Fingerprint-Based Fuzzy Vault: Implementation and Performance

Karthik Nandakumar, *Student Member, IEEE*, Anil K. Jain, *Fellow, IEEE*, and Sharath Pankanti, *Senior Member, IEEE*

Abstract—Reliable information security mechanisms are required to combat the rising magnitude of identity theft in our society. While cryptography is a powerful tool to achieve information security, one of the main challenges in cryptosystems is to maintain the secrecy of the cryptographic keys. Though biometric authentication can be used to ensure that only the legitimate user has access to the secret keys, a biometric system itself is vulnerable to a number of threats. A critical issue in biometric systems is to protect the template of a user which is typically stored in a database or a smart card. The fuzzy vault construct is a biometric cryptosystem that secures both the secret key and the biometric template by binding them within a cryptographic framework. We present a fully automatic implementation of the fuzzy vault scheme based on fingerprint minutiae. Since the fuzzy vault stores only a transformed version of the template, aligning the query fingerprint with the template is a challenging task. We extract high curvature points derived from the fingerprint orientation field and use them as helper data to align the template and query minutiae. The helper data itself do not leak any information about the minutiae template, yet contain sufficient information to align the template and query fingerprints accurately. Further, we apply a minutiae matcher during decoding to account for nonlinear distortion and this leads to significant improvement in the genuine accept rate. We demonstrate the performance of the vault implementation on two different fingerprint databases. We also show that performance improvement can be achieved by using multiple fingerprint impressions during enrollment and verification.

Index Terms—Biometric cryptosystems, fingerprint alignment, fuzzy vault, helper data, minutiae, minutiae mosaicing, polynomial reconstruction.

I. INTRODUCTION

CRYPTOGRAPHIC techniques are being widely used for ensuring the secrecy and authenticity of information [1]. Although several cryptosystems have proven security guarantees (e.g., AES and RSA), the security relies on the assumption that the cryptographic keys are known only to the legitimate user. Maintaining the secrecy of keys is one of the main challenges in practical cryptosystems. The keys are usually stored
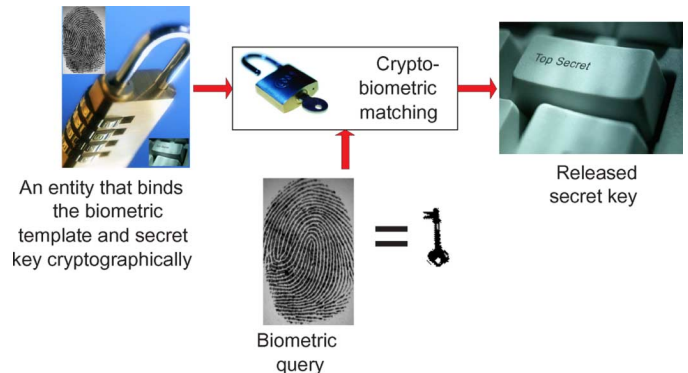
Fig. 1. Operation of a biometric cryptosystem in the key binding mode.

in a secure location (e.g., tamper-resistant hardware) and password-based authentication is commonly used for controlling access to cryptographic keys [2]. However, passwords can be easily lost, stolen, forgotten, or guessed using social engineering [3] and dictionary attacks. Limitations of password-based authentication can be alleviated by using stronger authentication schemes, such as biometrics. Biometric systems [4] establish the identity of a person based on his or her anatomical or behavioral traits, such as face, fingerprint, iris, voice, etc. Biometric authentication is more reliable than password-based authentication because biometric traits cannot be lost or forgotten and it is difficult to share or forge these traits. Hence, biometric systems offer a natural and reliable solution to the problem of user authentication in cryptosystems.

Biometric cryptosystems can operate in one of the following three modes: 1) key release; 2) key binding; and 3) key generation. In the key-release mode, biometric authentication is completely decoupled from the key release mechanism. The biometric template and the key are stored as separate entities and the key is released only if the biometric matching is successful. In the key binding mode, the key and the template are monolithically bound within a cryptographic framework (see Fig. 1). It is computationally infeasible to decode the key or the template without any knowledge of the user's biometric data. A crypto-biometric matching algorithm is used to perform authentication and key release in a single step. In the key generation mode, the key is derived directly from the biometric data and is not stored in the database.

Though it is easy to implement a biometric cryptosystem in the key release mode, such a system is not appropriate for high security applications because it has two major vulnerabilities. First, the biometric template is not secure. Template security is a critical issue in biometric systems because stolen templates cannot be revoked. Second, since authentication and key release

Fig. 2. Illustration of intraclass variability in fingerprints. Two different impressions of the same finger obtained on different days are shown with minutia points marked on them. Due to translation, rotation, and distortion, the number and location of minutiae in the two images are different. The number of minutiae in the left and right images is 33 and 26, respectively. The number of common minutiae in the two images is 16 and few of these correspondences have been indicated in the figure.

are decoupled, it is possible to override the biometric matcher using a Trojan horse program [5]. Biometric cryptosystems that work in the key binding/generation modes are more secure but difficult to implement due to large intraclass variations in biometric data (i.e., samples of the same biometric trait of a user obtained over a period of time can differ substantially). For example, factors, such as translation, rotation, nonlinear distortion, skin conditions, and noise lead to intraclass variations in fingerprints [6] (see Fig. 2).

Juels and Sudan [7] proposed a cryptographic construction called a fuzzy vault that operates in the key binding mode and, in principle, can compensate for intraclass variations in the biometric data. We present a fully automatic implementation of the fuzzy vault scheme based on fingerprint minutiae. Since the fuzzy vault scheme stores only a transformed version of the template, the main implementation challenge is the alignment (registration) of query fingerprint to the original template. In our implementation, we store high curvature points derived from the orientation field of the template fingerprint as helper data to assist in alignment. The helper data accurately align the template and query minutiae but do not reveal any information about the minutia points that form the template.

The rest of this paper is organized as follows. Section II introduces the basic fuzzy vault construction and Section III gives the details of the proposed fuzzy vault implementation using fingerprint minutiae. The automatic fingerprint alignment technique based on helper data is described in Section IV. The experimental results are presented in Section V, and Section VI presents a discussion on the security of the system. Section VII summarizes our work and provides pointers for future research.

II. BACKGROUND

Several schemes have been proposed in the literature for implementing a biometric cryptosystem in key binding/generation modes [8]–[13]. It is also worth mentioning that the problem of key binding/generation in a biometric cryptosystem is closely related to other research issues in biometrics, such as template protection [14], [15] and generation of cancelable templates [16]. A detailed review of different biometric cryptosystems has been presented in [17]. Based on this study, it is clear that

the techniques proposed in [8]–[15] have their own advantages and limitations in terms of security, computational cost, storage requirements, applicability to different kinds of biometric representations, and ability to handle intraclass variations in biometric data. In this paper, we focus on a specific algorithm, known as fuzzy vault, which was originally proposed by Juels and Sudan in [7].

A. Fuzzy Vault

Fuzzy vault [7] is a cryptographic construction that is designed to work with biometric features which are represented as an unordered set (e.g., minutiae in fingerprints). The security of the fuzzy vault scheme is based on the infeasibility of the polynomial reconstruction problem, which is a special case of the Reed–Solomon list decoding problem. The ability to deal with intraclass variations in the biometric data along with the ability to work with unordered sets, which is commonly encountered in biometrics, makes the fuzzy vault scheme a promising solution for biometric cryptosystems.

Fig. 3 depicts the operation of a fuzzy vault scheme. Suppose that a user wishes to hide a secret K (e.g., a cryptographic key) using his or her biometric sample (template) which is represented as an unordered set X. The user selects a polynomial $\mathcal{P}$ that encodes the secret K and evaluates the polynomial on all elements in X. The user then chooses a large number of random chaff points which do not lie on the polynomial $\mathcal{P}$. The entire collection of points consisting of both points lying on $\mathcal{P}$ and those that do not lie on $\mathcal{P}$ constitute the vault V. The chaff points conceal the genuine points lying on $\mathcal{P}$ from an attacker. Since the points lying on $\mathcal{P}$ encode the complete information about the template X and the secret K, concealing these points secures both the template and the secret simultaneously.

The user can retrieve the secret K from the vault V by providing another biometric sample (query). Let the query be represented as another unordered set X'. If X' overlaps substantially with X, then the user can identify many points in V that lie on $\mathcal{P}$. If a sufficient number of points on $\mathcal{P}$ can be identified, an error correction scheme can be applied to exactly reconstruct $\mathcal{P}$ and thereby decode the secret K. If X' does not overlap substantially with X, it is infeasible to reconstruct $\mathcal{P}$ and the authentication is unsuccessful. Since the secret can be retrieved from the vault even when X and X' are not exactly the same, this scheme is referred to as a fuzzy vault.

The three main parameters in the fuzzy vault scheme are r, s, and n. The parameter r denotes the number of points in the vault that lie on the polynomial $\mathcal{P}$ and it depends on the number of features that can be extracted from the template (e.g., number of minutia points in the user's fingerprint). The parameter s represents the number of chaff points that are added and this parameter influences the security of the vault. If no chaff points are added, the vault reveals the information about the template and the secret. As more chaff points are added, the security increases. Typically, the number of chaff points is an order of magnitude larger than the number of genuine points $(s \gg r)$. Parameter n denotes the degree of the encoding polynomial and it controls the tolerance of the system to errors in the biometric data.

