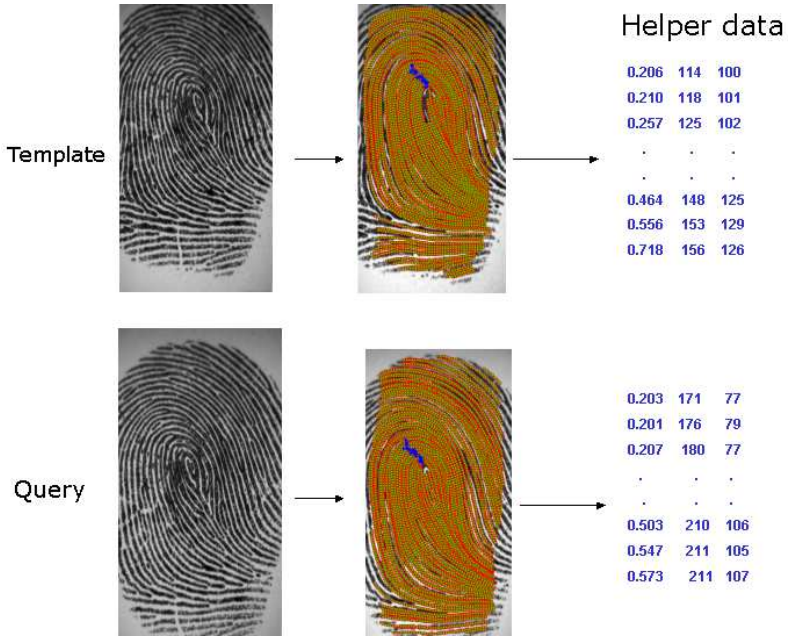


Figure 4. Extracting helper data from template (during vault encoding) and from query (during vault decoding).

clidean distance in ICP algorithm: higher α values emphasize the effect of curvature (hence, more appropriate when the Euclidean distances are unreliable, e.g., due to noisy helper data points). Accordingly, no single α value is found to be optimal for all the fingerprints. Hence, our algorithm uses multiple (3 in the current implementation) α values (100, 150, and 400) successively until the vault is decoded.

Furthermore, ICP algorithm is applied separately to individual clusters in helper data H_Q and H_E; especially for the whorl class of fingerprints, the helper data has two strong clusters (one above the core and one below the core). Applying ICP to the conglomerate data is not logical when one

of clusters is missing from either query or template helper data. This clustering is done using the Euclidean distances between neighboring points in the helper data set: if the distance between two neighboring points is larger than 30 pixels, a new cluster is formed. Fig. 5 shows this alignment process for the template and query fingerprints shown in Fig. 4.

6. Experiments and Results

We present the vault performance when automatic alignment scheme using helper data is employed (for experimental results not involving this alignment scheme, please see [19]). We have used DB2 database of FVC 2002 study [12] which contains 8 impressions for each of the 100 distinct fingers. Image size is 560x296 at a resolution of 569 dpi. For vault processing, a block size of 11x11 pixels and 24 genuine minutiae points dispersed among 200 chaff points are used.

Initially, two impressions per finger (impression number 1 for locking the vault, and impression number 2 for unlocking the vault) are used. The Genuine Accept Rate (GAR) is found to be 72.6% at FAR = 0%. Note that 16 fingers (out of 100) were rejected since they did not contain a sufficient number (24) of minutiae for locking or unlocking the vault. Using two impressions per finger (impression number 2 and number 7) for decoding the vault increases the GAR to 84.5% at 0% FAR. The false rejects were due to (i) errors in helper data (7 fingers), (ii) poor quality images where reliable helper data could not be extracted (4 fingers), and (iii) number of common minutiae between locking and unlocking prints less than the required number (2 fingers). Note that majority of these errors can be eliminated with increased user cooperation and habituation. We must also point out that the failure to enroll rate for fingerprint biometric can be as high as 4% [12].

The proposed helper data does not leak *any* information about the minutiae information used for locking and unlocking the vault. The helper data is extracted from the global flow of orientation field. On the other hand, the minutiae are local the discontinuities in the structure of ridges, and there is no way to learn minutiae features from the public helper data (cf. Fig. 5 shows the helper data overlaid with the critical minutiae information).

7. Conclusions

Privacy and security of biometric systems is gaining widespread attention as an increased number of such systems are being deployed in both commercial and government applications. Among the various issues, protecting the biometric template is a major concern. Biometric cryptosystems have been proposed to increase the associated security: they make use of transformed versions of the templates, and not the original template. Fuzzy vault construct is an example of such systems. In this paper, we have presented the results of a fuzzy vault implementation using fingerprint minutiae data. The vault performs with reasonable accuracy. It is shown that 128-bit AES keys can be feasibly secured using the proposed architecture. We have developed an automatic alignment scheme (between query and template) based on helper data derived from the orientation field of fingerprints. Utilizing maximum curvature information (invariant to translation and rotation of fingerprints) of orientation field flow curves, we align the query fingerprint with respect to the template via a variant of Iterative Closest Point (ICP) algorithm. The proposed alignment routine achieves reasonable accuracy, considering the small amount of data used for alignment. Further, the helper data does not leak *any* information about the minutiae-based fingerprint template. User habituation and cooperation has the potential to increase the authentication accuracy even further. In other words, since the system is developed for a positive identification scenario where the user is expected to be cooperative (for user convenience), the false rejects will reduce with increased user cooperation.

