1. Introduction

The current cryptographic systems (e.g., DES, AES, RSA) have high theoretical (related to the complexity of the underlying building blocks in their construction) and proven security for *containing* the plain text data. Nevertheless, illegal key sharing (key management problem) is one of their major drawbacks: regardless of the security of the algorithms, if the keys that need to be known *only* to the legitimate parties in the communication are shared freely, it is trivial to convert cipher text back to plain text. Another limitation of the cryptographic systems is that they require the keys to be very long and random for high security. For example, AES requires at least 128-bits (which corresponds to a 19 character key from a 7-bit ASCII code). This makes it impossible for users to remember the keys. As a result, the cryptographic keys are stored within a physical medium (e.g., in a computer or on a smart card) and release based on some alternative authentication mechanism. If this mechanism succeeds, the released key can be used in encryption/decryption procedures. The most popular authentication mechanism used for this purpose is based on passwords, which are again cryptographic key-like strings but they are simple enough for users to remember (hence it is not necessary for users to store this information within a physical medium). But, passwords can be hacked by various forms of attacks and are not considered to be very safe. Furthermore, passwords are unable to provide non-repudiation: a subject may deny releasing the key using password authentication, claiming that her password was stolen and that a thief released the key. Many of the above limitations of password-based authentication can be eliminated by incorporating biometric authentication into the cryptographic system.

Biometrics plays an important role in taking care of all these flaws. It completely depicts your identity and eliminates the case of non-repudiation. And also you don't have to remember long passwords. The fingerprint based fuzzy vault scheme secures the cryptographic key by encoding it the help of fingerprint minutiae [16]. The project consists of two phases: (i) Feature extraction from fingerprint and, (ii) Alignment and fuzzy vault implementation. Our implementation improves the existing ones as it doesn't need any helper data for aligning the fingerprints.

1.1 Motivation

An improved fingerprint based fuzzy vault implementation has been proposed in this thesis. Securing the key is one of the major challenges of secret key cryptographic systems. Biometrics is one of the most reliable ways to accomplish this and fingerprint is the most widely used biometric feature. At the same time, we need to ensure that the features of the fingerprint are not exposed while sending the key encrypted with fingerprint. Fuzzy vault [18] can be used to accomplish this. We don't have to find the exact match to unlock the vault as it is very difficult to exactly match two fingerprints.

Biometrics is being widely used in many places for authentication. The noisy nature of fingerprints due to dirt, sweat, physical wear and tear etc poses a lot of challenges in extracting the minutiae and the singularity points (core and delta). Although a lot of research has been done in this field, but there still is a lot of scope for improving the fingerprint image enhancement techniques and methods to find the singularity points. The existing fingerprint matching algorithms don't need very strict

alignment of the fingerprint images. But, for fuzzy vault scheme we need to have an accurate alignment of the query and template minutiae.

The biometric cryptosystems is a very interesting field of research. The biometric features are extracted and transformed to some other form so that they can be used in cryptography. In fuzzy vault scheme, the secret key is broken down to form a polynomial which is then evaluated over the biometric features and are sent through unsecured channel after adding CRC bits. Lagrange interpolation is used to reconstruct the polynomial and thus retrieve the secret key. Thus, the research work encompasses image processing, cryptography, mathematics and computation, making it a much diversified challenge.

1.2 Related Work

A lot of work has been done in the field of biometric cryptosystems. Soutar et al. [43, 44] proposed a “key binding" algorithm for an optical correlation- based fingerprint matching system. This algorithm binds a cryptographic key with the user's fingerprint image at the time of enrollment. The key is then retrieved only upon successful authentication. The main criticism of Soutar et al.'s work in the literature [5] is that the method does not carry rigorous security guarantees. The authors do not explain how much entropy is lost at each stage of their algorithm. Further, the resulting FAR and FRR values associated with key re-lease are unknown. The authors also assume that the input and database fingerprint images are perfectly aligned. Even with a very constrained image acquisition system, it is unrealistic to acquire fingerprint images without any misalignment. Monrose et al. [45] proposed a method to make passwords more secure by combining keystroke biometrics with passwords. Their

technique was inspired by password “salting", where a user's password (*pwd*) is salted by prepending it with an *s*-bit random number (the “salt"), resulting in a hardened password (*hpwd*). In their “fuzzy commitment" scheme [6], Juels and Wattenberg generalized and significantly improved Davida et al.'s methods [46] to tolerate more intra-class variation in the biometric characteristics and to provide stronger security. The authors acknowledge that one of the major shortcomings of the fuzzy commitment scheme is that it requires the biometric representations *X* and *Y* to be ordered so that their correspondence is obvious.

Juels and Sudan's fuzzy vault scheme [5] is an improvement upon previous work by Juels and Wattenberg [6]. Assume Alice is a legitimate user of this scheme, and Bob is an attacker. In [5] Alice can place a secret *K* (e.g., secret encryption key) in a vault and lock (secure) it using an unordered set *A*. Here, an unordered set means that the relative positions of set elements do not change the characteristics of the set: e.g., the set {-2,-1, 3} conveys the same information as {3,-1,-2}. Bob, using an unordered set *B*, can unlock the vault (access K) only if *B* overlaps with *A* to a large extent. The procedure for constructing the fuzzy vault is as follows: First, Alice selects a polynomial *p* of variable *x* that encodes K (e.g., by fixing the coefficients of *p* according to K). She computes the polynomial projections, *p*(*A*), for the elements of *A*. She adds some randomly generated chaff points that do not lie on *p*, to arrive at the final point set *R*. When Bob tries to learn K (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), it is assumed that he can reconstruct *p* (and hence K). A simple numerical example for this process is as follows: Assume Alice selects the polynomial $p(x) = x^2-3x+1$, where the coefficients (1, -3, 1)

encode her secret K. If her unordered set is A = {-1,-2, 3, 2}, she will obtain the polynomial projections as {(A, p(A))} = {(-1, 5), (-2, 11), (3, 1), (2,-1)}. To this set, assume Alice adds two chaff points C = {(0, 2), (1, 0)} that do not lie on p, to find the final point set R = {(-1, 5), (-2, 11), (3, 1), (2,-1), (0, 2), (1, 0)}. Now, if Bob can separate at least 3 points from R that lie on p, he can reconstruct p, hence decode the secret represented as the polynomial coefficients (1, -3, 1). Otherwise, he will end up with an incorrect p, and he will not be able to access the secret K. The security of the scheme is based on the infeasibility of the polynomial reconstruction problem (i.e., if Bob does not locate many points that lie on p, he cannot feasibly find the parameters of p, hence he cannot access K). The scheme can tolerate some differences between the entities (unordered sets A and B) that lock and unlock the vault, so Juels and Sudan named their scheme *fuzzy* vault. The fuzzy vault scheme is expected to tolerate these intra-class variations. On the other hand, in traditional cryptography, if the keys are not exactly the same, the decryption operation will fail. Note that since the fuzzy vault can work with unordered sets (common in biometric templates, including fingerprint minutiae data), it is a promising candidate for biometric cryptosystems. Having said this, the fuzzy vault scheme requires *pre-aligned* biometric templates. Alignment is an essential requirement due to different types of distortion that can occur during biometric data acquisition. Further, the number of feasible operating points (where the vault operates with *negligible* complexity, e.g., measured via the number of required access attempts to reveal the secret for a genuine user and with *considerable* complexity for an imposter user) for the fuzzy vault is limited.

Dodis et al. [47] proposed theoretical foundations for generating keys from the

“key material" that is not exactly reproducible (e.g., passphrases, answers to questionnaires, biometric data that changes between enrollment and verification). They also propose a modification of the Juels and Sudan's fuzzy vault scheme [5]: instead of adding chaff points to the projections of the polynomial *p*, Dodis et al. [47] propose to use a polynomial p' (of degree higher than p) which overlaps with p only for the points from the genuine set *A*. The security of the scheme is based on the degree of this new polynomial p', which replaces the final point set *R* of Juels and Sudan's scheme [5]. Clancy et al. [48] propose a “fingerprint vault" based on the scheme of Juels and Sudan [5]. At the enrollment time, multiple (typically 5) fingerprints of users are acquired. The fingerprint representation (minutiae positions) is extracted from each fingerprint. Correspondence between feature points (minutiae) extracted from the multiple prints is established using a bounded nearest-neighbor algorithm. Yang and Verbauwhede [14] attempted to eliminate the pre-alignment requirement of Clancy et al.'s [48] algorithm. First, the enrollment fingerprint is analyzed to find a “reference minutia": it is defined as a minutia (i) that is present in all of the multiple (e.g., 3) fingerprint impressions, and (ii) whose local structure (based on its distance and orientation with respect to its two nearest-neighbor minutiae) does not change. The fingerprint that will lock the vault is registered with respect to this reference minutia (using the reference minutia as the origin of a new coordinate frame).

Uludag et al. [18] proposed a fuzzy vault for fingerprints without using the Reed Solomon encoding. Chung et al. [19] proposed an automatic alignment of fingerprint features for fuzzy fingerprint vault. While Sungji [20] gave a geometric hashing based technique for the fuzzy vault. Uludag et al. [20] modified their own scheme [18], to implement alignment with the help of helper data. Peng et al. [39]

proposed an alignment free system, while Xin et al. [40] proposed a topological structure based alignment.

1.3 Problem Statement

Finger print based fuzzy vault is a biometric cryptosystem which can be used to secure the secret key. The aim of this work is to study the various options available for implementing a fuzzy vault and propose an improved algorithm which doesn't require any helper data to be sent for aligning the fingerprints. The work consists of two major sections: (i) Biometric image enhancement and feature extraction, and (ii) Alignment of fingerprints and fuzzy vault implementation.

We justify the efficiency of our method by applying it to two different finger print databases. We also explore different possibilities and compare the experimental results obtained.

Our problem statement can be summarized as follows:

"To develop a fingerprint based fuzzy vault which doesn't require additional helper data for alignment"

1.4 Scope of the Work

In this project, a fuzzy vault scheme has been implemented and its efficiency is compared to the existing implementations. Our work involves a detailed study of the biometric (fingerprints) feature extraction and various key generation and key release schemes based on biometrics.

Our implementation of the fuzzy vault is tested on the publicly available fingerprint databases and some live samples taken by us. The accuracy of our implementation mainly depends on the alignment of the fingerprints. Broadly, the scope of this work can be summarized as:

- To perform fingerprint image enhancement and extract features from it.
- To extract the singularity points of the fingerprints accurately.
- To explore various ways for aligning the fingerprint templates.
- To improve the existing fuzzy vault implementation
- To implement a modified fuzzy vault which doesn't require helper data as we have developed a new method for finding the core point accurately.
- To compare the performance with the existing methods by using some standard databases [7,8] and some live data.

1.5 Organization of the Thesis

The remainder part of this thesis is organized in the following chapters:

Chapter 2: Biometrics

This section consists of the introduction to biometrics and various techniques available for enhancement and extraction of features. A complete literature survey of fingerprint feature extraction is presented.

Chapter 3: Biometric Cryptosystems

This chapter discusses about the cryptographic techniques and the need for securing the secret keys. Various options available are discussed in this section and an overview of the fuzzy vault is presented.

Chapter 4: Our Fuzzy vault implementation

This chapter discusses the step by step implementation of the entire system. The techniques used for biometric feature extraction, alignment and the fuzzy vault implementation details are discussed. The various options explored by us are discussed in this section.

Chapter 5: Experiments and Results

Here, we present the experimental results of our implementation and analyze and compare it with the other proposed methods.

Chapter 6: Conclusion and Future Scope

This chapter concludes the discussion in the thesis while opening up the new challenges that can be taken upon.

References: This section gives the reference details used for completing this work.

2. Biometrics

This chapter covers the biometrics and the way it can be used for authenticating people. The various enhancement techniques and feature extraction is discussed in this chapter. [9]

2.1 Introduction

Biometric recognition, or simply biometrics, refers to the use of distinctive anatomical and automatically recognizing a person. Questions such as “Is this person authorized to enter the facility?”, “Is this individual entitled to access the privileged information?”, and “Did this person previously apply for a passport?” are routinely asked in a variety of organizations in both person’s identity. Because biometric identifiers cannot be easily misplaced, forged, or shared, they are considered more reliable for person recognition than traditional token- (e.g., keys or ID cards) or knowledge- (e.g., password or PIN) based methods. Biometric recognition provides better security, higher efficiency, and, in many instances, increased user convenience. It is for these reasons that biometric recognition systems are being increasingly deployed in a large number of government (e.g., border crossing, national ID card, e-passports) and civilian (e.g., computer network logon, mobile phone, Web access, smartcard) applications.

A number of biometric technologies have been developed and several of them have been successfully deployed. Among these, fingerprints, face, iris, voice, and hand geometry are the ones that are most commonly used. Each biometric trait has its strengths and weaknesses and the choice of a particular trait typically depends on the requirements of the application. Various biometric identifiers can also be compared on

the following factors; universality, distinctiveness, permanence, collectability, performance, acceptability and circumvention. Because of the well-known distinctiveness (individuality) and persistence properties of fingerprints as well as cost and maturity of products, fingerprints are the most widely deployed biometric characteristics. It is generally believed that the pattern on each finger is unique. Given that there are about 6.5 billion living people on Earth and assuming each person has 10 fingers, there are 65 billion unique fingers! In fact, fingerprints and biometrics are often considered synonyms! Fingerprints were first introduced as a method for person identification over 100 years back. Now, every forensics and law enforcement agency worldwide routinely uses automatic fingerprint identification systems (AFIS). While law enforcement agencies were the earliest adopters of the fingerprint recognition technology, increasing concerns about national security, financial fraud and identity fraud have created a growing need for fingerprint technology for person recognition in a number of non-forensic applications.

Fingerprint recognition system can be viewed as a pattern recognition system. Designing algorithms capable of extracting salient features from fingerprints and matching them in a robust way are quite challenging problems. This is particularly so when the users are uncooperative, the finger surface is dirty or scarred and the resulting fingerprint image quality is poor. There is a popular misconception that automatic fingerprint recognition is a fully solved problem since automatic fingerprint systems have been around for almost 40 years. On the contrary, fingerprint recognition is still a challenging and important pattern recognition problem because of the large intra-class variability and large inter-class similarity in fingerprint patterns.

2.2 Fingerprint Analysis

A fingerprint is the reproduction of the exterior appearance of the fingertip epidermis. The most evident structural characteristic of a fingerprint is a pattern of interleaved *ridges* and *valleys* (Ashbaugh, 1999); in a fingerprint image, ridges (also called ridge lines) are dark whereas valleys are bright (see Figure 2.1). Ridges vary in width from 100 μm, for very thin ridges, to 300 μm for thick ridges. Generally, the period of a ridge/valley cycle is about 500 μm. Most injuries to a finger such as superficial burns, abrasions, or cuts do not affect the underlying ridge structure, and the original pattern is duplicated in any new skin that grows.