In order to retrieve the secret from the vault V, the user selects a subset of r points from V, which is known as the unlocking set. The unlocking set is selected based on the query
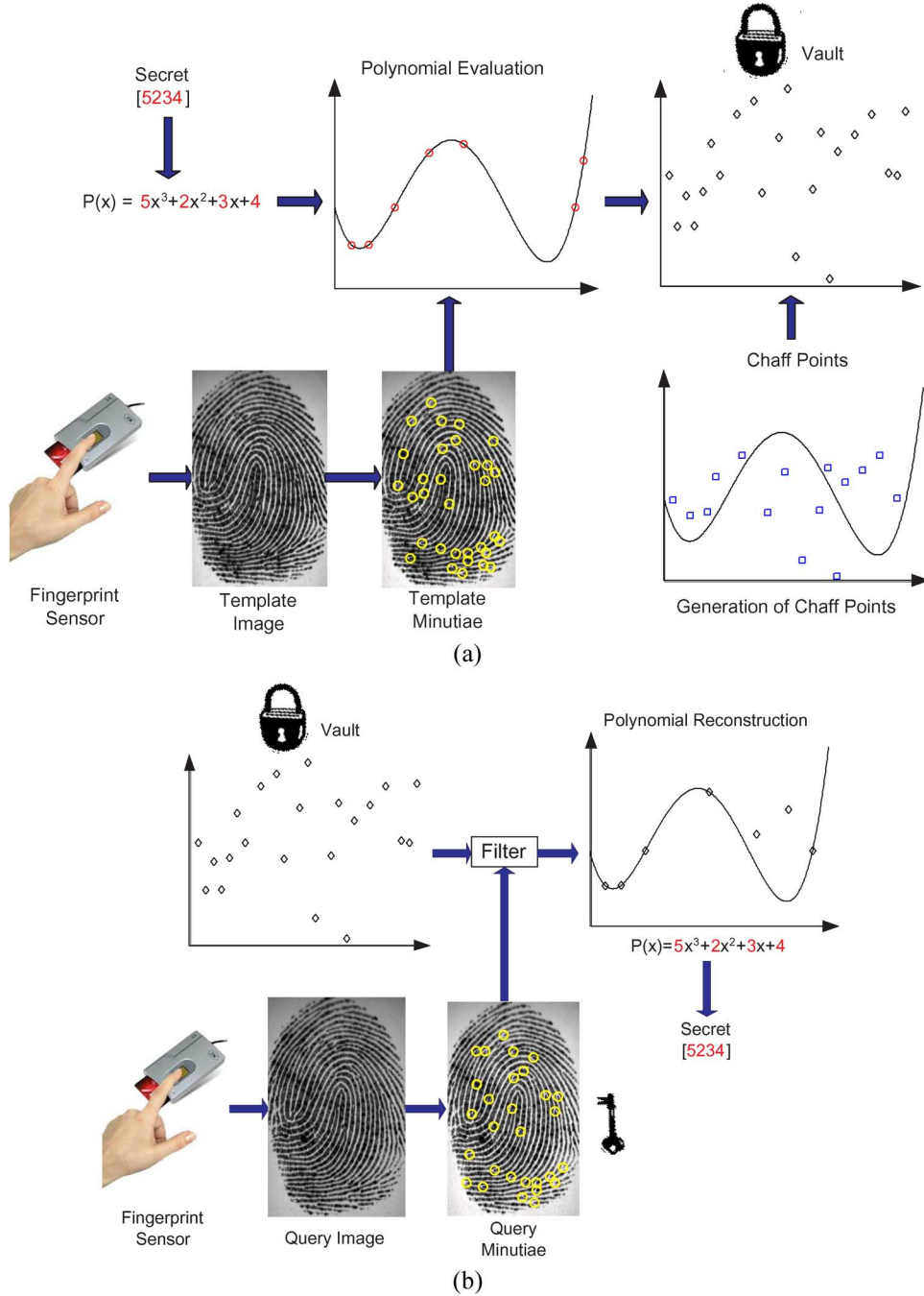
Fig. 3. Operation of the fuzzy vault scheme in [7] based on fingerprint minutiae. (a) Vault encoding. (b) Vault decoding.

X'. A (r, n) Reed–Solomon decoding algorithm [18] is then applied to search for a polynomial $\mathcal{P}$ of degree n such that there are more than $(r + n)/2$ points[1] in the unlocking set lie on $\mathcal{P}$. If the number of discrepancies in the biometric data $(|X - X'|)$ are less than $(r - n)/2$, a valid polynomial $\mathcal{P}$ can be found and the secret K can be successfully retrieved.

B. Fuzzy Vault Implementation

Since the introduction of the fuzzy vault scheme, several researchers have attempted to implement it in practice. Clancy *et*

al. [19] proposed a fuzzy vault scheme based on the location of minutia points in a fingerprint. They assumed that the template and query minutiae sets are prealigned, which is not a realistic assumption in practice. Further, four fingerprint impressions of a user were used during enrollment for identifying the reliable minutia points and the error-correction step was simulated without being actually implemented. The false reject rate[2] of their system was 20–30% and they claimed that retrieving the secret was 2^{69} times more difficult for an attacker than for a genuine user.

---

[1]The Berlekamp–Walsh algorithm [18] for Reed–Solomon list decoding can correct up to e errors only if $(n + 2e) < r$.

[2]The false reject rate (FRR) refers to the percentage of attempts in which the authentication is unsuccessful for a genuine user.

The fingerprint-based fuzzy vault proposed by Yang and Verbauwhede [20] also used only the location information about the minutia points. Four impressions were used during enrollment to identify a reference minutia and the relative position of the remaining minutia points with respect to the reference minutia was represented in the polar coordinate system. This scheme was evaluated on a small database of ten fingers and an FRR of 17% was reported. Chung *et al.* [21] proposed a geometric hashing technique to perform alignment in a minutiae-based fingerprint fuzzy vault. Fuzzy vault implementations based on other biometric modalities, such as face [22] and handwritten signature [23], have also been proposed.

Uludag *et al.* [24] introduced a modification to the fuzzy vault scheme, which eliminated the need for Reed–Solomon polynomial decoding. They also proposed the use of helper data to automatically align the template and query minutiae sets [25]. Our fuzzy vault implementation extends the ideas presented in [24] and [25] in order to achieve better recognition performance.

III. PROPOSED FUZZY VAULT IMPLEMENTATION

The proposed fuzzy vault implementation is based on minutiae which are local ridge characteristics (endings and bifurcations) in a fingerprint. We use both the location and orientation attributes of a minutia point in our implementation. These attributes are represented as a three-tuple (u, v, θ), where u and v indicate the row and column indices in the image, and θ represents the orientation of the minutia with respect to the horizontal axis ($1 \leq \theta \leq 360$). The algorithm described in [26] is used for minutiae extraction.

We have implemented the modified fuzzy vault construction proposed in [24]. This modified fuzzy vault scheme does not require Reed–Solomon decoding. Instead, several candidate sets of size $(n + 1)$ (where n is the degree of the polynomial which encodes the secret) are generated from the unlocking set and polynomials are reconstructed using Lagrange interpolation. This method gives rise to several candidate secrets and a cyclic redundancy check (CRC)-based error detection technique is used to identify the correct polynomial and, hence, decode the correct secret. The advantage of this scheme is its increased tolerance to biometric intraclass variations. Since only $(n + 1)$ points are required to uniquely determine a polynomial of degree n, this scheme can retrieve the secret K when the number of discrepancies in the biometric data ($|X - X'|$) is less than $r - n$. However, this method has a higher computational cost because it requires a large number of polynomial interpolations.

Our fuzzy vault implementation significantly differs from the implementation proposed in [25] in the following aspects.

1) In our implementation, we apply a minutiae matcher [26] during decoding to account for nonlinear distortion in fingerprints whereas in [25], the minutia location information is coarsely quantized to compensate for distortion. Since deformation of the fingerprint ridges increases as we move away from the center of the fingerprint area toward the periphery, uniform quantization alone, as used in [25], is not sufficient to handle distortion. The minutiae matcher used in our implementation [26] employs an adaptive bounding box that accounts for distortion more effectively. This is one of the main reasons why the proposed approach leads to a significant improvement in the genuine accept rate (GAR).

2) Only the location of minutia points was used for vault encoding in [25]. We use both minutia location and orientation attributes which increases the number of chaff points that can be added because we can now add a chaff point whose location is close to a true minutia but with a different direction. Chang *et al.* [27] have shown that the number of possible chaff points affects the security of the vault. Hence, using both minutia location and orientation makes it more difficult for an attacker to decode the vault. At the same time, when a genuine user is attempting to decode the vault, it is now easier to filter out chaff points from the vault because it is less probable that a chaff point will match the query minutia in both location and direction. This reduces the decoding complexity by eliminating most of the chaff points from the unlocking set.

3) We use a local image-quality index estimated from the fingerprint in order to select the most reliable minutiae for vault encoding and decoding. This also leads to a higher GAR compared to the implementation in [25].

4) Although our alignment technique is similar to the one proposed in [25], we have made significant changes to the curvature estimation and alignment steps compared to [25] which results in more accurate alignment between the template and query.

A. Vault Encoding

Fig. 4 shows the block diagram of the proposed fuzzy vault encoding scheme. We use the Galois field $\mathcal{F} = GF(2^{16})$ for constructing the vault. The specific field $GF(2^{16})$ was chosen because it offers a sufficiently large universe (number of elements in the field) to ensure vault security [12] and is computationally convenient for the fuzzy vault application. The vault is encoded as follows.

1) Given the template fingerprint image T, we first obtain the template minutiae set $M^T = \{m_i^T\}_{i=1}^{N^T}$, where N^T is the number of minutiae in T. The local quality index proposed in [28] is used to estimate the quality of each minutia in T. Let $q(m_i^T)$ be the quality of the ith minutia and $q^T = \{q(m_i^T)\}_{i=1}^{N^T}$ be the quality set corresponding to minutiae set M^T. We also extract the helper data set H^T from the template image to be used for alignment during decoding. The details of helper data extraction are presented in Section IV-A.