References

[1] P. Besl and N. McKay. A method for registration of 3-d shapes. *IEEE Trans. PAMI*, 14(2):239–256, February 1992. 5

[2] Y. Chung, D. Moon, S. Lee, S. Jung, T. Kim, and D. Ahn. Automatic alignment of fingerprint features for fuzzy fingerprint vault. In *Proc. CISC 2005, First SKLOIS Conference on Information Security and Cryptology*, pages 358–369, 2005. 2

[3] T. C. Clancy, N. Kiyavash, and D. J. Lin. Secure smartcard-based fingerprint authentication. In *Proc. ACM SIGMM Multimedia, Biometrics Methods and Applications Workshop*, pages 45–52, 2003. 2

[4] S. Dass and A. K. Jain. Fingerprint classification using orientation field flow curves. In *Proc. Indian Conf. Computer Vision, Graphics and Image Processing*, pages 650–655, 2004. 5

[5] J. G. Daugman. High confidence visual recognition of persons by a test of statistical independence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1148–1161, November 1993. 2

[6] G. I. Davida, Y. Frankel, and B. J. Matt. On enabling secure applications through off-line biometric identification. In *Proc. 1998 IEEE Symp. Privacy and Security*, pages 148–157, 1998. 2

[7] G. I. Davida, Y. Frankel, B. J. Matt, and R. Peralta. On the relation of error correction and cryptography to an offline biometric based identification scheme. In

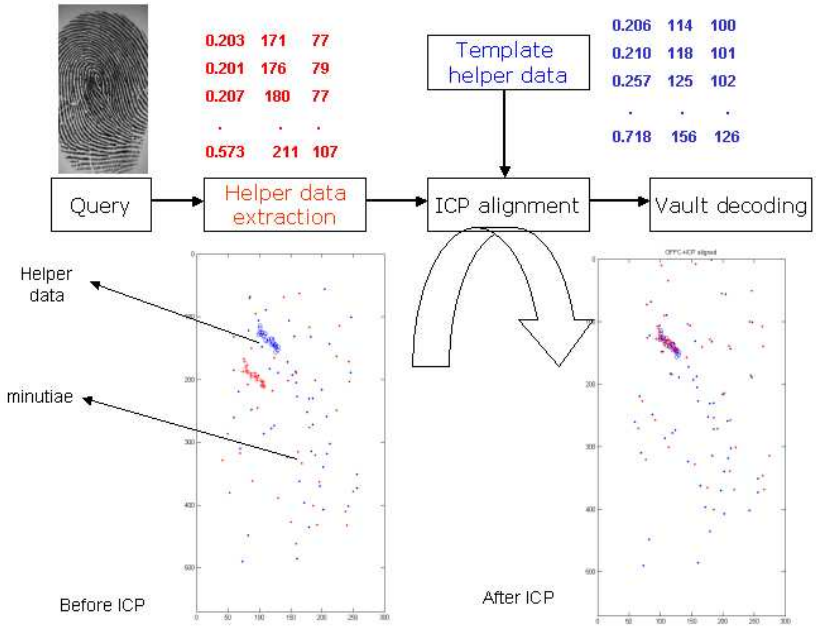


Figure 5. Aligning query and template using helper data.

Proc. Workshop on Coding and Cryptography (WCC), pages 129–138, 1999. 2

[8] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In *Proc. Int. Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 523–540, 2004. 2

[9] A. Juels and M. Sudan. A fuzzy vault scheme. In A. Lapidoth and E. Teletar, editors, *Proc. IEEE Int. Symp. Information Theory*, page 408, 2002. 1, 2, 3

[10] S. Lin and D. J. Costello. *Error Control Coding*. Pearson Prentice Hall, New Jersey, Second edition, 2004. 1

[11] J.-P. Linnartz and P. Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In *Proc. 4th Int. Conf. Audio- and Video-based Biometric Person Authentication*, pages 393–402, 2003. 1, 2

[12] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. Springer, New York, 2003. 1, 7

[13] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel. Cryptographic key generation from voice. In *Proc. IEEE Symp. Security and Privacy*, pages 202–213, 2001. 2

[14] F. Monrose, M. K. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. In *Proc. 6th ACM Conf. Computer and Communications Security*, pages 73–82, 1999. 2

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, Second edition, 1992. 3

[16] N. K. Ratha, J. H. Connell, and R. M. Bolle. An analysis of minutiae matching strength. In *Proc. Third Int. Conf. Audio- and Video-Based Biometric Person Authentication*, pages 223–228, 2001. 1

[17] G. Sharp, S. Lee, and D. Wehe. ICP registration using invariant features. *IEEE Trans. PAMI*, 24(1):90–102, January 2002. 5

[18] C. Soutar, D. Roberge, S. A. Stojanov, R. Gilroy, and B. V. K. V. Kumar. Biometric encryption. In R. K. Nichols, editor, *ICSA Guide to Cryptography*. McGraw Hill, 1999. 1

[19] U. Uludag, S. Pankanti, and A. Jain. Fuzzy vault for fingerprints. In *Proc. AVBPA 2005, 5. International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 310–319, 2005. 2, 7

[20] U. Uludag, S. Pankanti, S. Prabhakar, and A. K. Jain. Biometric cryptosystems: issues and challenges. *Proceedings of the IEEE*, 92(6):948–960, June 2004. 1

[21] S. Yang and I. Verbauwhede. Automatic secure fingerprint verification system based on fuzzy vault scheme. In *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pages 609–612, 2005. 2