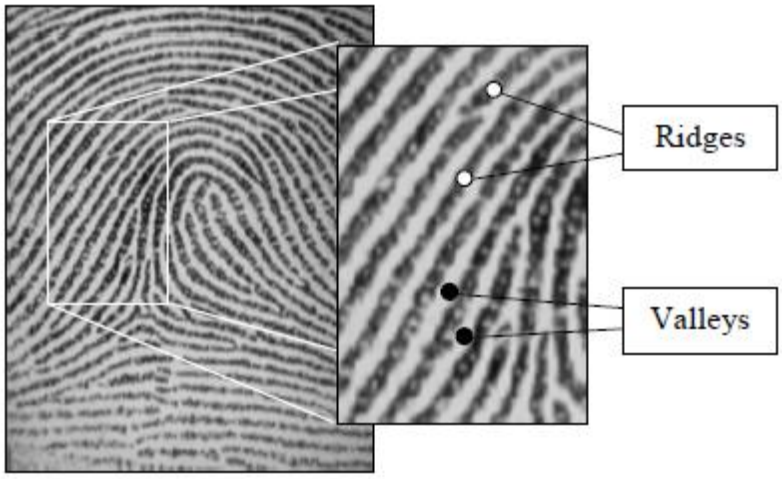


Figure 2.1. Ridges and valleys in a fingerprint image

Ridge details are generally described in a hierarchical order at three different levels, namely, Level 1 (the overall global ridge flow pattern), Level 2 (minutiae points), and Level 3 (pores, local shape of ridge edges, etc.).

At the global level (Level 1), ridges often run smoothly in parallel but exhibit one or more regions where they assume distinctive shapes (characterized by high curvature, frequent ridge terminations, etc.). These regions, called *singularities* or

singular regions, may be broadly classified into three typologies: *loop*, *delta*, and *whorl* (see Figure 2.2). Singular regions belonging to loop, delta, and whorl types are typically characterized by ∩, Δ, and O shapes, respectively. Sometimes whorl singularities are not explicitly introduced because a whorl type can be described in terms of two loop singularities facing each other.

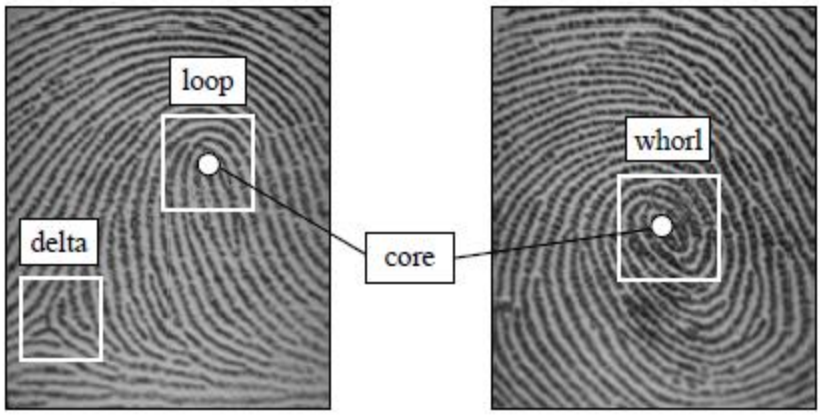


Figure 2.2. Singular regions (white boxes) and core points (small circles) in fingerprint images

Fingerprint matching algorithms can pre-align fingerprint images according to a landmark or a center point, called the *core*. Henry (1900) defined the core point as "the north most point of the innermost ridge line." In practice, the core point corresponds to the center of the north most loop type singularity. For fingerprints that do not contain loop or whorl singularities (e.g., those belonging to the Arch class in Figure 2.3), it is difficult to define the core. In these cases, the core is usually associated with the point of maximum ridge line curvature. Unfortunately, due to the high variability of fingerprint patterns, it is difficult to reliably locate a registration (core) point in all the fingerprint images. Singular regions are commonly used for fingerprint classification (see Figure 3.3), that is assigning a fingerprint to a class among a set of distinct classes, with the aim of simplifying search and retrieval.

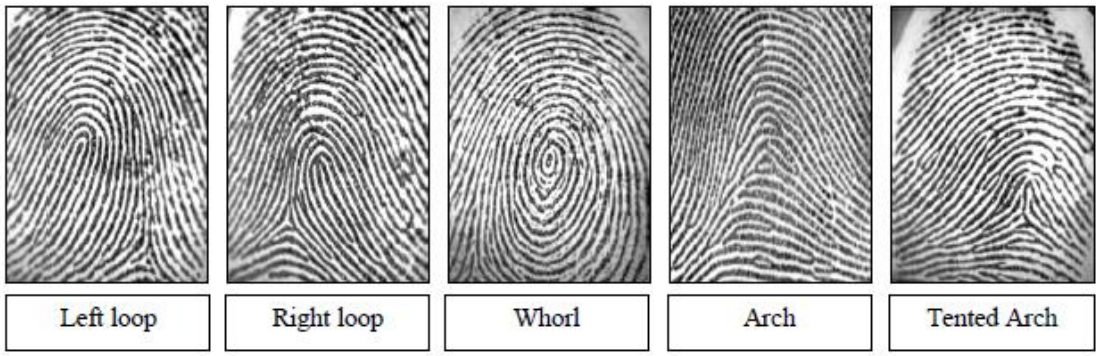


Figure 2.3. One fingerprint from each of the five major classes defined by Henry (1900).

At the local level (Level 2), other important features, called *minutiae* can be found in the fingerprint patterns. Minutia means small detail; in the context of fingerprints, it refers to various ways that the ridges can be discontinuous (see Figure 2.4). For example, a ridge can suddenly come to an end (ridge ending), or can divide into two ridges (bifurcation). Minutiae are the most commonly used features in automatic fingerprint matching. Although several types of minutiae can be considered (the most common types are shown in Figure 2.4), usually only a coarse minutiae classification is adopted to deal with the practical difficulty in automatically discerning the different types with high accuracy. The American National Standards Institute (ANSI/NIST-ITL 1, 2007) proposes a minutiae taxonomy based on four classes: *ridge ending*, *bifurcations*, *compound* (trifurcation or crossovers), and *type undetermined.*

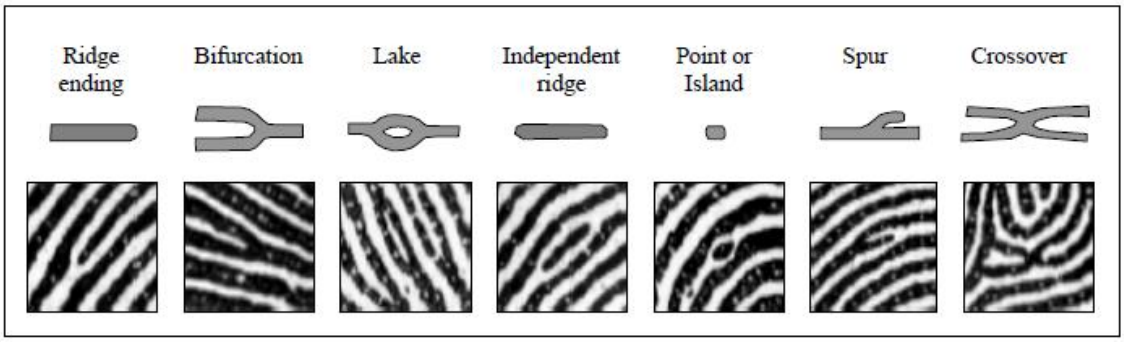


Figure 2.4. Seven most common minutiae types.

In ANSI/NIST-ITL 1 (2007) Type-9 records each minutia is denoted by its class, the *x*- and *y*-coordinates and the angle between the tangent to the ridge line at the minutia position and the horizontal axis (Figure 2.5). In practice, an ambiguity exists between ridge ending and bifurcation minutiae types; depending on the finger pressure against the surface where the fingerprint impression is formed, ridge endings may appear as bifurcations and vice versa. However, thanks to the convention used to define minutiae angle, there is no significant change in the angle if the minutia appears as a ridge ending in one impression and as a bifurcation in another impression of the same finger.

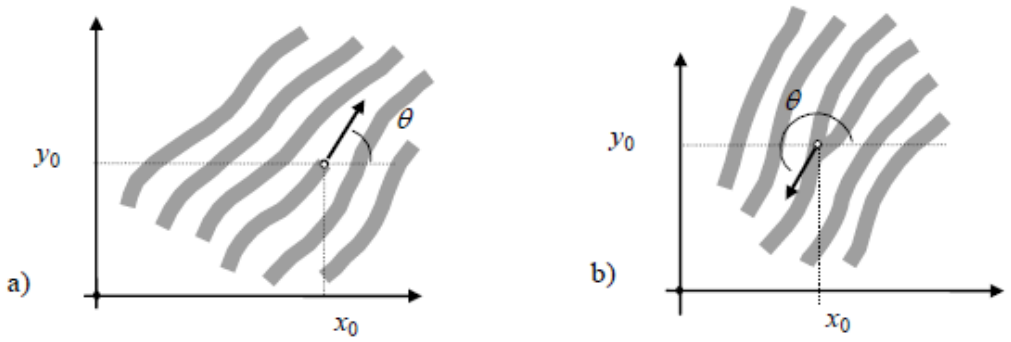


Figure 2.5. a) a ridge ending minutia: [x0,y0] are the minutia coordinates; θ is the angle that the minutia tangent forms with the horizontal axis; b) a bifurcation minutia: θ is now defined by means of the ridge ending minutia corresponding to the original bifurcation that exists in the negative image.

2.3 Fingerprint Enhancement Techniques

The performance of minutiae extraction algorithms and other fingerprint recognition techniques relies heavily on the quality of the input fingerprint images. In an ideal fingerprint image, ridges and valleys alternate and flow in a locally constant direction. In such situations, the ridges can be easily detected and minutiae can be precisely located in the image. Figure 2.6a shows an example of a good quality

fingerprint image. However, in practice, due to skin conditions (e.g., wet or dry, cuts, and bruises), sensor noise, incorrect finger pressure, and inherently low-quality fingers (e.g., elderly people, manual workers), a significant percentage of fingerprint images (approximately 10%, according to our experience) is of poor quality like those in Figures 2.6b, c. In many cases, a single fingerprint image contains regions of good, medium, and poor quality where the ridge pattern is very noisy and corrupted (Figure 2.7). In general, there are several types of degradation associated with fingerprint images:

1. The ridges are not strictly continuous; that is, the ridges have small breaks (gaps).
2. Parallel ridges are not well separated. This is due to the presence of noise which links parallel ridges, resulting in their poor separation.
3. Cuts, creases, and bruises on the finger.

Figure 2.6. a) A good quality fingerprint; b) a medium quality fingerprint characterized by scratches and ridge breaks; c) a poor quality fingerprint containing a lot of noise.

These three types of degradation make ridge extraction extremely difficult in the highly corrupted regions. This leads to the following problems in minutiae extraction: (i) a significant number of spurious minutiae are extracted, (ii) a large number of genuine minutiae are missed, and (iii) large errors in the location (position and orientation) of minutiae are introduced. In order to ensure good performance of the ridge and minutiae extraction algorithms in poor quality fingerprint images, an enhancement algorithm to improve the clarity of the ridge structure is necessary.

A fingerprint expert is often able to correctly identify the minutiae by using various visual clues such as local ridge orientation, ridge continuity, ridge tendency, and so on. In theory, it is possible to develop an enhancement algorithm that exploits these visual clues to improve image quality. Generally, for a given fingerprint image, the fingerprint areas resulting from the segmentation step may be divided into three categories (Figure 2.7):

- *Well-defined region*: ridges can be clearly differentiated from each another.
- *Recoverable region*: ridges are corrupted by a small amount of gaps, creases, smudges, links, and the like, but they are still visible and the neighboring regions provide sufficient information about their true structure.
- *Unrecoverable region*: ridges are corrupted by such a severe amount of noise and distortion that no ridges are visible and the neighboring regions do not allow them to be reconstructed.

Good quality regions, recoverable, and unrecoverable regions may be identified according to several criteria; in general, image contrast, orientation consistency, ridge frequency, and other local features may be combined to define a quality index. Since the estimation of fingerprint quality is central for a number of algorithms and practical

applications, a section devoted to quality computation is provided at the end of this chapter. The goal of an enhancement algorithm is to improve the clarity of the ridge structures in the recoverable regions and mark the unrecoverable regions as too noisy for further processing. Usually, the input of the enhancement algorithm is a gray-scale image. The output may either be a gray-scale or a binary image, depending on the algorithm and goal.

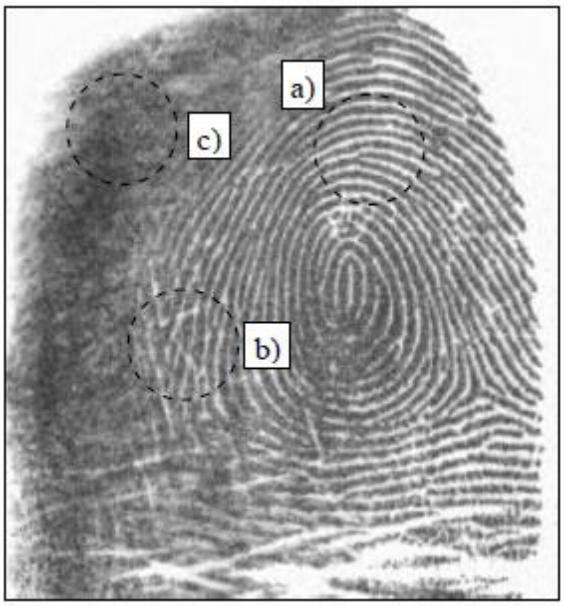


Figure 2.7. A fingerprint image containing regions of different quality: a) a well-defined region; b) a recoverable region; c) an unrecoverable region.

2.3.1 Pixel-wise enhancement

In a pixel-wise image processing operation the new value of each pixel only depends on its previous value and some global parameters (but not on the value of the neighboring pixels). Pixel-wise techniques do not produce satisfying and definitive results for fingerprint image enhancement. However, contrast stretching, histogram manipulation, normalization, and Wiener filtering have been shown to be effective as initial processing steps in a more sophisticated fingerprint enhancement algorithm.

The normalization approach determines the new intensity value of each pixel in an image as

$$\mathbf{I}'[x,y] = \begin{cases} m_0 + \sqrt{(\mathbf{I}[x,y]-m)^2 \cdot v_0 / v} & \text{if } \mathbf{I}[x,y] > m \\ m_0 - \sqrt{(\mathbf{I}[x,y]-m)^2 \cdot v_0 / v} & \text{otherwise,} \end{cases}$$

where m and v are the image mean and variance and $m0$ and $v0$ are the desired mean and variance after the normalization. Figure 2.8 shows an example. Since the mean and variance can change in different regions of a fingerprint image, the above global technique can be implemented in a local fashion: Kim and Park (2002) introduced a block-wise implementation of the above Equation where m and v are the block mean and variance, respectively, and $m0$ and $v0$ are adjusted for each block according to the block features.

Figure 2.8. An example of normalization using (m_0=100, v_o=100)

2.3.2 Contextual filtering

The most widely used technique for fingerprint image enhancement is based on *contextual filters*. In conventional image filtering, only a single filter is used for convolution throughout the image. In contextual filtering, the filter characteristics change according to the local context. Usually, a set of filters is pre-computed and one

of them is selected for each image region. In fingerprint enhancement, the context is often defined by the local ridge orientation and local ridge frequency. In fact, the sinusoidal-shaped wave of ridges and valleys is mainly defined by a local orientation and frequency that varies slowly across the fingerprint area. An appropriate filter that is tuned to the local ridge frequency and orientation can efficiently remove the undesired noise and preserve the true ridge and valley structure.