2) Since only r genuine minutiae points are required to construct the vault, we apply a minutiae selection algorithm to the template minutiae set M^T. This selection algorithm sorts the minutiae based on their quality and sequentially selects the minutiae starting with the highest quality minutia. Moreover, the algorithm selects only well-separated minutiae (i.e., the minimum distance between any two selected minutia points is greater than a threshold δ_1). The distance D_M between two minutia points m_i and m_j is defined as

$$D_M(m_i, m_j) = \sqrt{(u_i - u_j)^2 + (v_i - v_j)^2} + \beta_M \Delta(\theta_i, \theta_j) \quad (1)$$

where $\Delta(\theta_i, \theta_j) = \min(|\theta_i - \theta_j|, 360 - |\theta_i - \theta_j|)$ and β_M are the weight assigned to the orientation attribute (set
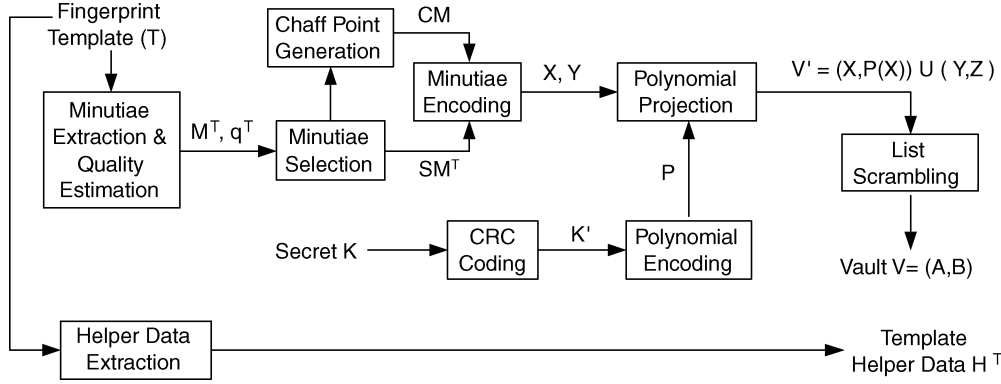
Fig. 4. Proposed implementation of vault encoding.

to 0.2 in our experiments[3]). The selection of well-separated minutiae ensures that they are assigned unique values when they are encoded into the field $\mathcal{F}$. Let $SM^T = \{m_j^T\}_{j=1}^r$ denote the selected minutiae set. Note that if the number of minutia points in T is less than r or if the selection algorithm fails to find r well-separated minutiae, we consider it as a failure-to-capture (FTC) error and no further processing takes place.

3) The chaff point set $CM = \{m_k\}_{k=1}^s$ is generated iteratively as follows. A chaff point $m = (u, v, \theta)$ is randomly chosen such that $u \in \{1, 2, \cdots, U\}$, $v \in \{1, 2, \cdots, V\}$, and $\theta \in \{1, 2, \cdots, 360\}$. The point m is added to CM if the minimum distance [as defined in (1)] between m and all points in the set $SM^T \cup CM$ is greater than δ_1.

4) The minutia attributes u, v, and θ are quantized and represented as bit strings of length $\mathcal{B}_u$, $\mathcal{B}_v$, and $\mathcal{B}_\theta$, respectively. If $\mathcal{B}_u$, $\mathcal{B}_v$, and $\mathcal{B}_\theta$ are chosen such that they add up to 16, we can obtain a 16-b number by concatenating the bit strings corresponding to u, v, and θ. Using this method, minutia points are encoded as elements in the field $\mathcal{F} = GF(2^{16})$. Let $X = \{x_j\}_{j=1}^r$ and $Y = \{y_k\}_{k=1}^s$ be the encoded values of selected template minutiae and chaff points, respectively, in the field $\mathcal{F}$.

5) Our scheme is designed to secure a secret K of length $16n$ b, where n is the degree of the encoding polynomial. We append a 16-b CRC code to secret K to obtain a new secret K' containing $16(n + 1)$ bits. The generator polynomial $G(w) = w^{16} + w^{15} + w^2 + 1$, which is commonly known as IBM CRC-16, is used for generating the CRC bits.

6) The secret K' is encoded into a polynomial $\mathcal{P}$ of degree n in $\mathcal{F}$ by partitioning it into $(n + 1)$ 16-b values $c_0, c_1, \cdots, c_n$ and considering them as coefficients of $\mathcal{P}$ (i.e., $\mathcal{P}(x) = c_n x^n + \cdots + c_0$).

7) The polynomial $\mathcal{P}$ is evaluated at all of the points in the selected minutiae set X to obtain the set $\mathcal{P}(X) = \{\mathcal{P}(x_j)\}_{j=1}^r$. The corresponding elements of the sets X and $\mathcal{P}(X)$ form the locking set $L = \{(x_j, \mathcal{P}(x_j))\}_{j=1}^r$. A set $Z = \{z_k\}_{k=1}^s$ is ob-

tained by randomly selecting values $z_k \in \mathcal{F}$ such that the points (y_k, z_k) do not lie on the polynomial $\mathcal{P}$ (i.e, $z_k \neq \mathcal{P}(y_k), \forall, k = 1, 2, \cdots, s)$. The chaff set is defined as $C = \{(y_k, z_k)\}_{k=1}^s$. The union of locking and chaff sets is denoted as V'.

8) The elements of V' are randomly reordered to obtain the vault V which is represented as $V = \{(a_i, b_i)\}_{i=1}^t$, where $t = r + s$. Only the vault V and the helper data H^T are stored in the system.

B. Vault Decoding

The process of decoding the vault consists as follows (see Fig. 5).

1) Given the query fingerprint image Q, we obtain the query minutiae set $M^Q = \{m_i^Q\}_{i=1}^{N^Q}$ and the helper data set H^Q. The quality of each minutia in Q is estimated and the quality set $q^Q = \{q(m_i^Q)\}_{i=1}^{N^Q}$ corresponding to M^Q is obtained.

2) The alignment algorithm described in Section IV-B is applied and the aligned query minutiae set $M^{AQ} = \{m_i^{AQ}\}_{i=1}^{N^Q}$ is obtained.

3) A minutiae selection algorithm is applied to select r minutiae from the set M^{AQ} based on their quality. The selected minutiae $SM^Q = \{m_j^Q\}_{j=1}^r$ are well separated in the sense that the minimum distance [as defined in (1)] between any two selected minutiae is greater than δ_1. If $N^Q < r$ or if the number of well-separated query minutiae is less than r, it is considered as FTC and no further processing takes place.

4) The selected query minutiae are used to filter the chaff points in the vault as follows [see Fig. 5(b)]. The abscissa values of the points in the vault (i.e., $A = \{a_i\}_{i=1}^t$) are first represented as 16-b strings. The 16-b strings are partitioned into three strings of lengths $\mathcal{B}_u$, $\mathcal{B}_v$, and $\mathcal{B}_\theta$, which are then converted into quantized minutia attribute values u, v and θ. Thus, we obtain the set $M^V = \{m_i^V = (u_i, v_i, \theta_i)\}_{i=1}^s$.

5) The ith element of the set M^V is marked as a chaff point if the minimum distance between the point $m_j^V \in M^V$ and all of the selected minutiae in the query $m_j^Q \in SM^Q$ is greater than a threshold δ_2. We refer to this process as a coarse filter and it filters out a significant proportion of the chaff points (approximately 80%). Let $SM^V = \{m_k^V\}_{k=1}^{N^V}$
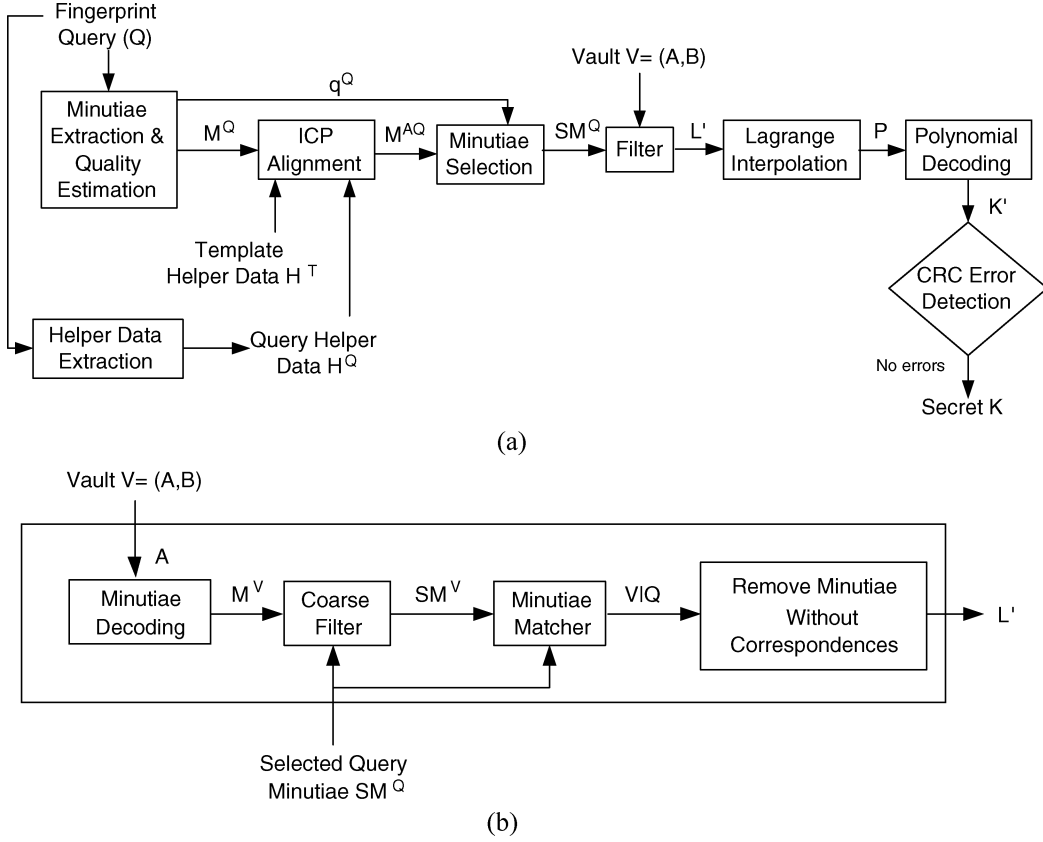
Fig. 5. Proposed implementation of vault decoding. (a) Block diagram of the complete decoding process. (b) Details of the filter used to eliminate the chaff points.

be a subset of M^V containing only those elements that are not marked as chaff. Here, N^V is the number of points in M^V that are not marked as chaff and $N^V \ll s$. At this stage, a minutiae matcher [26] is applied to determine the corresponding pairs of minutiae in the sets SM^V and SM^Q. Let $V|Q$ denote the set of correspondences and let r' be the number of correspondences. Since the size of the selected query minutiae set is r, we have $0 \leq r' \leq r$ because each query minutiae can have no more than one corresponding minutia in SM^V. Note that it is also possible to directly apply the minutiae matcher to find correspondences between M^V and SM^Q without any coarse filtering. However, such a method is not effective because the presence of a large number of chaff points in the vault leads to a number of false correspondences. Hence, the coarse filter step is essential before the minutiae matcher is applied.

6) Only those elements of V that are contained in SM^V and which have a corresponding minutia in SM^Q are added to the unlocking set L'. The unlocking set is represented as $L' = \{(a_i', b_i')\}_{i=1}^{r'}$, where $(a_i', b_i') = (a_j, b_j)$ if a_j has a corresponding minutia in SM^Q.

7) To find the coefficients of a polynomial of degree n, $(n+1)$ unique projections are necessary. If $r' < (n+1)$, it results in authentication failure. Otherwise, we consider all possible subsets L'' of size $(n+1)$ of the unlocking set L' and, for each subset, we construct a polynomial $\mathcal{P}^*$ by Lagrange interpolation. If $L'' = \{(a_i', b_i')\}_{i=1}^{n+1}$ is a specific candidate set, $\mathcal{P}^*(x)$ is obtained as

$$\mathcal{P}^*(x) = \frac{(x - a_2')(x - a_3') \cdots (x - a_{n+1}')}{(a_1' - a_2')(a_1' - a_3') \cdots (a_1' - a_{n+1}')} b_1' + \cdots$$
$$+ \frac{(x - a_1')(x - a_2') \cdots (x - a_n')}{(a_{n+1}' - a_1')(a_{n+1}' - a_2') \cdots (a_{n+1}' - a_n')} b_{n+1}'. \quad (2)$$

The aforementioned operations result in a polynomial $\mathcal{P}^*(x) = c_n^* x^n + c_{n-1}^* x^{n-1} + \cdots + c_0^*$.

8) The coefficients $c_0^*, c_1^*, \cdots, c_n^*$ of the polynomial $\mathcal{P}^*$ are 16-b values which are concatenated to obtain a $16(n+1)$-bit string K^*, and CRC error detection is applied to K^*. If an error is detected, it indicates that an incorrect secret has been decoded and we repeat the same procedure for the next candidate set L''. If no error is detected, it indicates $K^* = K'$ with very high probability. In this case, the 16-b CRC code is removed from K^* and the system outputs the secret K.

IV. FINGERPRINT ALIGNMENT BASED ON HELPER DATA

The first step in matching two fingerprint images is to align them and determine the area of overlap. Although aligning two fingerprints is a difficult problem in any fingerprint authentication system, it is more difficult in a biometric cryptosystem such as a fuzzy vault. This is because the original fingerprint template is not available during authentication and only a transformed version of the template is available in the fuzzy vault. Previous implementations of the fingerprint-based fuzzy vault
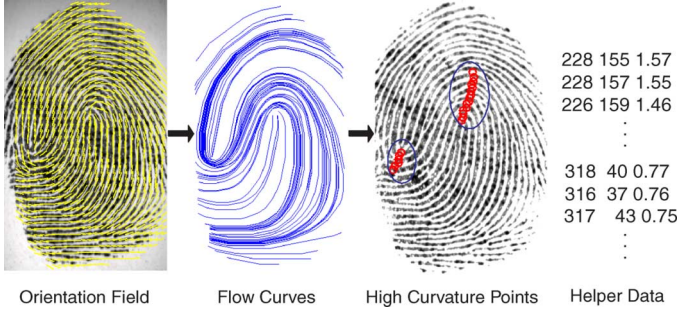
Fig. 6. Algorithm for helper data extraction.

either assumed that the template and query fingerprint images are prealigned [19] or used as a reference point (e.g., core point [29] or a reference minutia point [20]) for alignment. Though alignment based on a reference point is simple and computationally efficient, it is difficult to determine the reference point reliably and errors in locating the reference point could lead to false rejects. To avoid this problem, we use additional information derived from the fingerprint image, called helper data, to assist in alignment.

Since the helper data are stored as public information, it should not reveal any information about the template minutiae used for constructing the vault because any such leakage would compromise the security. On the other hand, the helper data should carry sufficient information for accurate alignment. Following the approach of Uludag and Jain [25], we extract points of high curvature from the fingerprint orientation field. A trimmed iterative closest point (ICP) algorithm is used to determine the alignment between the template and the query based on these helper data. Since high curvature points are global features in the fingerprint pattern, they do not reveal any information about the minutia attributes which are local characteristics in the fingerprint. Moreover, the helper data do not contain sufficient information to estimate the orientation field or reconstruct the complete fingerprint pattern because the location and curvature values of high curvature points extracted from different classes of fingerprint patterns can be quite similar [30]. Therefore, the proposed helper data do not affect the security of the fuzzy vault.

A. Helper Data Extraction

An orientation field flow curve [31] is a set of piecewise linear segments whose tangent direction at each point is parallel to the orientation field direction at that point. Although flow curves are similar to fingerprint ridges, the extraction of flow curves is not affected by breaks and discontinuities which are commonly encountered in ridge extraction. Points of maximum curvature in the flow curves, along with their curvature values, constitute the helper data used in our alignment scheme. An algorithm for the extraction of helper data (see Fig. 6) consists of four steps: 1) orientation field estimation; 2) extraction of flow curves; 3) determination of maximum curvature points; and 4) clustering of high curvature points.

Let $\mathcal{I}$ be a fingerprint image with U rows and V columns. A robust estimate of the orientation field for the given fingerprint image is obtained using the algorithm described in [32].

The flow curve extraction technique described in [31] is used to obtain the set of flow curves based on the estimated orientation field. Each flow curve is represented as a set of points $\{\ell_j\}_{j=1}^{J}$, where J is the number of points in the curve and $\ell_j = (\lambda_j, \mu_j)$ is a point in $\mathcal{I}$, $1 \leq \lambda_j \leq U$ and $1 \leq \mu_j \leq V$, and $\forall\, j = 1, 2, \cdots, J$. The sampling interval for the points in a flow curve is set to five pixels and the maximum number of samples in a curve is set to 300. Midpoints of the thinned ridges and points in whose neighborhood the orientation field changes significantly are chosen as the starting points for the flow curve extraction.

The curvature (ω) of a point ℓ_j in a flow curve is defined as $\omega_{\ell_j} = 1 - \cos \alpha_{\ell_j}$, where α_{ℓ_j} is the angle between the tangents to the flow curve at the points $\ell_{j-\tau}$ and $\ell_{j+\tau}$, $\forall\, \tau \leq j \leq J - \tau$. The parameter τ is related to the sampling interval of the flow curve and is set to 5. The value of ω_{ℓ_j} is minimum (0) if there is no change in direction as we go from $\ell_{j-\tau}$ to $\ell_{j+\tau}$ through ℓ_j and is maximum (2) if the change in direction is π. The curvature values for the points in the flow curve are estimated and local maxima in the curvature are detected. If the value of the local maximum is greater than σ (set to 0.3), the point is marked as a high curvature point and the 3-tuple $h = (\lambda, \mu, \omega)$, where (λ, μ) is the location and ω is the curvature value, is added to the helper data set H. Fig. 7 shows the procedure for curvature estimation at a point and a trace of the curvature values for a sample flow curve. Maximum curvature points of all the flow curves are found and the final helper data set is obtained as $H^{\mathcal{I}} = \{h_i\}_{i=1}^{R^{\mathcal{I}}}$, where $R^{\mathcal{I}}$ is the number of helper data points in $\mathcal{I}$. High curvature points tend to occur near the singular points in the fingerprint image. If the image has more than one singularity, points in the helper data set may have many clusters which are identified by applying a single-link clustering algorithm.

B. Alignment Using ICP

Let T and Q be the template and query fingerprint images, respectively. Let $H^T = \{h_i^T\}_{i=1}^{R^T}$ and $H^Q = \{h_j^Q\}_{j=1}^{R^Q}$ be the helper data sets obtained from T and Q, respectively. Let $M^Q = \{m_j^Q\}_{j=1}^{N^Q}$ be the query minutiae set, where N^Q is the number of minutia points in Q. We use the ICP algorithm proposed by Besl and McKay [33] to align H^T and H^Q and estimate the rigid transformation F. The ICP algorithm starts with two coarsely aligned point sets and iteratively finds the point correspondences and the rigid transformation between them (for more details, see [30]). Since the ICP algorithm strictly assigns a correspondence between every query helper point and a template helper point, there may be alignment errors when the overlap between the two sets is partial. To overcome this problem, we use the trimmed ICP algorithm [34], which ignores a proportion of points in the query set whose distance to the corresponding points in the template set is large. The trimmed ICP algorithm is robust to outliers in the helper data.

Based on the rigid transformation F output by the ICP algorithm, we align the query minutiae set M^Q with the template. Let $M^{AQ} = F(M^Q) = \{m_j^{AQ}\}_{j=1}^{N^Q}$ represent the query minutiae set after alignment. Fig. 8 shows an example of successful minutiae alignment based on helper data and trimmed ICP algorithm. If the number of clusters in H^T and/or H^Q is more than one, the ICP algorithm is repeated for all possible cluster
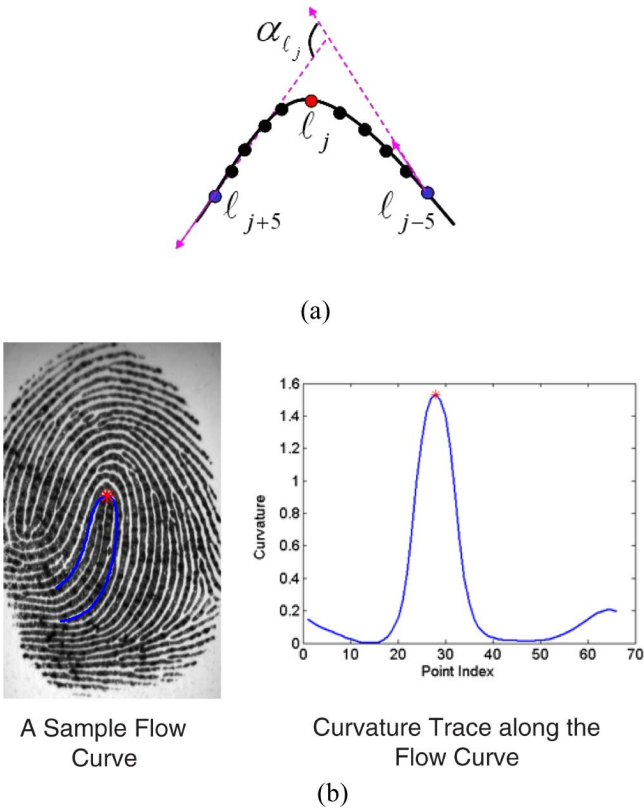
(a)

(b)

Fig. 7. Determination of maximum curvature points. (a) Curvature estimation at point ℓ_j. (b) Trace of curvature for a sample flow curve along with the local maximum.

pairs. In this scenario, there will be multiple aligned query minutiae sets. We select the aligned query minutiae set that gives the largest unlocking set L'.