Several types of contextual filters have been proposed in the literature for fingerprint enhancement. Although they have different definitions, the intended behavior is almost the same: (1) provide a low-pass (averaging) effect along the ridge direction with the aim of linking small gaps and filling impurities due to pores or noise; (2) perform a bandpass (differentiating) effect in the direction orthogonal to the ridges to increase the discrimination between ridges and valleys and to separate parallel linked ridges.

2.3.3 FFT Enhancement

Contextual filtering can be done in the Fourier domain; in fact, it is well-known that a convolution in the spatial domain corresponds to a point-by-point complex multiplication in the Fourier domain. The filter is defined in the frequency domain by the function:

$H(\rho, \theta) = H_{radial}(\rho) \cdot H_{angle}(\theta)$,

where H_{radial} depends only on the local ridge spacing $\rho = 1/f$ and H_{angle} depends only on the local ridge orientation θ. Both H_{radial} and H_{angle} are defined as bandpass filters and are characterized by a mean value and a bandwidth. A set of n discrete filters is derived by their analytical definition. Actually, in the experiments, to reduce the number of filters, only a single value is used for the local ridge frequency and, therefore, the

context is determined only by the orientation. The Fourier transform $\mathbf{P}_i$, $i = 1 \ldots n$ of the filters is pre-computed and stored. Filtering an input fingerprint image **I** is performed as follows (see Figure 2.9).

- The FFT (Fast Fourier Transform) **F** of **I** is computed.
- Each filter $\mathbf{P}_i$ is point-by-point multiplied by **F,** thus obtaining n filtered image transforms $\mathbf{PF}_i$, $i = 1 \ldots n$ (in the frequency domain).
- Inverse FFT is computed for each $\mathbf{PF}_i$ resulting in n filtered images $\mathbf{PI}_i$, $i = 1 \ldots n$ (in the spatial domain).

The enhanced image I_{enh} is obtained by setting, for each pixel [x, y], $I_{enh}[x, y] = PI_k[x, y]$, where k is the index of the filter whose orientation is the closest to θ_{xy}.

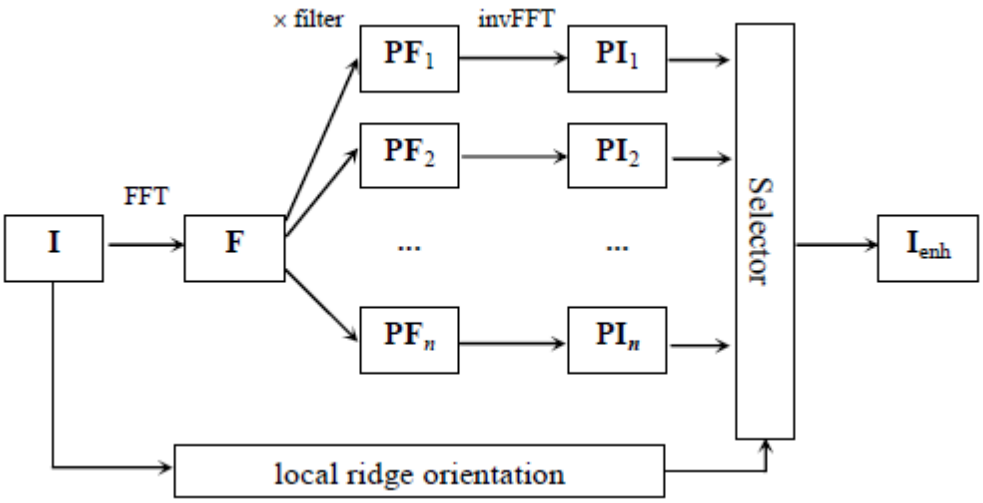


Figure 2.9. Enhancement of the fingerprint image I using FFT

2.3.4 Gabor Filters

One of the most widely cited fingerprint enhancement techniques is the method employed by Hong et al. [49], which is based on the convolution of the image with Gabor filters tuned to the local ridge orientation and ridge frequency. The main stages

of this algorithm include normalisation, ridge orientation estimation, ridge frequency estimation and filtering.

The first step in this approach involves the normalisation of the fingerprint image so that it has a prespecified mean and variance. Due to imperfections in the fingerprint image capture process such as non-uniform ink intensity or non-uniform contact with the fingerprint capture device, a fingerprint image may exhibit distorted levels of variation in grey-level values along the ridges and valleys. Thus, normalisation is used to reduce the effect of these variations, which facilitates the subsequent image enhancement steps.

An orientation image is then calculated, which is a matrix of direction vectors representing the ridge orientation at each location in the image. The widely employed gradient-based approach is used to calculate the gradient [50], which makes use of the fact that the orientation vector is orthogonal to the gradient. Firstly, the image is partitioned into square blocks and the gradient is calculated for every pixel, in the *x* and *y* directions. The orientation vector for each block can then be derived by performing an averaging operation on all the vectors orthogonal to the gradient pixels in the block. Due to the presence of noise and corrupted elements in the image, the ridge orientation may not always be correctly determined. Given that the ridge orientation varies slowly in a local neighbourhood, the orientation image is then smoothed using a low-pass filter to reduce the effect of outliers.

The next step in the image enhancement process is the estimation of the ridge frequency image. The frequency image defines the local frequency of the ridges contained in the fingerprint. Firstly, the image is divided into square blocks and an oriented window is calculated for each block. For each block, an x-signature signal is constructed using the ridges and valleys in the oriented window. The x-signature is the

projection of all the grey level values in the oriented window along a direction orthogonal to the ridge orientation. Consequently, the projection forms a sinusoidal-shape wave in which the centre of a ridge maps itself as a local minimum in the projected wave. The distance between consecutive peaks in the x-signature can then be used to estimate the frequency of the ridges.

Fingerprint enhancement methods based on the Gabor filter have been widely used to facilitate various fingerprint applications such as fingerprint matching and fingerprint classification. Gabor filters are bandpass filters that have both frequency-selective and orientation-selective properties, which means the filters can be effectively tuned to specific frequency and orientation values. One useful characteristic of fingerprints is that they are known to have well defined local ridge orientation and ridge frequency. Therefore, the enhancement algorithm takes advantage of this regularity of spatial structure by applying Gabor filters that are tuned to match the local ridge orientation and frequency.

Based on the local orientation and ridge frequency around each pixel, the Gabor filter is applied to each pixel location in the image. The effect is that the filter enhances the ridges oriented in the direction of the local orientation, and decreases anything oriented differently. Hence, the filter increases the contrast between the foreground ridges and the background, whilst effectively reducing noise. The even symmetric two-dimensional Gabor filter has the following form:

$$g(x, y : \theta, f) = exp\left\{-\frac{1}{2}\left[\frac{{x_\theta}^2}{{\sigma_x}^2} + \frac{{y_\theta}^2}{{\sigma_y}^2}\right]\right\} \cdot cos(2\pi f \cdot x_\theta)$$

where θ is the orientation of the filter, and $[x_\theta, y_\theta]$ are the coordinates of $[x, y]$ after a clockwise rotation of the Cartesian axes by an angle of (90°–θ).

$$\begin{bmatrix} x_\theta \\ y_\theta \end{bmatrix} = \begin{bmatrix} \cos(90° - \theta) & \sin(90° - \theta) \\ -\sin(90° - \theta) & \cos(90° - \theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sin\theta & \cos\theta \\ -\cos\theta & \sin\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

In the above expressions, f is the frequency of a sinusoidal plane wave, and σ_x and σ_y are the standard deviations of the Gaussian envelope along the x- and y-axes, respectively.

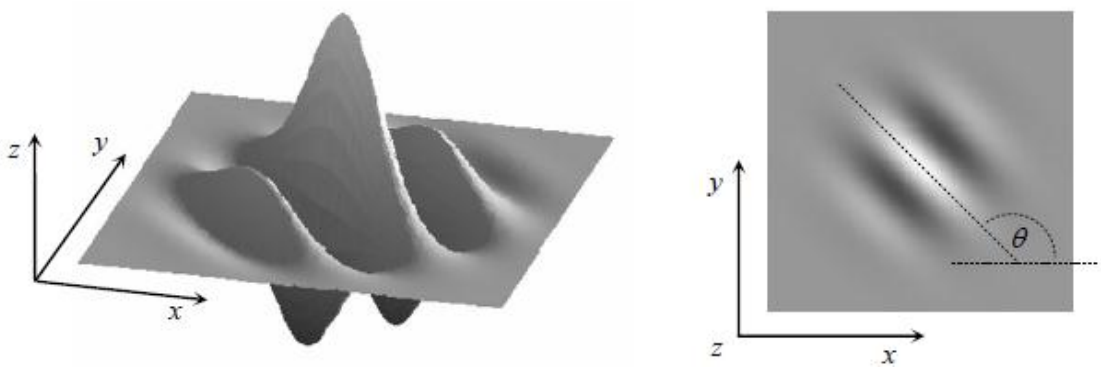


Figure 2.10. Graphical representation (lateral view and top view) of the Gabor filter defined by the parameters $\theta = 135°$, $f = 1/5$, and $\sigma_x = \sigma_y = 3$.

To apply Gabor filters to an image, the four parameters (θ, f, σ_x, σ_y) must be specified. Obviously, the frequency of the filter is completely determined by the local ridge frequency and the orientation is determined by the local ridge orientation. The selection of the values σ_x and σ_y involves a tradeoff. The larger the values, the more robust the filters are to the noise in the fingerprint image, but they are also more likely to create spurious ridges and valleys. On the other hand, the smaller the values, the less likely the filters are to introduce spurious ridges and valleys but then they will be less effective in removing the noise. In fact, from the Modulation Transfer Function (MTF) of the Gabor filter, it can be shown that increasing σ_x and σ_y decreases the bandwidth of the filter and vice versa.

2.4 Core Point Detection

Most of the approaches proposed in the literature for singularity detection operate on the fingerprint orientation image. In this section, the main approaches are discussed.

2.4.1 Poincaré index

An elegant and practical method based on the Poincaré index was proposed by Kawagoe and Tojo (1984). Let **G** be a vector field and *C* be a curve immersed in **G**; then the Poincaré index $P_{\mathbf{G},C}$ is defined as the total rotation of the vectors of **G** along *C* (see Figure 2.11).

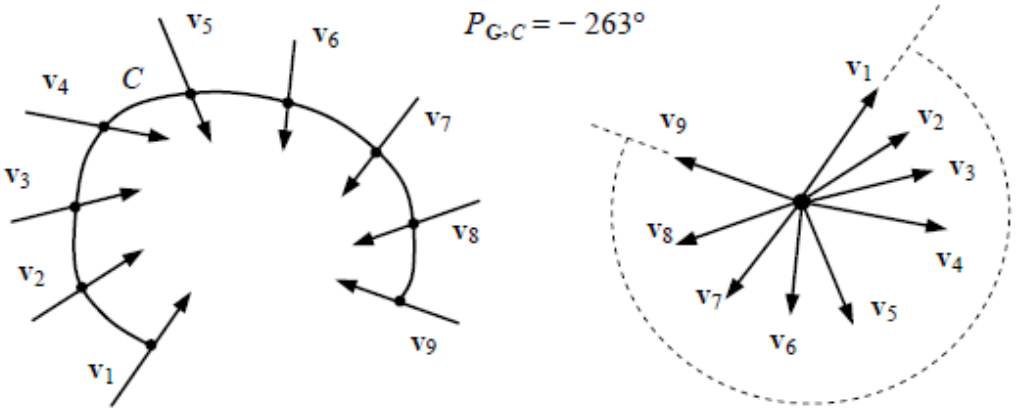


Figure 2.11. The Poincaré index computed over a curve C immersed in a vector field G.

Let **G** be the discrete vector field associated with a fingerprint orientation image **D** and let [*i*,*j*] be the position of the element θ_{ij} in the orientation image; then the Poincaré index $P_{\mathbf{G},C}(i,j)$ at [*i*,*j*] is computed as follows.

- The curve *C* is a closed path defined as an ordered sequence of some elements of **D**, such that [*i*,*j*] is an internal point.

- $P_{G,C}(i,j)$ is computed by algebraically summing the orientation differences between the adjacent elements of C. Summing orientation differences requires a direction (among the two possible) to be associated at each orientation. A solution to this problem is to randomly select the direction of the first element and assign the direction closest to that of the previous element to each successive element. It is well known and can be easily shown that, on closed curves, the Poincaré index assumes only one of the discrete values: 0°, ± 180°, and ± 360°. In the case of fingerprint singularities:

$$P_{G,C}(i,j)=\begin{cases} 0° & \text{if } [i,j] \text{ does not belong to any singular region} \\ 360° & \text{if } [i,j] \text{ belongs to a whorl type singular region} \\ 180° & \text{if } [i,j] \text{ belongs to a loop type singular region} \\ -180° & \text{if } [i,j] \text{ belongs to a delta type singular region.} \end{cases}$$

Figure 2.12 shows three portions of orientation image. The path defining C is the ordered sequence of the eight elements $\mathbf{d_k}$ (k = 0...7) surrounding [i, j]. The direction of the elements $\mathbf{d_k}$ is chosen as follows: $\mathbf{d_0}$ is directed upward; $\mathbf{d}$ (k = 1...7) is directed so that the absolute value of the angle between $\mathbf{d}_k$ and $\mathbf{d}_{k-1}$ is less than or equal to 90°. The Poincaré index is then computed as

$$P_{G,C}(i,j)=\sum_{k=0\ldots7} angle\left(\mathbf{d}_k,\mathbf{d}_{(k+1)\bmod 8}\right).$$

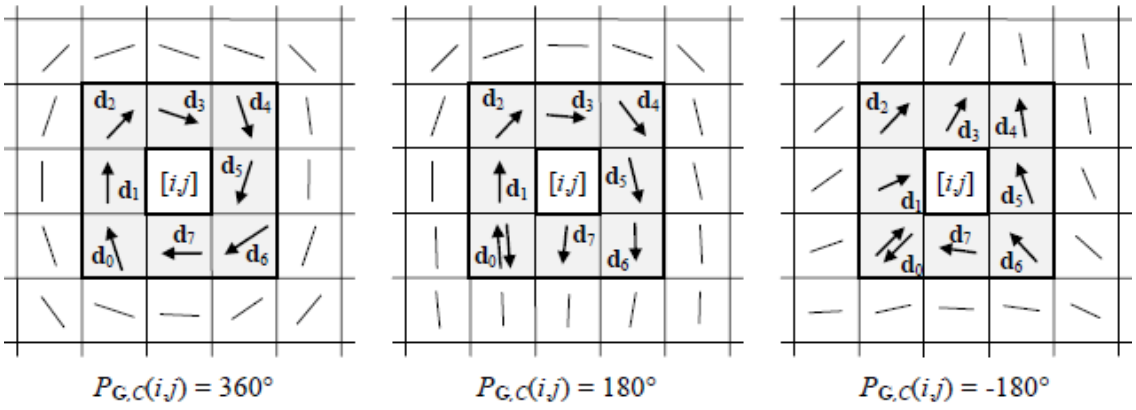


Figure 2.12. Examples of Poincaré index computation in the 8-neighborhood of points belonging (from left to right) to a whorl, loop, and delta singularity, respectively.

2.4.2 Geometry of Region Technique (GR)

The GR technique can be summarized as follows:

1. Compute the smoothed orientation field $\theta'(i, j)$.
2. Compute $\varepsilon(i, j)$, which is the sine component of $\theta'(i, j)$

   $\varepsilon(i, j) = sin(\theta'(i, j))$
3. Initialize a label image A which is used to indicate the core point.
4. Assign the corresponding pixel in *A* the value of the difference in integrated pixel intensity of each region

   $$A(i,j) = \sum_{R1} \varepsilon(i,j) - \sum_{R2} \varepsilon(i,j),$$

   The region *R1* and *R2* were determined empirically and also their geometry are designed to capture the maximum curvature in concave ridges. In practice, the region is defined within the radius of 10-15 pixels (should cover at least 1 ridge). In addition R1 that is sandwiched R2 is expected to hold the maximum point.
5. Find the maximum value in A and assign its coordinate as the core point.

6. If the core point still cannot be located, the steps (1-5) could be iterated for a number of times while decreasing the window size used in step 1 above. For instance; w = 15,10 *and* 5 pixels respectively.

2.4.3 Detection of Curvature Technique (DC)

1. Compute the local orientation $\theta(i, j)$. The input block size could be small as $w = 3$, ie, $k \times l = 3 \times 3$ pixels.
2. Smooth the orientation field $\theta'(i, j)$.
3. In every progressive block, the difference of direction components is computed.

$$Diff\ Y = \sum_{k=1}^{3} sin\,2\theta(k,3) - \sum_{k=1}^{3} sin\,2\theta(k,1)$$

$$Diff\ X = \sum_{l=1}^{3} cos\,2\theta(3,l) - \sum_{i=1}^{3} cos\,2\theta(1,l)$$

4. The curvature point (X) could be located at the corresponding (i, j) where *Diff X* and *Diff Y* are negative.
5. If can't find core point of interesting location. We decrease size image and core point detect again, until find core point or nearby.

There are various other methods that have been proposed [28, 29, 31, 32, 35, 36, 37]. We have developed our own method for locating the core point which will be discussed in chapter 4.

2.5 Minutiae Extraction

The fingerprint image is first binarized and then a thinning algorithm is applied on the binarized image to obtain one pixel wide ridges.

The Crossing Number (CN) method is used to perform minutiae extraction. This method extracts the ridge endings and bifurcations from the skeleton image by examining the local neighbourhood of each ridge pixel using a 3£3 window. The CN for a ridge pixel *P* is given by:

$$CN = 0.5\sum_{i=1}^{8}|P_i - P_{i+1}|, \qquad P_9 = P_1$$

where *Pi* is the pixel value in the neighbourhood of *P*. For a pixel *P*, its eight neighbouring pixels are scanned in an anti-clockwise direction as follows:

P_4	P_3	P_2
P_5	P	P_1
P_6	P_7	P_8

After the CN for a ridge pixel has been computed, the pixel can then be classified according to the property of its CN value. As shown in Figure 2.13, a ridge pixel with a CN of one corresponds to a ridge ending, and a CN of three corresponds to a bifurcation. For each extracted minutiae point, the following information is recorded:

- *x* and *y* coordinates,
- orientation of the associated ridge segment, and
- type of minutiae (ridge ending or bifurcation).

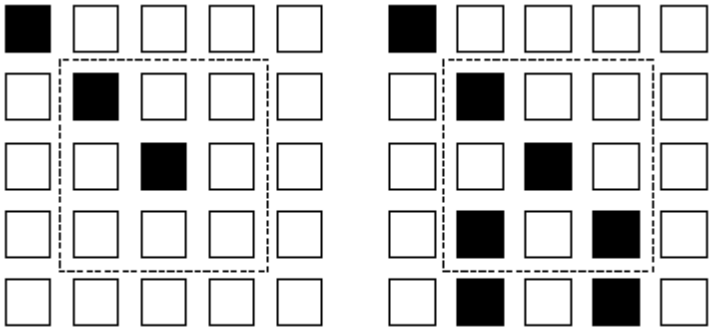


Figure 2.13: Examples of a ridge ending and bifurcation pixel. (a) A Crossing Number of one corresponds to a ridge ending pixel. (b) A Crossing Number of three corresponds to a bifurcation pixel.

3. Biometric Cryptosystems

In this chapter we will discuss about the various biometric cryptosystems and the need for fuzzy vault. We will also explore about the fingerprint based fuzzy vault.

3.1 Cryptography

In traditional cryptographic systems, one or more keys are used to convert the plain text (i.e. data to be encrypted: audio files) to cipher text (i.e. encrypted data: encrypted audio files). The encrypting key(s) maps the plain text to essentially a sequence of pseudo random bits (modern crypto algorithms are designed with this criteria), that can only be mapped back to the plain text using the appropriate decrypting key(s). Without the knowledge of the correct decrypting keys, the conversion of cipher text to the plain text is infeasible considering time and cost limitations. Hence, the cipher text is secured: even if an attacker obtains the cipher text, she cannot extract useful information (i.e. plain text) from it. Here, the plain text can be any data that needs to be stored or transmitted securely: financial transactions, e-mail communication, health records, fingerprint images, secret cryptographic keys, etc.

Fig. 3.1 shows block diagrams of symmetric and asymmetric key cryptographic systems, in the realm of two entities that want to communicate securely (denoted as Alice and Bob). In the symmetric system, the decrypting key is the same as the encrypting key (namely, Alice and Bob share the key K_{AB}). Whereas in the asymmetric system, the decrypting key is not the same as the encrypting key, and it is only known to the recipient of the message: Alice can access Bob's public key (encrypting key) K^{+}_{B}, but only Bob knows his private key (decrypting key) K^{--}_{B}.

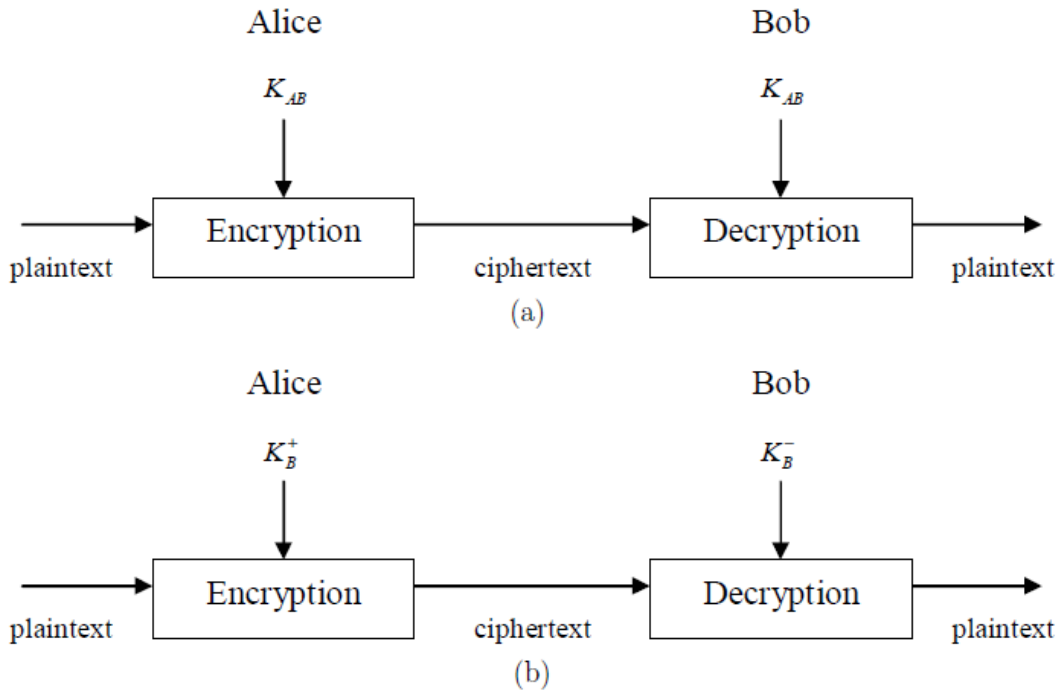


Figure 3.1: Traditional cryptography: (a) symmetric key system, (b) asymmetric key system.

Current cryptographic algorithms (e.g., symmetric key systems Advanced Encryption Standard (AES), Data Encryption Standard (DES), or asymmetric key system RSA) have high theoretical and proven security. That is, there are no publicly known *feasible* procedures to invert the associated cipher text back to the plain text, given the computational resources (processor speed, processor quantity, and storage capacity) available to attackers today. As a result, encryption of multimedia data (by the copyright owner or the data distributor) can be utilized to eliminate the problems of unauthorized copying and distribution: the data will be useless without the knowledge of the correct encrypting and decrypting keys. But this solution also has problems, as explained below.

3.1.1 Problem of Key Sharing

Illegal key sharing (key management problem) is one of their major drawbacks in the current cryptographic systems (e.g., DES, AES, RSA). Regardless of the security of the algorithms, if the keys that need to be known *only* to the legitimate parties in the communication are shared freely, it is trivial to convert cipher text back to plain text. Another limitation of the cryptographic systems is that they require the keys to be very long and random for high security. For example, AES requires at least 128-bits (which corresponds to a 19 character key from a 7-bit ASCII code). This makes it impossible for users to remember the keys. As a result, the cryptographic keys are stored within a physical medium (e.g., in a computer or on a smart card) and released based on some alternative authentication mechanism. If this mechanism succeeds, the released key can be used in encryption/decryption procedures.

The most popular authentication mechanism used for this purpose is based on passwords, which are again cryptographic key-like strings but they are simple enough for users to remember (hence it is not necessary for users to store this information within a physical medium). As an example, the string “CaDburY1990+" can be selected as a password by someone who was born in 1990 and has a cat named *Cadbury*; she can remember it easily *and* she hopes that it cannot be guessed by attackers. Hence, the plain text (e.g., e-mail records, financial records) protected by a cryptographic algorithm is only as secure as the password (the weakest link) that releases the correct decrypting keys. Simple passwords (e.g., “cadbury") compromise security: they can be guessed, either by using social engineering methods (observing the names of pets, relatives, favorite movies . . .), or by brute force search. In fact, even though the theoretical password space can be quite large (for 8 character passwords from 7-bit ASCII code, there are $128^8 \sim 7.2 * 10^{16}$ different passwords), in an

experiment involving 13,797 Unix passwords, Klein [53] was able to crack approximately 25% of the passwords using a dictionary including just 62,727 words. As a natural remedy to this problem, complex passwords are, however, difficult to remember and expensive to maintain (e.g., due to calls to helpdesks to reset a password if the user forgets it). Furthermore, passwords are unable to provide non-repudiation: a subject may deny releasing the key using password authentication, claiming that natural remedy to this problem, complex passwords are, however, difficult to remember and pensive to maintain (e.g., due to calls to helpdesks to reset a password if the user forgets it). Furthermore, passwords are unable to provide non-repudiation: a subject may deny releasing the key using password authentication, claiming that her password was stolen and that a thief released the key.

Many of the above limitations of password-based authentication can be eliminated by incorporating biometric authentication into the cryptographic system. The next section discusses about the biometric cryptosystems.

3.2 Biometric Cryptosystems: An Introduction

Biometric cryptosystems (BCSs) are designed to securely bind a digital key to a biometric or generate a digital key from biometrics, offering solutions to biometric-dependent key-release and biometric template protection. Replacing password-based key-release, BCSs brings about substantial security benefits. It is significantly more difficult to forge, copy, share, and distribute biometrics compared to passwords. Most biometric characteristics provide an equal level of security across a user-group (physiological biometric characteristics are not user selected). Due to biometric variance, conventional biometric systems perform “fuzzy comparisons” by applying decision thresholds which are set up based on score distributions between genuine and

non-genuine subjects. In contrast, BCSs are designed to output stable keys which are required to match a 100% at authentication. Original biometric templates are replaced through biometric-dependent public information which assists the key-release process.

The majority of BCSs require the storage of biometric-dependent public information, applied to retrieve or generate keys, which is referred to as helper data. Due to biometric variance it is not feasible for most biometric characteristics to extract keys directly. Helper data, which must not reveal significant information about original biometric templates, assists in reconstructing keys. Biometric comparisons are performed indirectly by verifying key validities, where the output of an authentication process is either a key or a failure message. Since the verification of keys represents a biometric comparison in encrypted domain, BCSs are applied as a means of biometric template protection [54], in addition to providing biometric-dependent key-release. Based on how helper data are derived, BCSs are classified as key-release or key-generation systems (see Figure 3.2).

3.3 Fuzzy Vault

One of the most popular BCSs called fuzzy vault (see figure 3.3) was introduced by Juels and Sudan [5] in 2002. The key idea of the fuzzy vault scheme is to use an unordered set A to lock a secret key k, yielding a vault, denoted by V_A. If another set B overlaps largely with A, k is reconstructed, i.e., the vault V_A is unlocked. The vault is created applying polynomial encoding and error correction. During the enrollment phase a polynomial p is selected which encodes the key k in some way (e.g., the coefficients of p are formed by k), denoted by p ¬ k. Subsequently, the elements of A are projected onto the polynomial p, i.e., p (A) is calculated. Additionally, chaff