V. EXPERIMENTAL RESULTS

The performance of the proposed fuzzy vault implementation has been evaluated on FVC2002-DB2 [35] and MSU–DBI [36] fingerprint databases. The characteristics of these two databases are summarized in Table I. We consider the following three scenarios for vault implementation.

1) One impression is used for encoding and one impression is used for decoding.
2) Two impressions are used for encoding and one impression is used for decoding.
3) Two impressions are used for encoding and two impressions are used for decoding.

When multiple impressions are available for vault encoding, we apply a mosaicing technique [37] to combine the minutiae and helper data from the individual images into a single mosaiced template and helper set. When multiple impressions are available for decoding, we use them sequentially to unlock the vault. The decoding is successful if at least one of the two queries succeeds in unlocking the vault.

The MSU–DBI database contains two pairs of impressions for each user and these two pairs were collected six weeks apart. Hence, this database is suitable to study the multiple impression scenarios that are considered here. The FVC2002-DB2 database was selected because it is a public-domain database and the images are of relatively good quality. Among the eight impressions
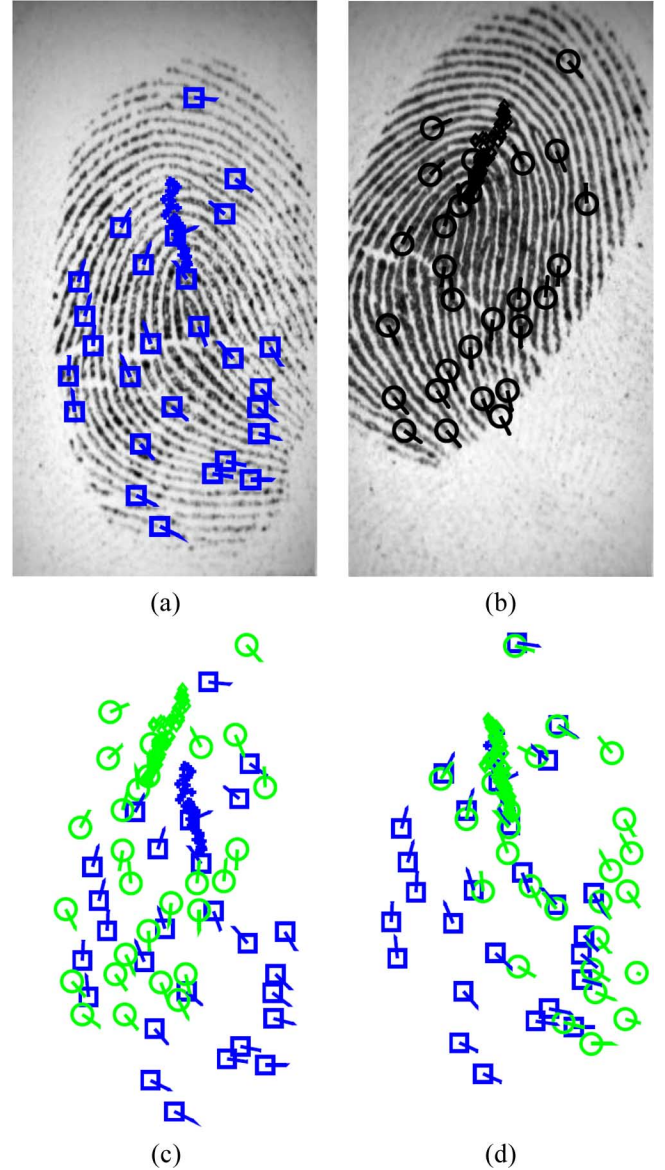

(a)   (b)

(c)   (d)

Fig. 8. Example of successful minutiae alignment based on helper data and the ICP algorithm. (a) Template image with minutiae and helper data. (b) Query image with minutiae and helper data. (c) Template and overlaid query minutiae prior to alignment. (d) Template and overlaid query minutiae after alignment. In this figure, the template minutiae are represented as squares (tails indicate the minutia direction) and the query minutiae are represented as circles. The template and query helper data points are represented as asterisks and diamonds, respectively.

available for each finger in FVC2002-DB2, we use only four impressions (impressions 1, 2, 7, and 8) in our experiments.[4]

The parameters used in our implementation for the two databases are listed in Table II. The choice of polynomial degree (n) is related to the size of the secret to be secured. For example, if $n = 8$, we can secure a key of size 128 b. Since the vault decoding is successful if $(n + 1)$ query minutiae match

[4]It is quite reasonable to assume that users in a biometric cryptosystem are cooperative and they are willing to provide good quality biometric data in order to retrieve their cryptographic keys. Impressions 3, 4, 5, and 6 in FVC2002 databases were obtained by requesting the users to provide fingerprints with exaggerated displacement and rotation. Hence, these impressions are not representative for the application under consideration. This explains our choice of impressions 1, 2, 7, and 8.

TABLE I
SUMMARY OF DATABASES USED IN OUR EXPERIMENTS

	FVC2002-DB2	MSU-DBI
No. of fingers	100	160
No. of impressions/finger	8	4
Sensor	Biometrika FX2000 (Optical)	Digital Biometrics, Inc. (Optical)
Image size	560×296 at 569 dpi resolution	640×480 at 500 dpi resolution
Image quality	Good	Medium

TABLE II
PARAMETERS USED FOR FUZZY VAULT IMPLEMENTATION

Parameter	FVC2002-DB2	MSU-DBI
No. of genuine points in the vault, r	18-24	24-36
Degree of encoding polynomial, n	7-10	10-12
Total no. of points in the vault, t	224	336
No. of chaff points in the vault, s	200-206	300-312
Minimum distance between selected minutiae, δ_1	25	25
Maximum distance between a query minutia and points selected by the coarse filter, δ_2	30	40

with the template minutiae, the parameter n also affects the error rates. Since the number of minutiae varies for different users, using a fixed value of r (the number of genuine minutiae points in the vault) across all users leads to several FTC errors. To overcome this problem, we fix the range of r and determine its value individually for each user. The number of chaff points in the vault (s) is chosen to be approximately ten times the number of genuine points in the vault which is a reasonable tradeoff between the complexity of a brute force attack and storage requirements of the vault. The number of bits used for encoding the minutia attributes u, v, and θ into the field $\mathcal{F} = GF(2^{16})$ are $\mathcal{B}_u = 6$, $\mathcal{B}_v = 5$, and $\mathcal{B}_\theta = 5$, respectively. The allocation of bits determines the quantization step size for u, v, and θ and it depends on the image size. For the selected databases, the aforementioned choice was found to be optimal in the sense that the distribution of the number of matching minutiae did not change significantly after quantization.

An example of successful vault operation for a user from FVC2002-DB2 when $n = 8$ is shown in Fig. 9. Fig. 9(f) shows that the ICP algorithm leads to the correct alignment of query minutiae with the template minutiae concealed in the vault. The coarse filter and minutiae matcher eliminate many chaff points from the vault. The unlocking set mostly consists of genuine points from the vault. For example, in Fig. 9(g), we observe that there are only two chaff points in the unlocking set. Since the number of genuine points in the unlocking set is more than 9, the decoding is successful in this example.

The criteria used for evaluating the performance are the failure-to-capture rate (FTCR), genuine accept rate (GAR), and false accept rate (FAR). When the number of well-separated minutiae in the template and/or query fingerprint are less than r, it results in failure to capture. The GAR is defined as the percentage of attempts made by genuine users that resulted in successful authentication. Since a vault is constructed for each finger, the number of genuine attempts is 100 and 160

for the FVC and MSU databases, respectively. The FAR is the percentage of attempts made by impostors that resulted in successful decoding of a vault corresponding to a legitimate user. Impostor attempts were simulated by trying to decode a user's vault using impressions from all of the other users. The number of impostor attempts is 9900 and 25 440 for the FVC and MSU databases, respectively.

The first row of Table III shows the performance of the proposed vault implementation on the FVC2002-DB2 database for different key sizes when a single impression is used for encoding and decoding (impression 1 is used for encoding and impression 2 is used for decoding). For example, when the key size is 128 b $(n = 8)$, 91 out of 100 genuine attempts were successful. Among the nine failed attempts, two were due to the lack of sufficient number of minutiae in the template (FTC error). So, only seven false rejects were actually encountered. For the same experiment, the fuzzy vault implemented in [25] was successful only in 61 out of 100 attempts with an FTCR of 16%. One reason for the high FTCR in [25] is the error in extraction of high curvature points. If the helper data extraction and alignment steps in the implementation of [25] are replaced with the ones proposed in this paper, the FTCR can be reduced to 2% and the GAR can be improved to 74%. This shows that the proposed helper data extraction and alignment algorithms are more robust compared to those presented in [25]. The selection of reliable minutiae based on image quality and use of a minutiae matcher to account for nonlinear distortion contribute to further improvement in the GAR from 74% to 91%. The net improvement in the GAR achieved by the proposed implementation over [25] is 30%.

In the case of MSU–DBI database, using a single impression for encoding and decoding results in a higher FTCR of 5.6% and a lower GAR of 82.5% for $n = 11$ (see the first row of Table IV). This decrease in performance is due to the lower quality of images in the MSU database compared to FVC2002-DB2. However, the average number of matching minutiae in the MSU database is higher than in FVC2002-DB2, which allows us to accommodate a larger key size.