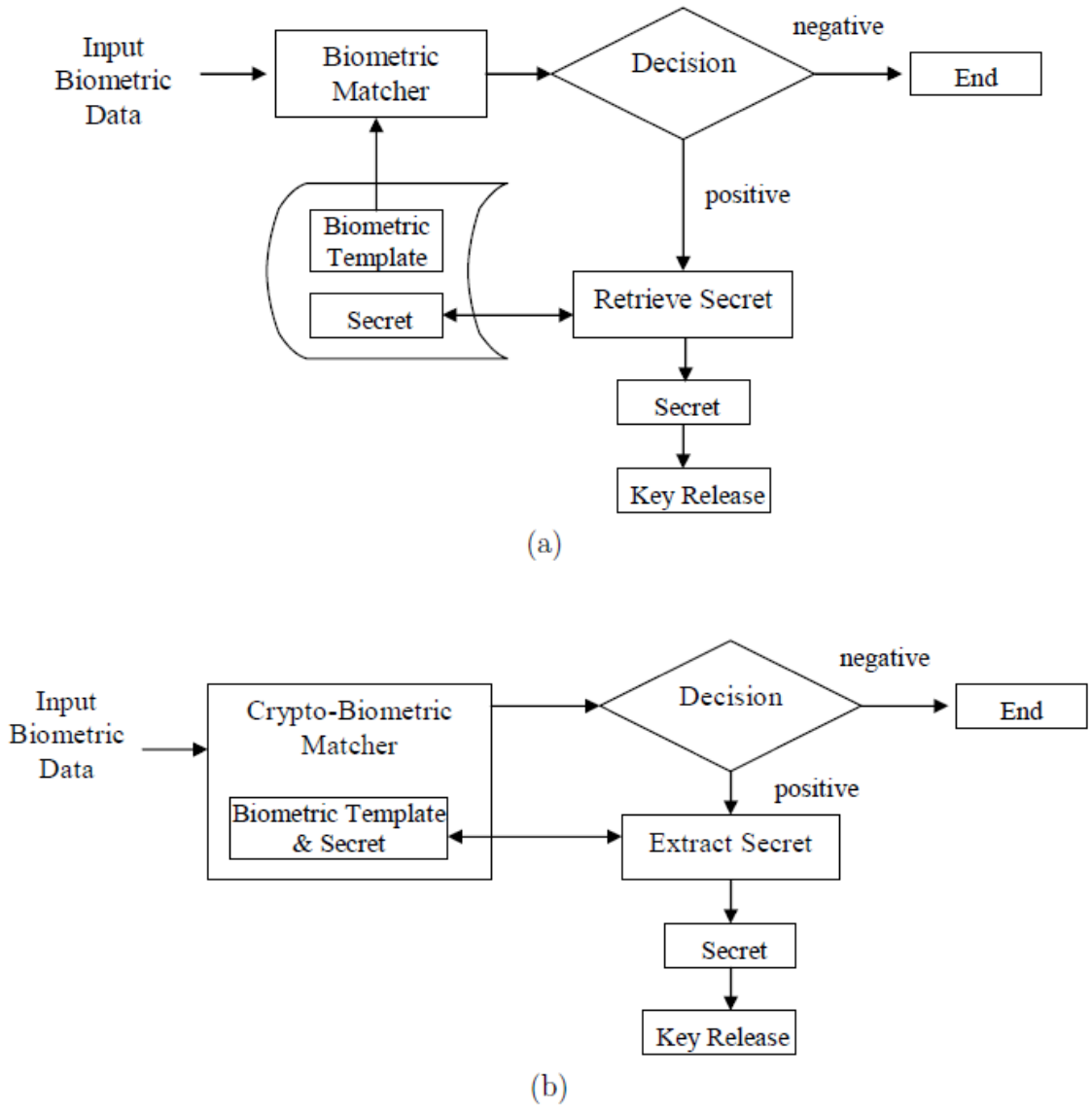


Figure 3.2: Two modes of combining biometrics with cryptography: (a) key release, (b) key generation.

points are added in order to obscure genuine points of the polynomial. The set of all points, R, forms the template. To achieve successful authentication another set B needs to overlap with A to a certain extent in order to locate a sufficient amount of points in R that lie on p. Applying error correction codes, p can be reconstructed and, thus, k. The security of the whole scheme lies within the infeasibility of the polynomial reconstruction and the number of applied chaff points. The main advantage of this

concept is the feature of order invariance, i.e., fuzzy vaults are able to cope with unordered feature set which is the case for several biometric characteristics (e.g., fingerprints).

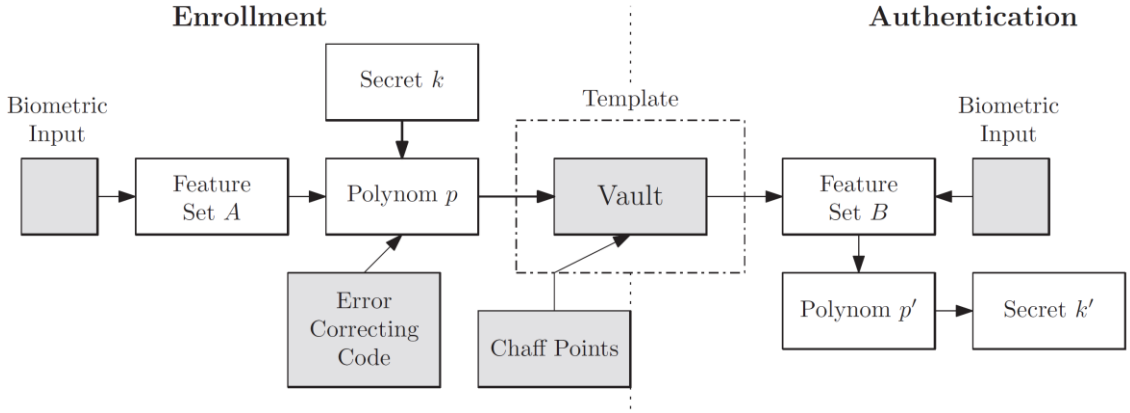


Figure 3.3: Fuzzy vault scheme

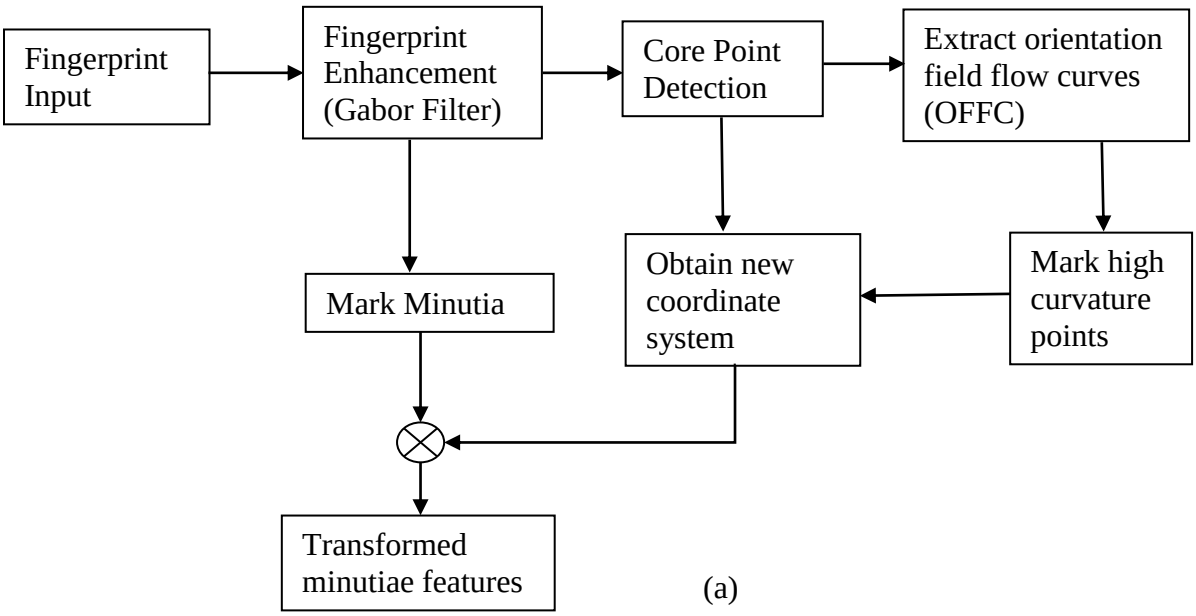
4. Fuzzy Vault Implementation

In this chapter we will discuss our implementation of the fingerprint based fuzzy vault in detail, along with all the steps involved. After enhancing the fingerprint image, the minutiae points are extracted. Also, the core point of the fingerprint is determined and high curvature points are marked to transform the minutiae points to the new coordinate system. After that the biometric features are used to form the fuzzy vault as discussed in 3.3.

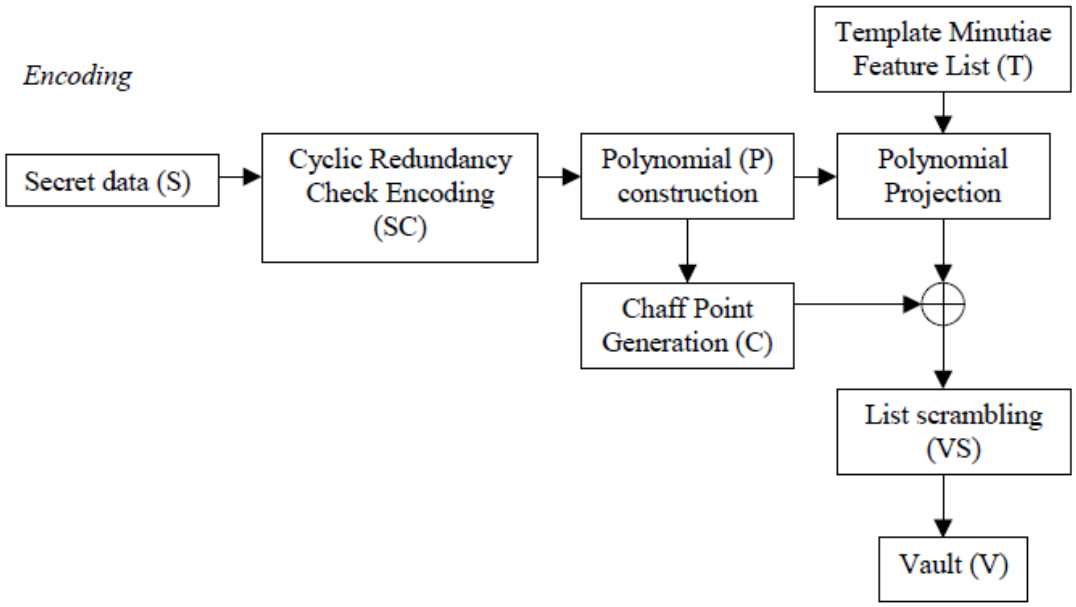
4.1 Architecture of the Proposed System

In this section we present our implementation of the fuzzy vault, operating on fingerprint minutiae features. Note that first we find out the invariant features from the fingerprint and then use that for encoding/decoding the fuzzy vault. Fig. 4.1 shows the block diagram of the proposed fingerprint fuzzy vault system.

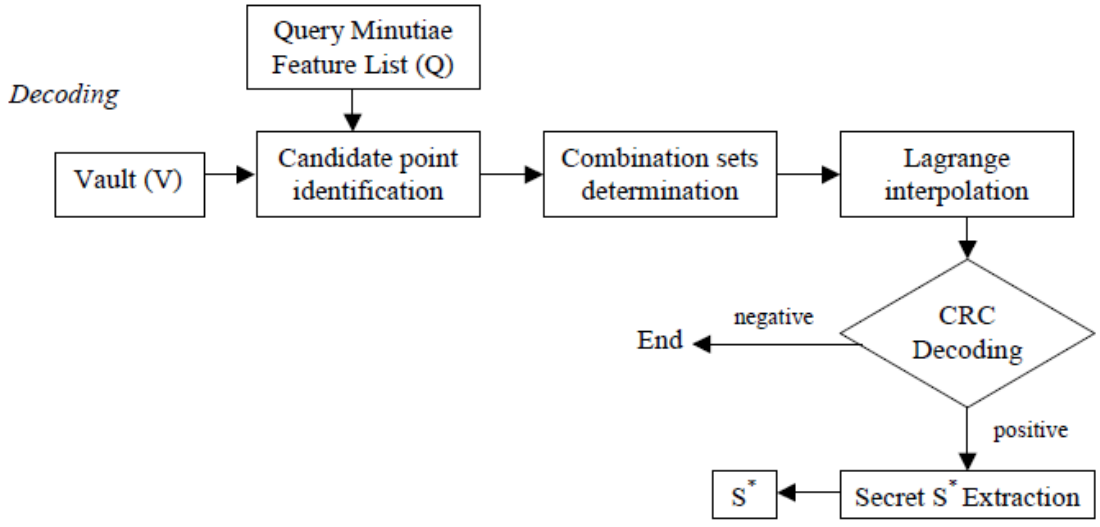
Feature Extraction:



(a)



(b)



(c)

Figure 4.1: Flowchart of the proposed fuzzy fingerprint vault: (a) Feature Extraction, (b) vault encoding, (c) vault decoding.

Fig. 4.2 shows the variables used in the system pictorially: the polynomial in Fig. 4.2(a) encodes the secret. It is evaluated at both genuine (black) and chaff (red) points in Fig. 4.2(b). Finally, the vault is the union of genuine and chaff points with no discriminating information (conveyed via color) attached to them.

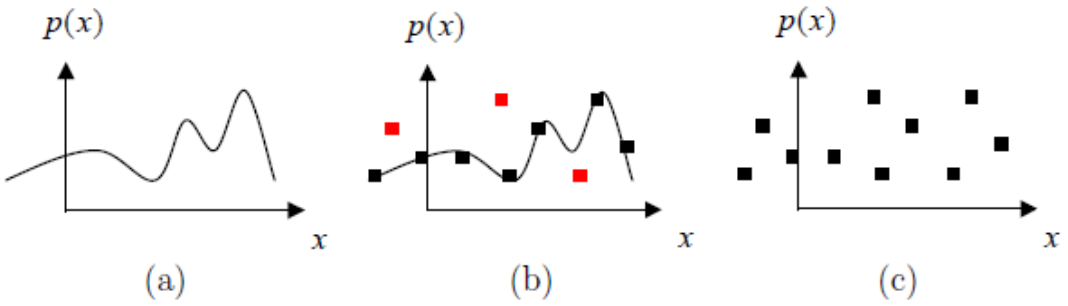


Figure 4.2: Pictorial representation of system variables: (a) polynomial, (b) evaluation of the polynomial (black: genuine points, red: chaff points), (c) final vault list.

4.1 Fingerprint Enhancement

We perform gabor filtering to enhance the fingerprint image. As discussed in section 2.3, various methods exist for enhancing the fingerprint template. But, after some experimentation we concluded that gabor filter provided the best results. Before filtering, we have to do segmentation to find out the region of interest in the fingerprint. The application of gabor filtering involves the following steps:

- normalisation,
- orientation estimation,
- ridge frequency estimation, and
- Gabor filtering.

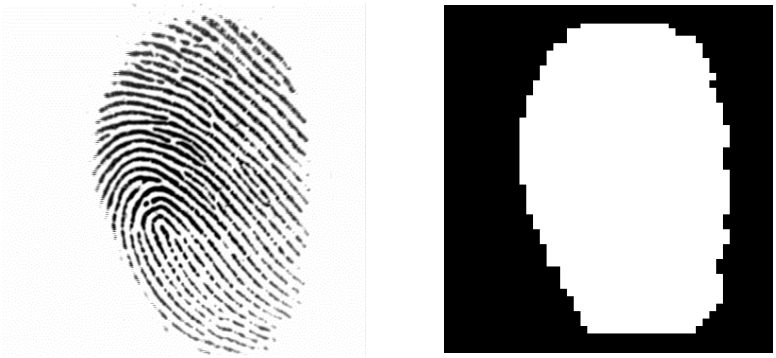
4.2.1 Segmentation

The first step of the fingerprint enhancement algorithm is image segmentation. Segmentation is the process of separating the foreground regions in the image from the background regions. The foreground regions correspond to the clear fingerprint area containing the ridges and valleys, which is the area of interest. The background corresponds to the regions outside the borders of the fingerprint area, which do not contain any valid fingerprint information. When minutiae extraction algorithms are applied to the background regions of an image, it results in the extraction of noisy and false minutiae. Thus, segmentation is employed to discard these background regions, which facilitates the reliable extraction of minutiae.