The proposed alignment technique based on high curvature points also performs better than the registration based on the core point. Since it is difficult to determine the core point reliably, alignment based on core points leads to larger false rejects and FTC errors. For example, when core-point-based alignment[5] is used (instead of high curvature points) in the proposed vault implementation, the FTCR increases from 2% to 6% in the FVC database and from 5.6% to 15.6% in the MSU database. The reasons for an increase in FTCR are 1) no core point is present in some of the images (e.g., arch fingerprints) and 2) the algorithm fails to find the core point in some images (e.g., images where the loops are not very prominent). These are well-known problems in core-point detection. Furthermore, errors in determining the exact location and direction of the core point lead to a reduction in the GAR. The GAR decreases from 91% to 81% $(n = 8)$ and from 82.5% to 77.5% $(n = 11)$ in the FVC and MSU databases, respectively. These results clearly demonstrate the merits of using alignment based on high curvature points compared to core-based alignment.

---

[5]The core point was detected using commercial-off-the-shelf fingerprint recognition software from Neurotechnologija.
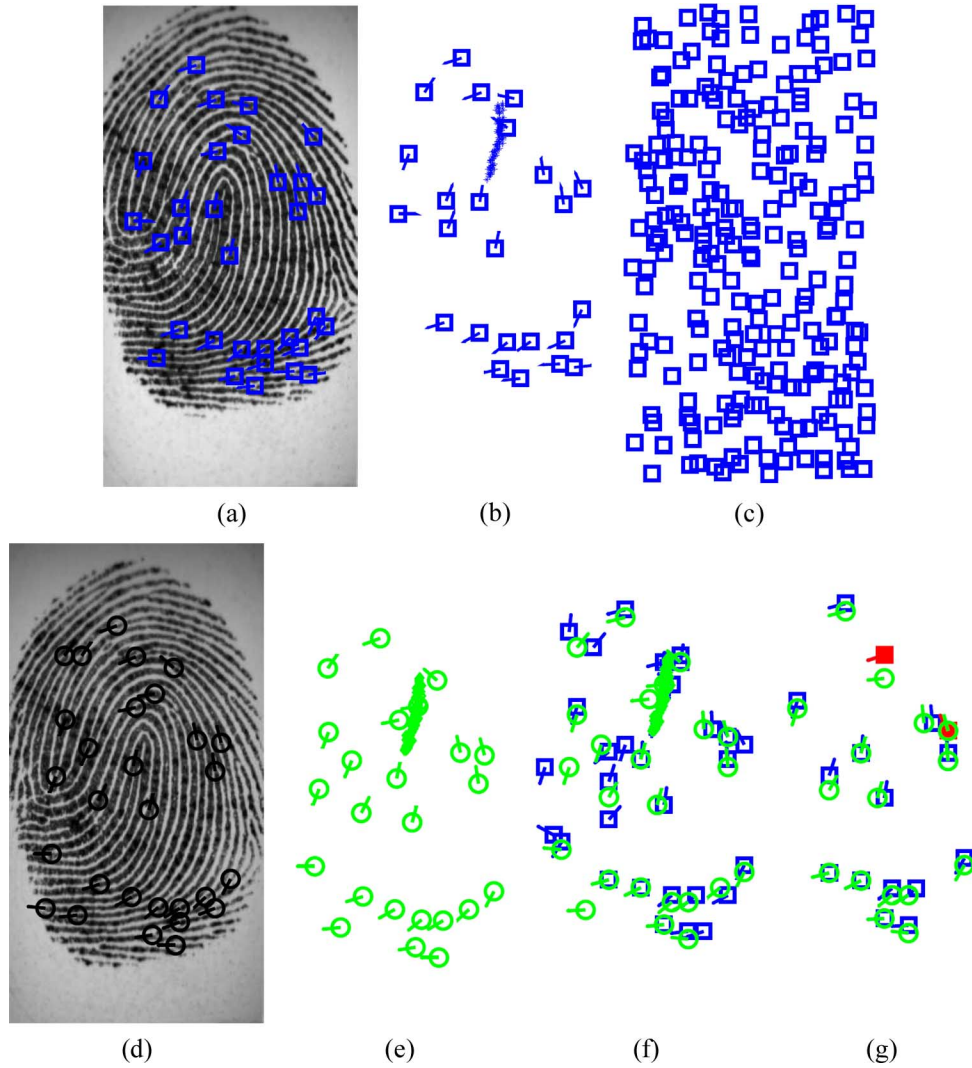
Fig. 9. Example of successful operation of the fuzzy vault. (a) Template fingerprint image with minutiae. (b) Selected template minutiae and helper data. (c) Vault in which the selected template minutiae are hidden among chaff points (for clarity, minutiae directions are not shown). (d) Query fingerprint image with minutiae. (e) Selected query minutiae and helper data. (f) ICP alignment of template and query helper data sets and coarse filtering of chaff points. (g) Unlocking set obtained by applying a minutiae matcher which eliminates almost all of the chaff points. The two points shown in filled squares in (g) are the only chaff points that remain in the unlocking set. Here, figures (a)–(c) represent vault encoding and (d)–(g) represent decoding.

TABLE III
PERFORMANCE SUMMARY OF THE PROPOSED FUZZY VAULT IMPLEMENTATION FOR FVC2002-DB2 DATABASE. HERE, n DENOTES THE DEGREE OF THE ENCODING POLYNOMIAL AND THE MAXIMUM KEY SIZE THAT CAN BE SECURED IS $16n$ BITS

Scenario	FTCR (%)	$n = 7$		$n = 8$		$n = 10$	
		GAR (%)	FAR (%)	GAR (%)	FAR (%)	GAR (%)	FAR (%)
1 Template, 1 Query	2	91	0.13	91	0.01	86	0
Mosaiced Template, 1 Query	1	95	0.12	94	0.02	88	0
Mosaiced Template, 2 Queries	1	97	0.24	96	0.04	90	0

TABLE IV
PERFORMANCE SUMMARY OF THE PROPOSED FUZZY VAULT IMPLEMENTATION FOR THE MSU–DBI DATABASE

Scenario	FTCR (%)	$n = 10$		$n = 11$		$n = 12$	
		GAR (%)	FAR (%)	GAR (%)	FAR (%)	GAR (%)	FAR (%)
1 Template, 1 Query	5.6	85	0.08	82.5	0.02	78.8	0
Mosaiced Template, 1 Query	2.5	88.1	0.09	83.1	0.02	81.2	0
Mosaiced Template, 2 Queries	0	96.9	0.16	92.5	0.03	87.5	0

One way to improve the performance of the vault is to use multiple impressions (templates) from the same finger during enrollment. However, we cannot create a vault for each enrolled image because an attacker can compare the multiple vaults and identify the chaff points. Therefore, we obtain a single mosaiced template from two impressions and use the mosaiced minutiae
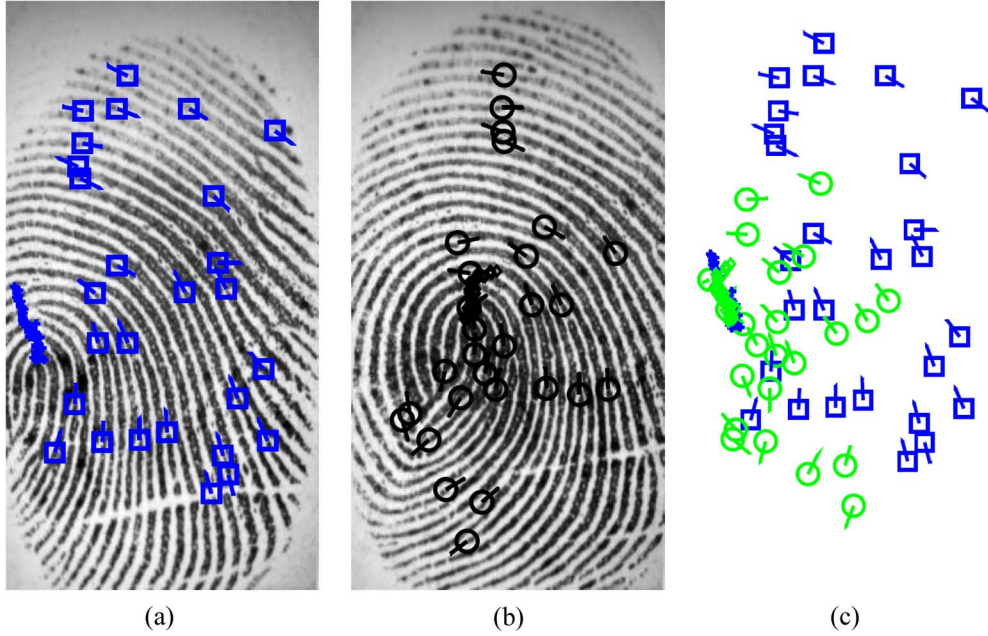
Fig. 10. Failure due to incorrect helper data extraction. (a) Template fingerprint image with minutiae and helper data. (b) Query fingerprint image with minutiae and helper data. (c) ICP alignment of template and query helper data sets along with aligned template and query minutiae. Helper data are incorrect in the template because the high curvature region is near the boundary.

to construct the vault. From row 2 of Table III, we observe that mosaicing reduces the FTCR from 2% to 1% and also increases the GAR of the system for all values of n. The performance can be further improved by using multiple queries during authentication. In case of a 128-b key size ($n = 8$) for the FVC2002-DB2 database, a mosaiced template leads to a GAR of 94% and using two queries instead of one query increases the GAR to 96%. The use of multiple impressions also leads to a significant reduction of FTCR and an increase in GAR for the MSU-DBI database (see rows 2 and 3 of Table IV).

The false rejects in our experiments were either due to errors in helper data extraction or due to an insufficient number of matching minutiae in the overlapping region between the template and query. Fig. 10 shows an example where the false reject is due to incorrect helper data extraction. In this case, the helper data for template fingerprint are inaccurate because the region of high curvature (core region) is close to the image boundary [see Fig. 10(a)]. An example of failure due to an insufficient number of overlapping minutiae is presented in Fig. 11. While the alignment between the template and query images in Fig. 11 is accurate, there are only five matching minutiae. This leads to a false reject because at least nine genuine minutiae must be identified in the vault for successful decoding.

The FAR of the proposed fuzzy vault implementation is nonzero for smaller values of n. In the single impression scenario for FVC2002-DB2, when $n = 8$, we observed one false accept in 9900 impostor attempts. The template and query fingerprint pair, which gives rise to a false accept, is shown in Fig. 12. Analysis of this false accept example indicates that there is indeed a set of nine minutiae in the query which matches with the template minutiae in both location and direction [see Fig. 12(c)]. Since the vault decoding is successful if $(n + 1)$ points in the query minutiae set (of size r) match with the template minutiae, the genuine accept and FARs vary with

n when r is fixed. Reducing n increases both GAR and FAR and increasing n lowers both GAR and FAR. As observed from Table III, FAR is high when $n = 7$ and is zero when $n = 10$. We also observe a marginal decrease in the GAR when n is increased from 7 to 10. The FAR for the MSU–DBI database also shows similar behavior.

As pointed out earlier, a drawback of the modified fuzzy vault scheme in [24] compared to the original scheme in [7] is the need to verify multiple candidate secrets. In [24], it was reported that an average of 201 candidate secrets was evaluated corresponding to 52 s of computation in Matlab with a 3.4-GHz processor system. In our implementation, the use of minutiae orientation in addition to the location eliminates almost all of the chaff points from the unlocking set. Therefore, the median number of candidate secrets that need to be evaluated is only 2 (the mean is 33) and the median decoding time is 3 s (mean is 8 s) on a similar processor.

VI. SECURITY OF PROPOSED FUZZY VAULT IMPLEMENTATION

The security provided by the fuzzy vault scheme has been studied theoretically in [7], [12], and [27]. Here, we consider three possible ways in which the proposed fingerprint-based fuzzy vault implementation can be circumvented in practice. Suppose an attacker attempts a brute-force attack on the proposed system by trying to decode the secret using all combinations of $(n + 1)$ points in the vault. If the secret size is 128 b and the number of genuine and chaff points in the vault are 24 and 200, respectively, the total number of possible combinations is $\binom{224}{9} \approx 3.3 \times 10^{15}$ and among these combinations, $\binom{24}{9} \approx 1.3 \times 10^6$ combinations will successfully decode the secret. The probability that a combination of points decodes the secret is $((1.3 \times 10^6)/(3.3 \times 10^{15})) \approx 4 \times 10^{-10}$ and the expected number of combinations that need to be evaluated is
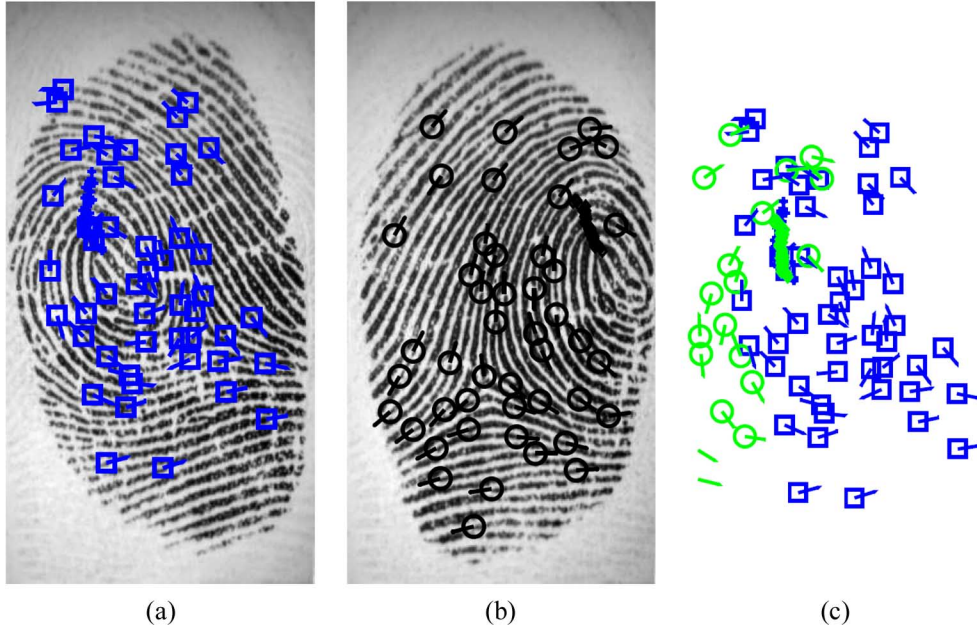
Fig. 11. Failure due to partial overlap. (a) Template fingerprint image with minutiae and helper data. (b) Query fingerprint image with minutiae and helper data. (c) ICP alignment of template and query helper data sets along with aligned template and query minutiae. Though the alignment is accurate, there are only few matching minutiae in these two images.
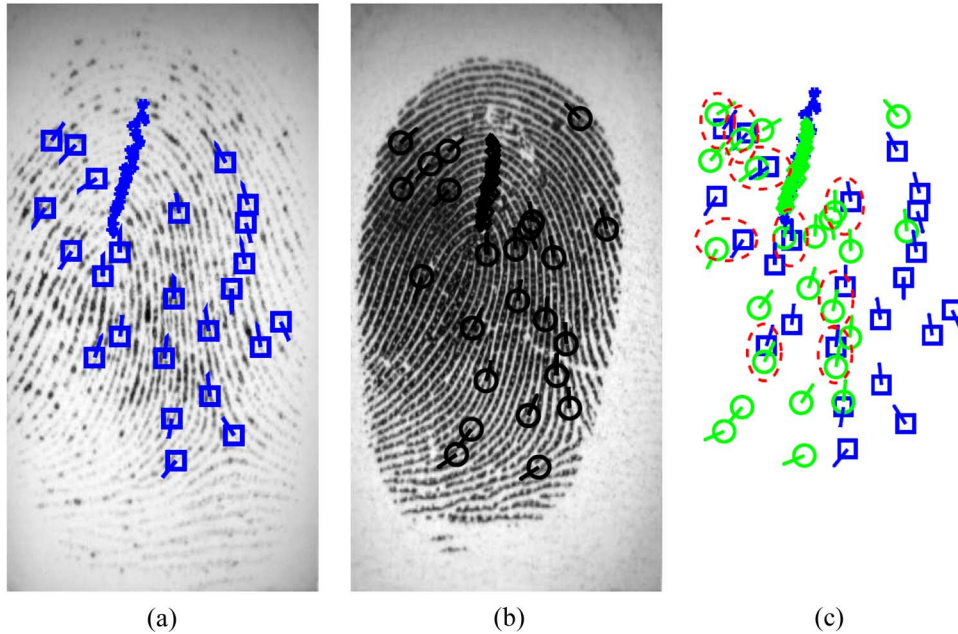

Fig. 12. Example of false accept when $n = 8$. (a) Template fingerprint image with minutiae and helper data. (b) Query fingerprint image with minutiae and helper data. (c) ICP alignment of template and query helper data sets along with aligned template and query minutiae. In (c), we observe that there are nine matching minutiae between the query and the template (represented as dotted ellipses).

2.5×10^9. This corresponds to a computational time of 13 years based on our current implementation.

The second method of circumvention involves the extraction of information from the helper data. We have already pointed out that high curvature points do not reveal any information about the minutiae and it is not possible to estimate the orientation field using only the high curvature points. However, suppose a smart attacker is able to extract the orientation field from the helper data and uses it to identify the chaff points. We can still defend against such an attack by adding a sufficient number of

chaff points that are consistent with the orientation field (i.e., the location of the chaff minutia is random, but its direction is determined by the orientation field) in addition to the completely random chaff points.

In our implementation, we assume that the spatial distribution of minutiae in a fingerprint image is uniform and use this property in the generation of chaff points. However, this assumption is not true because minutiae often tend to occur in clusters. Hence, the attacker may use statistical models for the minutiae distribution to classify the points in the vault as genuine and

chaff and attempt a brute-force attack using only the perceived genuine points. One way to defend against such attacks is to use statistical distribution of minutiae for the generation of chaff points during vault construction. Another solution is to use alternate fuzzy constructions that avoid the need for chaff points, such as the one proposed in [12]. However, such schemes usually require more robust techniques for alignment and nonlinear distortion correction, and their implementation based on fingerprint minutiae does not appear to be practical.

VII. Summary and Future Work

Given the rising magnitude of identity theft in our society, it is imperative to have reliable security mechanisms to protect information systems. Although cryptography is a powerful tool to achieve information security, the security of cryptosystems relies on the fact that cryptographic keys are secret and known only to the legitimate user. Biometric systems are being widely used to achieve reliable user authentication and these systems will proliferate into the core information infrastructure of the (near) future. When this happens, it is crucial that biometric authentication is secure. Fuzzy vault is one of the most comprehensive mechanisms for secure biometric authentication and cryptographic key protection. We have implemented a fully automatic and practical fuzzy vault system based on fingerprint minutiae that can easily secure secrets such as 128-b AES encryption keys. The main challenge in the implementation of a fingerprint-based fuzzy vault is the alignment of the query with the transformed template stored in the vault. We use helper data derived from the orientation field to align the template and query minutiae sets without leaking any information about the minutiae. Moreover, we have shown that applying a minutiae matcher during decoding can effectively compensate for nonlinear distortion, resulting in a significant improvement in the genuine accept rate. Evaluation on a public-domain fingerprint database demonstrates that our implementation has the highest GAR and a very low FAR among the known implementations of a fingerprint-based fuzzy vault.

The performance of the fuzzy vault can be further improved by using multiple biometric sources, such as multiple fingers or multiple modalities (e.g., fingerprint and iris). Apart from the location and orientation attributes of a minutia point, many minutiae-based fingerprint matchers use additional attributes, such as minutia type, ridge counts, ridge curvature, and ridge density to achieve high recognition rates. These attributes could also be incorporated into the fuzzy vault framework. The addition of new attributes will not only increase the number of possible chaff points that can added to the vault but also decrease the decoding complexity for genuine users and reduce the FAR. A well-known limitation of the fuzzy vault framework is its dependence on chaff points to achieve security. Therefore, fuzzy constructions that do not involve chaff points could be considered.