In a fingerprint image, the background regions generally exhibit a very low grey-scale variance value, whereas the foreground regions have a very high variance. Hence, a method based on variance thresholding can be used to perform the segmentation. Firstly, the image is divided into blocks and the grey-scale variance is calculated for each block in the image. If the variance is less than the global threshold, then the block is assigned to be a background region; otherwise, it is assigned to be part of the foreground. The grey-level variance for a block of size W x W is defined as:

$$V(k) = \frac{1}{W^2} \sum_{i=0}^{W-1} \sum_{j=0}^{W-1} (I(i,j) - M(k))^2$$

where $V(k)$ is the variance for block k, $I(I, j)$ is the grey-level value at pixel (I, j), and $M(k)$ is the mean grey-level value for the block k. In our implementation we have used W=8 and set the threshold value to 0.01. Figure 4.3 shows the output of segmentation. The region in white in fig 4.3 (b) is the region of interest.



(a) (b)

Figure 4.3: Segmentation, (a) Input Image, (b) Segmentation boundary

4.2.2 Normalisation

The next step in the fingerprint enhancement process is image normalisation. Normalisation is used to standardise the intensity values in an image by adjusting the range of grey-level values so that it lies within a desired range of values. Let $I(i, j)$ represent the grey-level value at pixel (i, j), and $N(i, j)$ represent the normalised grey-level value at pixel (i, j). The normalised image is defined as:

$$N(i,j) = \begin{cases} M_0 + \sqrt{\frac{V_0(I(i,j)-M)^2}{V}} & \text{if } I(i,j) > M, \\ M_0 - \sqrt{\frac{V_0(I(i,j)-M)^2}{V}} & \text{otherwise,} \end{cases}$$

where M and V are the estimated mean and variance of $I(i, j)$, respectively, and M_0 and V_0 are the desired mean and variance values, respectively. Normalisation does not change the ridge structures in a fingerprint; it is performed to standardise the dynamic levels of variation in grey-level values, which facilitates the processing of subsequent image enhancement stages. In our implementation, we have taken M_0 and V_0 as 50.

4.2.3 Orientation estimation

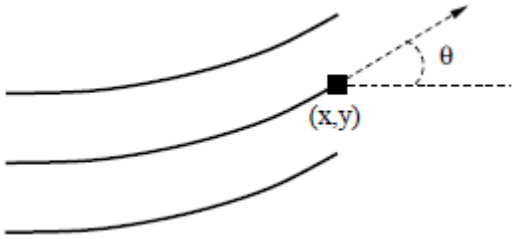


Figure 4.4: The orientation of a ridge pixel in a fingerprint.

The orientation field of a fingerprint image defines the local orientation of the ridges contained in the fingerprint (see Figure 4.4). The orientation estimation is a fundamental step in the enhancement process as the subsequent Gabor filtering stage relies on the local orientation in order to effectively enhance the fingerprint image. The least mean square estimation method employed by Hong et al. is used to compute the orientation image. However, instead of estimating the orientation block-wise, I have chosen to extend their method into a pixel-wise scheme, which produces a finer and more accurate estimation of the orientation field. The steps for calculating the orientation at pixel (i, j) are as follows:

1. Firstly, a block of size W x W is centred at pixel $(i; j)$ in the normalised fingerprint image.
2. For each pixel in the block, compute the gradients $\partial_x(i, j)$ and $\partial_y(i, j)$, which are the gradient magnitudes in the x and y directions, respectively. The horizontal Sobel operator is used to compute $\partial_x(i; j)$:

$$\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

The vertical Sobel operator is used to compute $\partial_y(i, j)$:

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

3. The local orientation at pixel ($i; j$) can then be estimated using the following equations:

$$V_x(i,j) = \sum_{u=i-w/2}^{i+w/2} \sum_{v=j-w/2}^{j+w/2} \partial_x^2(u,v) - \partial_y^2(u,v)$$

$$V_y(i,j) = \sum_{u=i-w/2}^{i+w/2} \sum_{v=j-w/2}^{j+w/2} 2\partial_x^2(u,v)\partial_y^2(u,v)$$

$$\theta(i,j) = \frac{\pi}{2} + \frac{1}{2}\angle\left(V_x(i,j), V_y(i,j)\right)$$

where,

$$\angle(a,b) = \begin{cases} \tan^{-1}(b/a) & if\ a \geq 0 \\ \tan^{-1}(b/a) + \pi & if\ a < 0, b \geq 0 \\ \tan^{-1}(b/a) - \pi & if\ a < 0, b < 0 \end{cases}$$

4.2.4 Ridge Frequency Estimation

In addition to the orientation image, another important parameter that is used in the construction of the Gabor filter is the local ridge frequency. The frequency image represents the local frequency of the ridges in a fingerprint. The first step in the frequency estimation stage is to divide the image into blocks of size W x W. The next step is to project the grey-level values of all the pixels located inside each block along a direction orthogonal to the local ridge orientation. This projection forms an almost

sinusoidal-shape wave with the local minimum points corresponding to the ridges in the fingerprint. An example of a projected waveform is shown in Figure 4.5.

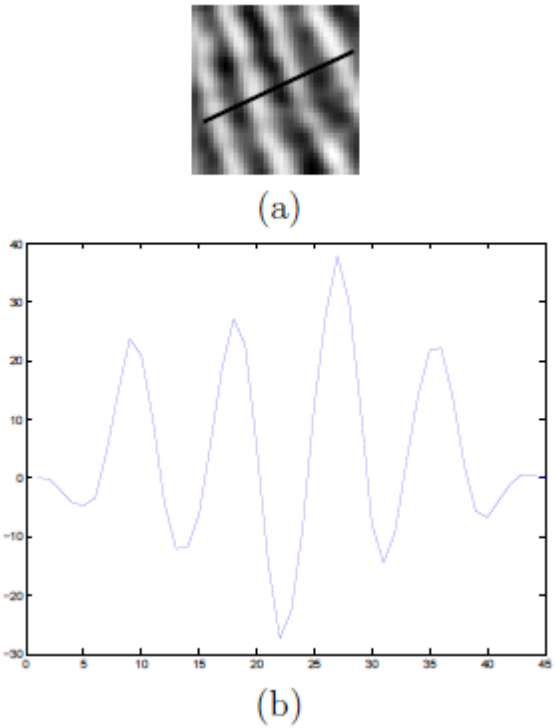


Figure 4.5: The projection of the intensity values of the pixels along a direction orthogonal to the local ridge orientation. (a) A 32 x 32 block from a fingerprint image. (b) The projected waveform of the block.

The ridge spacing $S(i, j)$ is then computed by counting the median number of pixels between consecutive minima points in the projected waveform. Hence, the ridge frequency $F(i, j)$ for a block centred at pixel (i, j) is defined as:

$$F(i,j) = \frac{1}{S(i,j)}.$$

Given that the fingerprint is scanned at a fixed resolution, then ideally the ridge frequency values should lie within a certain range. However, there are cases where a valid frequency value cannot be reliably obtained from the projection. Examples are

when no consecutive peaks can be detected from the projection, and also when minutiae points appear in the block. For the blocks where minutiae points appear, the projected waveform does not produce a well-defined sinusoidal-shape wave, which can lead to an inaccurate estimation of the ridge frequency. Thus, the out of range frequency values are interpolated using values from neighbouring blocks that have a well-defined frequency.

4.2.5 Gabor filtering

Once the ridge orientation and ridge frequency information has been determined, these parameters are used to construct the even-symmetric Gabor filter. A two-dimensional Gabor filter consists of a sinusoidal plane wave of a particular orientation and frequency, modulated by a Gaussian envelope. Gabor filters are employed because they have frequency-selective and orientation-selective properties. These properties allow the filter to be tuned to give maximal response to ridges at a specific orientation and frequency in the fingerprint image. Therefore, a properly tuned Gabor filter can be used to effectively preserve the ridge structures while reducing noise.

The even-symmetric Gabor filter is the real part of the Gabor function, which is given by a cosine wave modulated by a Gaussian (see Figure 4.6). An even-symmetric Gabor filter in the spatial domain is defined as [55]:

$$G(x, y; \theta, f) = \exp\left\{-\frac{1}{2}\left[\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}\right]\right\}\cos(2\pi f x_\theta),$$

$$x_\theta = x\cos\theta + y\sin\theta,$$

$$y_\theta = -x\sin\theta + y\cos\theta,$$

where θ is the orientation of the Gabor filter, f is the frequency of the cosine wave, σ_x and σ_y are the standard deviations of the Gaussian envelope along the x and y axes, respectively, and $x\mu$ and $y\mu$ define the x and y axes of the filter coordinate frame, respectively.

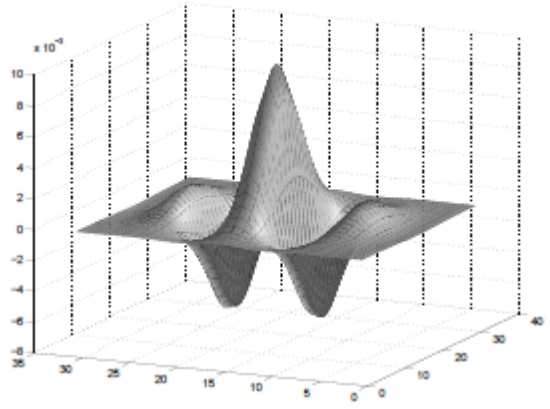


Figure 4.6: An even-symmetric Gabor filter in the spatial domain.

The Gabor filter is applied to the fingerprint image by spatially convolving the image with the filter. The convolution of a pixel (i, j) in the image requires the corresponding orientation value $O(i, j)$ and ridge frequency value $F(i, j)$ of that pixel. Hence, the application of the Gabor filter G to obtain the enhanced image E is performed as follows:

$$E(i,j) = \sum_{u=-\frac{w_x}{2}}^{\frac{w_x}{2}} \sum_{v=-\frac{w_y}{2}}^{\frac{w_y}{2}} G(u,v;O(i,j),F(i,j))N(i-u,j-v),$$

where O is the orientation image, F is the ridge frequency image, N is the normalised fingerprint image, and w_x and w_y are the width and height of the Gabor filter mask, respectively.

The filter bandwidth, which specifies the range of frequency the filter responds to, is determined by the standard deviation parameters σ_x and σ_y. Since the bandwidth

of the filter is tuned to match the local ridge frequency, then it can be deduced that the parameter selection of σ_x and σ_y should be related with the ridge frequency. However, we have empirically set σ_x and σ_y to fixed values of 4.0 and 4.0, respectively. Figure 4.7 shows the fingerprint image after applying the gabor filter.

(a) (b)

Figure 4.7: Gabor filtering, (a) Original Fingerprint image, (b) After applying gabor filter

4.3 Minutia Extraction

4.3.1 Binarization

Our minutiae extraction algorithm operates on binary images where there are only two levels of interest: the black pixels that represent ridges, and the white pixels that represent valleys. Binarization is the process that converts a greylevel image into a binary image. This improves the contrast between the ridges and valleys in a fingerprint image, and consequently facilitates the extraction of minutiae.

One useful property of the Gabor filter is that it has a DC component of zero, which means the resulting filtered image has a mean pixel value of zero. Hence, straightforward binarization of the image can be performed using a global threshold of zero. The binarization process involves examining the grey-level value of each pixel in the enhanced image, and, if the value is greater than the global threshold, then the pixel value is set to a binary value one; otherwise, it is set to zero. The outcome is a binary image containing two levels of information, the foreground ridges and the background valleys.

4.3.2 Thinning

The final step typically performed prior to minutiae extraction is thinning. Thinning is a morphological operation that successively erodes away the foreground pixels until they are one pixel wide. A standard thinning algorithm is employed, which performs the thinning operation using two subiterations. This algorithm is accessible in MATLAB via the `thin' operation under the bwmorph function. Each subiteration begins by examining the neighbourhood of each pixel in the binary image, and based on a particular set of pixel-deletion criteria, it checks whether the pixel can be deleted or not. These subiterations continue until no more pixels can be deleted.

The application of the thinning algorithm to a fingerprint image preserves the connectivity of the ridge structures while forming a skeletonised version of the binary image. This skeleton image is then used in the subsequent extraction of minutiae. Figure 4.8 shows the effect of thinning.

(a) (b)

Figure 4.8: Thinning, (a) Filterted image, (b) Thinned image

4.3.3 Mark Minutiae

The Crossing Number (CN) method is used to perform minutiae extraction. This method extracts the ridge endings and bifurcations from the skeleton image by examining the local neighbourhood of each ridge pixel using a 3x3 window. The CN for a ridge pixel *P* is given by:

$$CN = 0.5\sum_{i=1}^{8}|P_i - P_{i+1}|, \qquad P_9 = P_1$$

where *Pi* is the pixel value in the neighbourhood of *P*. For a pixel *P*, its eight neighbouring pixels are scanned in an anti-clockwise direction as follows:

P_4	P_3	P_2
P_5	P	P_1
P_6	P_7	P_8

After the CN for a ridge pixel has been computed, the pixel can then be classified according to the property of its CN value. As shown in Figure 4.9, a ridge

pixel with a CN of one corresponds to a ridge ending, and a CN of three corresponds to a bifurcation. For each extracted minutiae point, the following information is recorded:

- x and y coordinates,
- type of minutia (ridge ending or bifurcation).

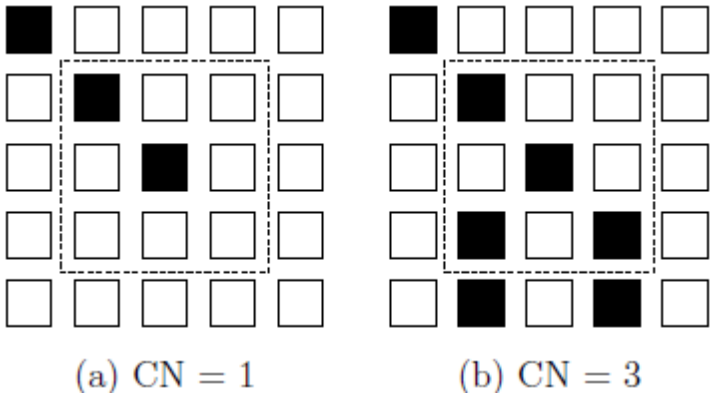


Figure 4.9: Examples of a ridge ending and bifurcation pixel. (a) A Crossing Number of one corresponds to a ridge ending pixel. (b) A Crossing Number of three corresponds to a bifurcation pixel.

After extracting the minutiae points we perform **postprocessing**:

- If the minutia point is less w pixels inside the segmentation boundary, it is removed. (We have used w=12).
- If the distance between a minutia point ‘a’ and ‘b’ is less than ‘d’, then all the minutia points which are at a distance less than ‘d’ from ‘a’ are iteratively removed. While doing this, we have ensured that if one of the points is a bifurcation, then it has been preserved. (we have used d=20).

All the false minutiae have been removed during postprocessing. Figure 4.10 shows the minutiae points in a fingerprint image.

Figure 4.10 Minutia points

4.4 Translation

The minutia points extracted during the previous stage are translated to a new coordinate system with origin at core point. The transformation and rotation factors have to be determined. We find out the high curvature points of the orientation field flow curves of the fingerprint template and the linear regression of all those points is chosen as one of the axes for the new coordinate system. We have explored the using this for cartesian system as well as the polar system. In cartesian system, the other axis would be orthogonal to the reference line we obtained. Whereas in polar system, the angle is calculated with reference to the axis obtained. The entire process has been explained step by step in this section.