References

[1] W. Stallings, *Cryptography and Network Security: Principles and Practices*. Upper Saddle River, NJ: Prentice-Hall, 2006.

[2] RSA Laboratories, PKCS #5: Password-Based Cryptography Standard, Version 2.0 RSA, Mar. 1999, Tech. Rep.

[3] K. Mitnick, W. Simon, and S. Wozniak, *The Art of Deception: Controlling the Human Element of Security*. New York: Wiley, 2002.

[4] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 1, pp. 4–20, Jan. 2004.

[5] N. K. Ratha, J. H. Connell, and R. M. Bolle, "An analysis of minutiae matching strength," in *Proc. Audio- and Video-Based Biometric Person Authentication*, Halmstad, Sweden, Jun. 2001, pp. 223–228.

[6] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. New York: Springer-Verlag, 2003.

[7] A. Juels and M. Sudan, "A fuzzy vault scheme," in *Proc. IEEE Int. Symp. Inform. Theory*, Lausanne, Switzerland, 2002, p. 408.

[8] G. I. Davida, Y. Frankel, and B. J. Matt, "On enabling secure applications through off-line biometric identification," in *Proc. IEEE Symp. Security and Privacy*, Oakland, CA, May 1998, pp. 148–157.

[9] C. Soutar, D. Roberge, A. Stoianov, R. Gilroy, and B. V. K. V. Kumar, "Biometric encrpytion," in *ICSA Guide Cryptograp.*, R. K. Nichols, Ed. New York: McGraw-Hill, 1999.

[10] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel, "Cryptographic key generation from voice," in *Proc. IEEE Symp. Security and Privacy*, Oakland, CA, May 2001, pp. 202–213.

[11] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proc. 6th ACM Conf. Computer and Communications Security*, Singapore, Nov. 1999, pp. 28–36.

[12] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," Cryptology ePrint Archive, Tech. Rep. 235, Feb. 2006.

[13] F. Hao, R. Anderson, and J. Daugman, "Combining crypto with biometrics effectively," *IEEE Trans. Comput.*, vol. 55, no. 9, pp. 1081–1088, Sep. 2006.

[14] P. Tuyls, A. H. M. Akkermans, T. A. M. Kevenaar, G.-J. Schrijen, A. M. Bazen, and R. N. J. Veldhuis, "Practical biometric authentication with template protection," in *Proc. AVBPA*, Rye Town, NY, Jul. 2005, pp. 436–446.

[15] E. Martinian, S. Yekhanin, and J. Yedidia, "Secure biometrics via syndromes," Mitsubishi Elect. Res. Labs, Tech. Rep. TR2005-112, Dec. 2005.

[16] N. K. Ratha, J. H. Connell, and R. M. Bolle, "Enhancing security and privacy in biometric authentication systems," *IBM Syst. J.*, vol. 40, no. 3, pp. 614–634, 2001.

[17] U. Uludag, S. Pankanti, S. Prabhakar, and A. K. Jain, "Biometric cryptosystems: Issues and challenges," *Proc. IEEE (Special Issue Multimedia Security for Digital Rights Management)*, vol. 92, no. 6, pp. 948–960, Jun. 2004.

[18] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.

[19] T. Clancy, D. Lin, and N. Kiyavash, "Secure smartcard-based fingerprint authentication," in *Proc. ACM SIGMM Workshop on Biometric Methods and Applications*, Berkley, CA, Nov. 2003, pp. 45–52.

[20] S. Yang and I. Verbauwhede, "Automatic secure fingerprint verification system based on fuzzy vault scheme," in *Proc. IEEE ICASSP*, Philadelphia, PA, Mar. 2005, vol. 5, pp. 609–612.

[21] Y. Chung, D. Moon, S. Lee, S. Jung, T. Kim, and D. Ahn, "Automatic alignment of fingerprint features for fuzzy fingerprint vault," in *Proc. Conf. Information Security Cryptology*, Beijing, China, Dec. 2005, pp. 358–369.

[22] Y. C. Feng and P. C. Yuen, "Protecting face biometric data on smartcard with Reed-Solomon code," in *Proc. CVPR Workshop Privacy Research in Vision*, New York, Jun. 2006, p. 29.

[23] M. Freire-Santos, J. Fierrez-Aguilar, and J. Ortega-Garcia, "Cryptographic key generation using handwritten signature," in *Proc. Biometric Technologies for Human Identification III*, Orlando, FL, Apr. 2006, vol. 6202, pp. 225–231.

[24] U. Uludag, S. Pankanti, and A. K. Jain, "Fuzzy vault for fingerprints," in *Proc. Audio- and Video-Based Biometric Person Authentication*, Rye Town, NY, Jul. 2005, pp. 310–319.

[25] U. Uludag and A. K. Jain, "Securing fingerprint template: Fuzzy vault with helper data," in *Proc. CVPR Workshop Privacy Research Vision*, New York, Jun. 2006, p. 163.

[26] A. K. Jain, L. Hong, and R. Bolle, "On-line fingerprint verification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 4, pp. 302–314, Apr. 1997.

[27] E.-C. Chang, R. Shen, and F. W. Teo, "Finding the original point set hidden among chaff," in *Proc. ACM Symp. Information, Computer Communications Security*, Taipei, Taiwan, R.O.C., 2006, pp. 182–188.

[28] Y. Chen, S. C. Dass, and A. K. Jain, "Fingerprint quality indices for predicting authentication performance," in *Proc. AVBPA*, Rye Brook, NY, Jul. 2005, pp. 160–170.

[29] N. K. Ratha, S. Chikkerur, J. H. Connell, and R. M. Bolle, "Generating cancelable fingerprint templates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 561–572, Apr. 2007.

[30] K. Nandakumar, A. K. Jain, and S. Pankanti, "Fingerprint-based fuzzy vault: implementation and performance," Michigan State Univ., East Lansing, Tech. Rep. TR-06-31., 2006. [Online]. Available: http://www.cse.msu.edu/cgi-user/web/tech/document?ID=675.

[31] S. C. Dass and A. K. Jain, "Fingerprint classification using orientation field flow curves," in *Proc. Indian Conf. Computer Vision, Graphics and Image Processing*, Kolkata, India, Dec. 2004, pp. 650–655.

[32] S. C. Dass, "Markov random field models for directional field and singularity extraction in fingerprint images," *IEEE Trans. Image Process.*, vol. 13, no. 10, pp. 1358–1367, Oct. 2004.

[33] P. Besl and N. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[34] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *Proc. Int. Conf. Pattern Recognition*, Quebec City, QC, Canada, Aug. 2002, pp. 545–548.

[35] D. Maio, D. Maltoni, J. L. Wayman, and A. K. Jain, "FVC2002: Second fingerprint verification competition," in *Proc. Int. Conf. Pattern Recognition*, Quebec City, QC, Canada, Aug. 2002, pp. 811–814.

[36] A. K. Jain, S. Prabhakar, and A. Ross, "Fingerprint matching: Data acquisition and performance evaluation," Michigan State Univ., East Lansing, MI, Tech. Rep. TR99-14, 1999.

[37] A. Ross, S. Shah, and J. Shah, "Image versus feature mosaicing: A case study in fingerprints," in *Proc. SPIE Conf. Biometric Technology for Human Identification*, Orlando, FL, Apr. 2006, vol. 6202, pp. 1–12.

Anil K. Jain (F'91) received the B.Tech. degree from the Indian Institute of Technology, Kanpur, India, and the M.S. and Ph.D. degrees from The Ohio State University, Columbus, in 1970 and 1973, respectively.

He is a University Distinguished Professor in the Department of Computer Science and Engineering at Michigan State University, East Lansing. He received the distinguished alumni award from The Ohio State University. His research interests include statistical pattern recognition, data clustering, and biometric authentication.

Dr. Jain is a Fellow of the ACM, IAPR, SPIE, and AAAS. He received a Fulbright Research Award, a Guggenheim Fellowship, the Alexander von Humboldt Research Award, and the 2003 IEEE Computer Society Technical Achievement Award. He holds six patents in the area of fingerprint matching and he is the author of a number of books, including the *Handbook of Multibiometrics* (Springer, 2006); *Biometric Systems, Technology, Design, and Performance Evaluation* (Springer, 2005); *Handbook of Face Recognition* (Springer, 2005); *Handbook of Fingerprint Recognition* (Springer, 2003), which received the PSP award from the Association of American Publishers; *BIOMETRICS: Personal Identification in Networked Society* (Kluwer, 1999); and *Algorithms for Clustering Data* (Prentice-Hall, 1988). ISI has designated him as a highly cited researcher. He is an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and *ACM Transactions on Knowledge Discovery in Data*. He received the 1996 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award and best paper awards from the Pattern Recognition Society in 1987 and 1991. He was the editor-in-chief of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. He is a member of the National Academies panels on Whither Biometrics and Improvised Explosive Device.

Karthik Nandakumar (S'03) received the B.E. degree from Anna University, Chennai, India, in 2002 and the M.S. degree in computer science and statistics from Michigan State University, East Lansing, in 2005 and 2007, respectively, where he is currently pursuing the Ph.D. degree in computer science and engineering.

His research interests include statistical pattern recognition, biometric authentication, image processing, and computer vision. He has coauthored a book titled *Handbook of Multibiometrics* (Springer, 2006).

Sharath Pankanti (SM'98) received the Ph.D. degree in computer science from Michigan State University, East Lansing, in 1995. His Ph.D. dissertation topic was the methods of integrating computer vision modules.

He joined IBM T. J. Watson Research Center, Yorktown Heights, NY, in 1995 and was with the IBM Advanced Identification Project until 1999. During 2000–2001, he worked on "footprints"—a system for tracking people based on their infrared emission. From 2001 to 2003, he worked on PeopleVision, a system for detecting and tracking individuals in indoor and outdoor environments. From 2003 to 2004, he worked on large-scale biometric indexing systems and since 2005, has worked on human interface designs for effective security and convenience. He has coedited a comprehensive book on biometrics *Biometrics: Personal Identification* (Kluwer, 1999) and coauthored *A Guide to Biometrics* (Springer, 2004).