4.4.1 Core Point Determination

Core point is one of the singularity points which can be located in most of the fingerprints. In practice, the core point corresponds to the center of the north most loop type singularity. For fingerprints that do not contain loop or whorl singularities, it is difficult to define the core. In these cases, the core is usually associated with the point of maximum ridge line curvature. Unfortunately, due to the high variability of fingerprint patterns, it is difficult to reliably locate a registration (core) point in all the fingerprint images.

We explored all the existing methods and then devised our own method which can detect the core point in most of the fingerprints with high accuracy. First of all, we have to calculate the following:

$$V_x(i,j) = \sum_{u=i-w/2}^{i+w/2} \sum_{v=j-w/2}^{j+w/2} \partial_x^2(u,v) - \partial_y^2(u,v)$$

$$V_y(i,j) = \sum_{u=i-w/2}^{i+w/2} \sum_{v=j-w/2}^{j+w/2} 2\partial_x^2(u,v)\partial_y^2(u,v)$$

$$\emptyset(i,j) = \frac{1}{2}\tan^{-1}\left(V_y(i,j) \Big/ V_x(i,j)\right)$$

where ∂_x and ∂_y are the gradient values in *x* and *y* direction which are calculated using the sobel operator (section 4.2.3). Figure 4.11 shows the plot of the sign of $\emptyset$ in the fingerprint image. The green region denotes positive $\emptyset$ and the red region marks the region where $\emptyset$ is negative.

Figure 4.11: Plot of sign(∅). Positive is denoted by green and negative by red.

From the above figure we can see that the core point is the point where the positive and the negative regions intersect. We have designed an algorithm that determines this point. The steps are:

1. Traverse in the upwards direction vertically as well as 45^0 in both directions to the vertical for 'p' pixels. While doing so, keep a count of the number of points where ∅ is positive. Do the same for negative values.
2. Repeat step 1 in downward direction as well.
3. Traverse in the horizontal direction and 45^0 to it on both the sides for 'p' pixels. Again maintain the count of positive and negative ∅.
4. Now, if any one of the verticals and horizontals have a total count of more than 'n' with each one of them having opposite sign, then we have located the core point.

We have chosen 'p' as 10 and 'n' as 38. We arrived at these values after performing some experiments on varied fingerprints. Figure 4.12 shows the core point marked in the fingerprint template. We have applied this algorithm on the filtered image as it doesn't contain any noise and is continuous.

Figure 4.12: Core point (marked in red)

4.4.2 High Curvature points

Orientation field flow curves of the fingerprint are obtained to mark the high curvature points. This is done to align the query and template fingerprints. The Orientation Field Flow Curves (OFFC) are sets of piecewise linear segments that represent the underlying flow of fingerprint ridges [2]. They are robust to noise arising from minutiae, islands, smudges, and cuts. These curves are obtained as follows.

Consider a site s in a fingerprint image I with r rows and c columns. The orientation field of I gives the direction of the ridge flow in a local neighborhood around s for all s € I. The value of the orientation at site s, o_s, is a vector $(\cos \theta_s, \sin \theta_s)^T$ where θ_s is the angle of the flow with respect to the horizontal axis. Opposite flow directions are equivalent, and therefore, θ_s can only be determined uniquely in $(-\Pi/2, \Pi/2)$. There are many algorithms in the literature that find orientations based on the gray intensities of a given image.

The orientation field estimate is obtained for sites s = (x, y) in I where x and y are integers such that $1 \leq x \leq r$ and $1 \leq y \leq c$. In order to obtain the value of the ridge orientation at any site s = (x, y) in I, we adopt an interpolation scheme. Let m and n be integers such that m = [x] and n = [y], where [g] stands for the greatest integer less than or equal to g. The orientation vector at site s = (x, y) is given by $o_s = (\cos \theta_s, \sin \theta_s)^T$ where

$$\theta_s = \frac{1}{2}\tan^{-1}\frac{\sum_{(i,j)\in\{0,1\}^2} u_i v_j \sin 2\theta_{(m+i,n+j)}}{\sum_{(i,j)\in\{0,1\}^2} u_i v_j \cos 2\theta_{(m+i,n+j)}},$$

with $u_0 = m + 1 - x$, $u_1 = 1 - u_0$, $v_0 = n + 1 - y$, and $v_1 = 1 - v_0$. This interpolation scheme is a weighted average of orientation field values at the integer sites (m, n), (m, n + 1), (m + 1, n) and (m + 1, n + 1). The weights are given by $u_i v_j$ with (i, j) taking values in $\{0, 1\}^2$. This interpolation scheme yields a value of orientation for all sites s € I while retaining the original values at the integer sites.

An OFFC with a starting point s_0 € I can be defined iteratively as

$$s_j = s_{j-1} + d_j * l_j * O_{sj-1}$$

for j = 1, 2,. . . . , n; dj, with values in {-1, +1}, is the flow direction from s_{j-1} to s_j, l_j is the length of the line segment from s_{j-1} to s_j, and O_{sj-1} is the orientation vector at site s_{j-1}. The point s_n denotes the termination point of the OFFC curve, which is achieved when either (i) the boundaries of the image are reached, or (ii) when n exceeds a pre-specified constant N_0. The lengths l_j specifies the sampling interval of the OFFC. Each point s_0 generates two segments of an OFFC which are obtained by fixing d_j first at +1, and then at -1, so that the points s_j trace opposite directions. The starting points s_0 are selected in the following way: Let rstart, rend, cstart and cend determine the top, bottom, left and right boundaries of the fingerprint pattern area, and w denote the sampling width. The points s_0 are selected such that

$$s_0 = [core(1,1), cstart + l*w]$$

with l = 1, 2, ... , (cend - cstart) / (w). In other words, the starting points are sampled along the horizontal line passing the through the core point. Figure 4.13 shows the orientation field flow curves of a fingerprint image.

4.13 Orientation field flow curves

The next step is to find out the high curvature points of each of these curves. The highest curvature point would be the one whose neighbours subtend the minimum angle. The set of points on the piece-wise linear OFF curves are used to find the maximum curvature points locations. Namely, given a curve $OFFC_1$, (i) the curvature angle is calculated for every point on the curve, (ii) the point with the maximum curvature (i.e. minimum angle) is identified, and (iii) its location is added to the set of high curvature points (H).

The angles subtended by multiple (10) neighbors of a point *p*, and a linearly weighted average of those angles is calculated, with more weight given to the farther neighbours.

1. For a point *p*, calculate the angles subtended by its 1-neighbors, 2- neighbors, ..., 10-neighbors (a_1, a_2,. . . . . , a_{10}).
2. Calculate the average curvature angle as

$$C = \frac{[a_1 \ a_2 \ \ldots\ldots a_{10}] * [w_1 \ w_2 \ \ldots\ldots w_{10}]^t}{w_1 + \ w_2 + \ \ldots. + w_{10}}$$

The weight vector used is

$$[w_1 \ w_2 \ w_3 \ \ldots\ldots w_{10}]^t = [1.0 \ \ 1.5 \ \ 2.0 \ \ldots \ldots \ldots .5.5]^t$$

A point on each of the curves with minimum value of C is added to the set H. The set of high curvature points is marked with blue colour in figure 4.14.

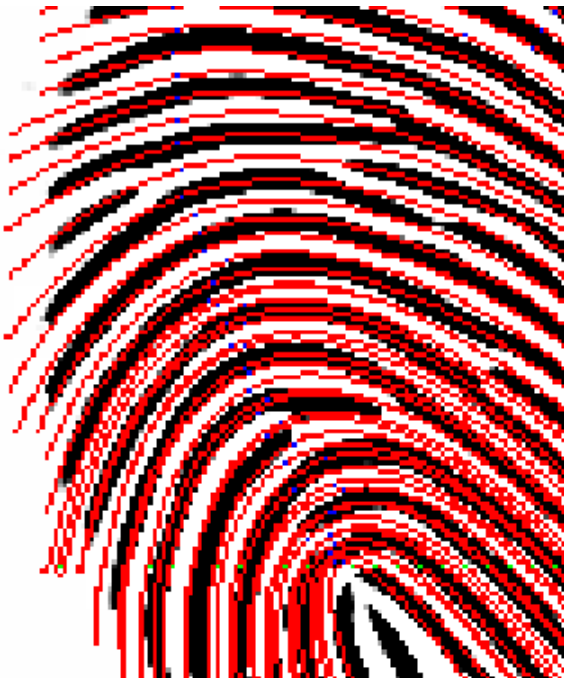


Figure 4.14 High Curvature Points

4.4.3 Determination of Axis

We have used the high curvature points obtained in the previous section to determine an axis for the new coordinate system. We have used linear regression to determine the closest line passing through most of the points and then refined it iteratively by applying learning technique. The steps are:

1. Find the distance of all the high curvature points from the core point and remove the points which are at a distance greater than '*thresh'.*
2. Add core point to the list of high curvature points (H) which was modified in the previous step.
3. Find the centroid of all the points in the set H and remove the points which are at a distance greater than *'dist'* from the centroid.
4. The linear regression is calculated according to the formula:

$$Slope\ (m) = \frac{N * \sum xy - (\sum x) * (\sum y)}{N * \sum x^2 - (\sum x)^2}$$

$$Intercept\ (c) = \frac{\sum y - m * (\sum x)}{N}$$

   And , y=m*x + c is the equation of the line.

5. Calculate the correlation coefficient, r. It gives the degree of association between two variables.

$$r = \frac{n * \sum xy - (\sum x) * (\sum y)}{\sqrt{N * \sum x^2 - (\sum x)^2} * \sqrt{N * \sum y^2 - (\sum y)^2}}$$

6. Find the distance of all the points in H from the line of regression obtained. Remove the point having the maximum distance *'d'* from H.
7. Exit if abs(r) is greater than or equal to 0.93 or the number of points left in H is less than 20. Go to step 4 if 'd' is greater than 7, otherwise exit.

Thus, the line with slope, m, is the axis for the new coordinate system. Figure 4.15 shows the axis obtained in a fingerprint image.

Figure 4.15: Fingerprint image showing the axis

4.4.4 Translation and Rotation

To find the minutia points in the new coordinate system, we need to apply translation and rotation on all the points. The origin is the core point and the slope is the slope (m) obtained in the previous section.

Translated_point = Point – core

Next, we perform the rotation by using,

$$\theta = \tan^{-1} m$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

4.5 Fuzzy Vault Encoding

Secret *S* is any data that needs to be protected, but the size of S that can be feasibly protected is limited by the capacity of the entity used for locking and unlocking the vault. We have used two different features of minutia points for locking/unlocking the vault:

1. *x* and *y* coordinates of minutia points (in the translated domain)
2. Polar coordinates, r and θ (in the translated domain)

The encoding operation secures *S* with fingerprint minutiae data: if a query minutiae set that is similar to the template minutiae set is presented during decoding, it indicates the presence of an authorized person, and *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 template 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. This can simulate securing AES symmetric encryption keys.

Our current decoding implementation does not include any error-correction scheme, as proposed by Juels and Sudan [5], since realizing the necessary polynomial reconstruction via error-correction has not been demonstrated in the literature. Instead, our algorithm decodes many candidate secrets (as explained below). To identify which one of these candidates is the actual secret, we need to put some structure into the

initial secret *S*. By checking the validity of this structure during decoding, the algorithm can identify whether a given candidate secret is correct or not.

Cyclic Redundancy Check (CRC) is a generalization of simple parity bit checking. It is commonly used in communication channel applications for error detection, where the errors are introduced due to channel noise. In our case, using incorrect minutiae points during decoding will cause an incorrect polynomial reconstruction, resulting in errors. In our current implementation, we generate 16-bit CRC data from the initial secret *S*. Hence, the chance of a random error being undetected (i.e., failing to identify an incorrect decoding) is 2^{-16}. The 16-bit primitive polynomial we use for CRC generation is called "CRC-16":

$$G_{CRC(a)} = a^{16} + a^{12} + a^5 + 1$$

Appending the 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 fields with cardinality 65,536, namely $GF(2^{16})$. To obtain the 16-bit locking/unlocking data unit u, we have two different approaches:

1. Cartesian Coordinates:

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 nonlinear distortions), raw minutiae data are first quantized. We apply linear quantization separately on *x* and *y* coordinates, which allows for a variation of ± 10 pixels.

2. Polar Coordinates:

We apply a non linear quantization on r by using the μ-law. This is done because the chance of error increases as the distance from the core increases. And, the overall angle of the circle, 2π, is divided into 16 equal sectors. Based upon the sector in which θ lies, it is assigned one of the 16 discrete values. After that, r (9 bits) and θ(7 bits), are concatenated to obtain the 16-bit locking/unlocking unit u.

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 $D = 8$:

$$p(u) = c_8u^8 + c_7u^7 + \ldots + c_1u + c_0$$

Simply, *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**.

Now, two sets composed of point pairs need to be generated. The first one, called genuine set G, is found by evaluating $p(u)$ on the template minutiae features (T). Assuming that we have N template minutiae, $u_1, u_2, \ldots, u_N$, we construct the set G

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

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. Assuming we need to add M chaff points, we first generate M unique random points, c_1, c_2, ... , c_M in the field $GF(2^{16})$, with the constraint that they do not overlap with u_1, u_2, ... , u_N:

$$c_j \neq u_i \, , j = 1, 2, \ldots . ,M, i = 1, 2, \ldots . ,N$$

Then, we generate another set of M random points, d_1, d_2, ... , d_M, with the constraint that the pairs (c_j , d_j), j = 1, 2, ... ,M do not fall onto the polynomial $p(u)$. The chaff set C is then

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

where

$$d_j \neq p(c_j), j = 1, 2, \ldots . ,M$$

The union of these two sets, G U 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 $V S$

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

Note that increasing M increases the security of the system (namely, it decreases FAR (False Accept Rate) but it also has the potential to increase FRR (False Reject

Rate), and N is limited by the maximum number of feature points that can be extracted from the fingerprint. Along with VS, the polynomial degree D forms the final vault, V. Note that V can be transferred over insecure communication channels (e.g., Internet), stored in a local computer or a smart card, etc.

4.6 Decoding

Here, a user tries to unlock the vault V using the query minutiae features. Assuming that we have N (note that this number is the same as the number of genuine template minutiae in order to balance the complexity, e.g., measured via the number of required access attempts to reveal the secret for genuine and imposter users) 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 $K \leq N$.

Now, 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. Hence, we end up with ${}^{k}C_{D+1}$ combinations. For each of these combinations, we construct the Lagrange interpolating polynomial. For a specific combination set given as

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

the corresponding polynomial is

$$p^*(u) = \frac{(u-v_2)(u-v_3)\dots(u-v_{D+1})}{(v_1-v_2)(v_1-v_3)\dots(v_1-v_{D+1})}w_1 + \frac{(u-v_1)(u-v_3)\dots(u-v_{D+1})}{(v_2-v_1)(v_2-v_3)\dots(v_2-v_{D+1})}w_2 + \dots$$
$$\dots + \frac{(u-v_1)(u-v_2)\dots(u-v_D)}{(v_{D+1}-v_1)(v_{D+1}-v_2)\dots(v_{D+1}-v_D)}w_{D+1}.$$

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

$$P^*(u) = c_8^* u^8 + c_7^* u^7 + \dots + c_1^* u + c_0^*$$

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, $G_{CRC(a)} = a^{16} + a^{12} + a^5 + 1$. 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 list Q overlaps with template minutiae list 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.

Note that CRC is an error detection method, and it does not leak any information that can be utilized by an imposter attacker (e.g., Bob). He cannot learn which one of the polynomial projections is wrong; hence, he cannot separate genuine points from chaff points.

5. Experiments and Results

In this section we discuss about the experimental setup and the way we performed our experiments along with the challenges that we faced during the implementation and the way we overcame those challenges.

5.1 Experimental Analysis

The first step in our implementation is to extract features from the fingerprint. To facilitate the process, we have used filtering to enhance the fingerprint template. First of all, we did histogram equalization and normalization. After that we tried FFT filtering and Gabor filtering. After performing the experiment on a set of fingerprints, we found that Gabor filter gave better results and a much smoother fingerprint. Though Gabor filtering takes considerable amount of time to run, we have chosen it as it gives superior results and efficiency can be improved with better algorithm and hardware. Minutiae extraction becomes a trivial task after filtering. We have used matlab functions to perform binarization and thinning of the fingerprint template, followed by the extraction of bifurcation and end point minutiae. Post-processing has been done to remove the false minutiae. The minutia points near the segmentation boundary were removed and mintia ponts that were very close to each other were also ignored.

Then we tried out various methods for extracting the core point of the fingerprint. Extracting the core is one of the most challenging tasks and many methods have been proposed earlier. We implemented the optimal core point technique [37] and the virtual core point technique [31]. While the optimal core point technique gave good results, but was not accurate and failed for certain types of fingerprints, like the arch type. The virtual core point technique didn’t give the exact core point, but some point

which was claimed to remain steady across various impressions of the same person. But its accuracy was very poor and hence couldn't be used in our case. After going through many different methods, we devised our own method to find the core point. The detail of our method has been discussed in detail in 4.4.1. This was a major achievement, as we can now find out the core point in all types of fingerprints with complete accuracy. The core point was detected correctly in all the fingerprint templates that we used for our experiments.

After detecting the core point, we have to align the fingerprint templates. For that, we transformed the minutiae points to a new coordinate system, with core point as the origin. To find an axis, we used the orientation field flow curves and marked the highest curvature point on each of the flow curves. We then used linear regression to find a line passing through all the high curvature points which are near core point (distance<30). Further, learning has been applied to optimize the linear regression by removing the points far away from the line, one by one. By this process, we have obtained one of the axes and the origin for the new coordinate system. The translation and the rotation parameters were calculated and all the minutia points were transformed to the new coordinate system. We tried both Cartesian system as well as the polar coordinate system. In the Cartesian system, we used a quantization factor of 10 in both x and y direction. Whereas in the polar system, we divided the angle into 16 parts and non-linear quantization (μ -law) was applied to the distance. After some experiments, we set μ to 1.25 as that was the optimum value.

The two coordinates were concatenated, 8 bits each in Cartesian system and distance (9-bits) & angle (7-bits) in polar system. The value is then converted to GF(2^{16}). We have used a random number generator to obtain the 128-bit secret key which has to encoded. The 128-bit key is appended by a 16-bit CRC code for error

detection. The total 144 bit number is then divided into 9 parts of 16-bits each and these 9 parts become the coefficients of the polynomial. All the values are then converted to GF(2^{16}) and further calculations are performed in this field. The transformed minutia points obtained above are applied on this polynomial to obtain the (u,v) pairs. Then random chaff points are added to the set to make it 1000 pairs. The chaff points are selected in such a manner that they do not lie on the polynomial and do not correspond with the minutia points. The list is then scrambled to obtain the fuzzy vault which can be sent through an unsecure network.

During decoding all the steps are same until the extraction of the transformed minutiae features. After that, the obtained values are compared with the abscissa values in the vault and the (u,v) pairs where a match occurs, are taken out. Lagrange interpolation is then applied to reconstruct the polynomial. All the calculations till now are done in galois field. CRC checking is then performed and if there are no errors detected, the key has been successfully retrieved.

5.2 Experimental Setup

We used the publicly available database, DB1 and DB2 of FVC 2002 [7] and the UPEK fingerprints [33]. DB1 and DB2 had 8 impressions of 10 different people each. The UPEK fingerprints had 8 impressions of 16 different people. We used the fuzzy vault to encode a 128 bit key, with CRC-16 encoding for error detection. We have included 1000 chaff points in our implementation. The testing environment:

OS used: Windows XP, SP2

RAM: 1.25 GB

Platform: Matlab

We randomly selected one fingerprint of each person for encoding. The fingerprint is usable only if has more than 15 minutia points, otherwise it is rejected. After that, we randomly selected two more fingerprints for decoding the vault. Thus, we performed experiments on a total of 72 combinations of fingerprints. We also tried to unlock the vault by using many fingerprints of different people to see if there are any false accepts.

5.3 Results

Out of the 40 combinations in in FVC2002 database that we tried, the vault was opened successfully on 26 occasions when Cartesian system was used and 28 occasions when polar system was used. Thus, it gives a genuine accept rate (GAR) of 70%. We also tried to open the vault with other fingerprints. Many such combinations were tried, but the vault was not opened by a wrong fingerprint. Thus the False accept Rate can be claimed to be almost zero with a very high probability. Also the false reject rate (FRR) is 30%. The fingerprint images were not of very high quality, but still we have got results comparable to [21].

In UPEK database we had 32 genuine combinations, out of which the vault was successfully opened on 24 occasions in case of Cartesian system and 26 occasions in case of Polar system. Thus we got a Genuine accept rate (GAR) of 81.25%. And a False reject rate (FRR) of 18.75%. As in the previous case there were no false accepts. The fingerprint images were of better quality than the FVC database. Better results can be obtained if we carry out the experiments on a large database. We can also see that there is a slight improvement in the GAR when we used Polar system with non uniform quantization. Significant improvements can be seen if we perform the experiment on a larger database.

Apart from that, we also performed some experiments on live fingerprints using the fingerprint sensor in the laboratory. In most of the cases, the vault was successfully decoded with the fingerprint of the same person.

Database	FVC 2002 (DB1 and DB2)	UPEK Database
No. of fingerprints	20 X 8	16 X 8
No. of genuine combinations tried	40	32
No. of times vault opened successfully	28	26
Genuine Accept Rate (GAR)	70%	81.25%
False Reject Rate (FRR)	30%	18.75%
False Accept Rate (FAR)	0	0

Table 5.1 Experimental Results

We have achieved significant results without the need for transmitting any helper data. This was possible mainly because of the new technique that we developed for detecting the core point. This, along with the optimizations done while finding out the axis has provided us with these results.

The false rejects were due to (i) improper alignment (ii) very poor quality images where the region near core point was completely distorted, and (iii) number of common minutiae between locking and unlocking prints is less than the required number. Note that majority of these errors can be eliminated with increased user cooperation and habituation. Moreover, with the latest sensors, we can have fingerprint

images of much better quality. The fuzzy vault worked well for all the fingerprints. The errors were due to the fingerprint features only. We must also point out that the failure to enroll rate for fingerprint biometric can be as high as 4%: namely, nearly 4% of the population may not generate acceptable fingerprint images (either during vault encoding or decoding) with current sensors. Hence, the figures that we cite above implicitly assume that acceptable fingerprints can be acquired (ie., for approximately 96% of the population).

6. Conclusion 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 have used the high curvature points and the core point to obtain a new coordinate system, to which the minutia points are transformed. Evaluation on a public-domain fingerprint database demonstrates that our implementation has a very good GAR and a very low FAR.

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. Apart from this, we can use the biometric features obtained from the fingerprints to generate secret keys. It would be possible if we can obtain the same invariant features from the fingerprint every time. A better quantization strategy could also be devised so that we can minimize the errors due to points near the quantization boundaries.

References

[1] Arathi Arakala, Jason Jeffers, and K.J. Horadam: "Fuzzy Extractors for Minutiae-Based Fingerprint Authentication". In Springer-Verlag Berlin Heidelberg 2007.

[2] S. Dass and A. K. Jain: "Fingerprint classification using orientation field flow curves". In *Proc. Indian Conference on Computer Vision, Graphics and Image Processing*, pages 650-655, 2004.

[3] S. C. Dass. "Markov random field models for directional field and singularity extraction in fingerprint images." *IEEE Transactions on Image Processing*, 13(10):1358-1367, October 2004.

[4] A. K. Jain and U. Uludag. "Hiding biometric data." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1494-1498, November 2003.

[5] A. Juels and M. Sudan. "A fuzzy vault scheme." In A. Lapidoth and E. Teletar, editors, *Proc. IEEE International Symposium on Information Theory*, page 408, 2002.

[6] A. Juels and M. Wattenberg. "A fuzzy commitment scheme." In G. Tsudik, editor, *Proc. Sixth ACM Conference on Computer and Communications Security*, pages 28-36, 1999.

[7] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain. "FVC2002: Second Fingerprint Verification Competition." In *Proc. International Conference on Pattern Recognition*, pages 811-814, 2002.

[8] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain. "FVC2004: Third Fingerprint Verification Competition." In *Proc. International Conference on Biometric Authentication (ICBA)*, pages 1-7, 2004.

[9] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar. "*Handbook of Fingerprint Recognition*, Second Edition, Springer, New York, 2009".

[10] N. K. Ratha, J. H. Connell, and R. M. Bolle. "Enhancing security and privacy in biometrics-based authentication systems." *IBM Systems Journal*, 40(3):614-634, 2001.

[11] R. Snelick, U. Uludag, A. Mink, M. Indovina, and A. K. Jain. "Large-scale evaluation of multimodal biometric authentication using state-of-the-art systems." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):450-455, March 2005.

[12] W. Stallings. "*Cryptography and Network Security: Principles and Practices*. Prentice Hall, New York, Third edition, 2003".

[13] 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.

[14] S. Yang and I. Verbauwhede. "Automatic secure fingerprint verification system based on fuzzy vault scheme." In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 609-612, 2005.

[15] B.G.Sherlock, D.M.Monro, and K.Millard. "Fingerprint enhancement by directional fourier filtering." In *Visual Image Signal Processing*, volume 141, pages 87–94, 1994.

[16] Karthik Nandakumar, *Student Member, IEEE,* Anil K. Jain, *Fellow, IEEE,* and Sharath Pankanti, *Senior Member, IEEE.* "Fingerprint-Based Fuzzy Vault: Implementation and Performance." *In IEEE transactions on information forensics and security, Vol.2, No. 4,December 2007.*

[17] Johannes Merkle, Matthias Niesing, Michael Schwaiger, Heinrich Ihmor, Ulrike Korte. "Performance of the Fuzzy Vault for Multiple Fingerprints." *In arXiv:1008.0807v5 [cs.CR] 29 Nov 2011*.

[18] 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.

[19] 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.

[20] Sungju Lee, Daesung Moon and Yongwha Chung. "Inserting Chaff Minutiae for the Geometric Hashing-based Fuzzy Fingerprint Vault." *In Journal of information science and engineering 25, 1177-1190 (2009).*

[21] 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.

[22] 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.

[23] 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.

[24] Feng Hao, Ross Anderson, and John Daugman: "Combining Crypto with Biometrics Effectively." *In IEEE transactions on computers, Vol. 55, No. 9, September 2006.*

[25] Manvjeet Kaur, Mukhwinder Singh, Akshay Girdhar, and Parvinder S. Sandhu: "Fingerprint Verification System using Minutiae Extraction Technique." *In World Academy of Science, Engineering and Technology 46 2008.*

[26] Haris Papasaika-Hanusch, Institute of Geodesy and Photogrammetry, ETH Zurich: "Digital Image Processing Using Matlab."

[27] "MATLAB 6.5 Image Processing Toolbox Tutorial."

[28] Atipat Julasayvake and Somsak Choomchuay: "An Algorithm for Fingerprint Core point detection." 1-4244-0779-6/07/$20.00 ©2007 IEEE.

[29] Piotr Poewik and Lukasz Wieclaw: "A new approach to reference point location in fingerprint recognition." *In IEICE Electronics Express, Vol.1 , No.18, 575-581.*

[30] Yang Xu , Xuedong Zhang: "Gabor Filterbank and Its Application in the Fingerprint Texture Analysis." *Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05).*

[31] Sarnali Basak, Md. Imdadul Islam, M. R. Amin: "Detection of Virtual Core Point of A Fingerprint: A New Approach." *International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-2, May 2012.*

[32] Luigi Rosa: "Core Point Detection Using Orthogonal Gradient Magnitudes of Fingerprint Orientation Field."

[33] UPEK Fingerprint Database, http://www.advancedsourcecode.com/PNGfingerprint.rar, 2012.

[34] Shivang Patel, "Fingerprint Verification System," http://fvs.sourceforge.net/fingerprint_bitmaps.zip , 2004.

[35] Ashish Mishra and Dr.Madhu Shandilya: "Fingerprint's Core Point Detection using Gradient Field Mask." *International Journal of Computer Applications (0975 – 8887) Volume 2 – No.8, June 2010.*

[36] H B Kekre and V A Bharadi: "Fingerprint Core Point Detection Algorithm Using Orientation Field Based Multiple Features." *International Journal of Computer Applications (0975 - 8887) Volume 1 – No. 15*

[37] Navrit Kaur Johal, Prof. Amit Kamra: "A Novel Method for Fingerprint Core Point Detection." *International Journal of Scientific & Engineering Research Volume 2, Issue 4, April-2011 1 ISSN 2229-5518.*

[38] Xunqiang Tao, Xin Yang, Kai Cao, Ruifang Wang, Peng Li and Jie Tian: "Estimation of fingerprint orientation field by weighted 2D fourier expansion model." *2010 International Conference on Pattern Recognition.*

[39] Peng Li, XinYang, KaiCao, XunqiangTao, RuifangWang, JieTian: "An alignment-free fingerprint cryptosystem based on fuzzy vault scheme." *Journal of Network and Computer Applications 33 (2010) 207–220.*

[40] Jianjie Li, Xin Yang, Jie Tian_, Peng Shi, and Peng Li: "Topological Structure-based Alignment for Fingerprint Fuzzy Vault." 978-1-4244-2175-6/08/$25.00 ©2008 IEEE.

[41] Johannes Merkle, Matthias Niesing, Michael Schwaiger, Heinrich Ihmor, Ulrike Korte. "Security Capacity of the Fuzzy Fingerprint Vault."

[42] Gregory C. Sharpy, Sang W. Leey, and David K. Wehez: "ICP Registration using Invariant Features."

[43] C. Soutar, D. Roberge, S. A. Stojanov, R. Gilroy, and B. V. K. Vijaya Kumar. "Biometric encryption - enrollment and verification procedures." In *Proc. SPIE, Optical Pattern Recognition IX, vol. 3386*, pages 24-35, 1998.

[44] C. Soutar, D. Roberge, S. A. Stojanov, R. Gilroy, and B. V. K. Vijaya Kumar. "Biometric encryption using image processing." In *Proc. SPIE, Optical Security and Counterfeit Deterrence Techniques II, vol. 3314*, pages 178-188, 1998

[45] F. Monrose, M. K. Reiter, and S.Wetzel. "Password hardening based on keystroke dynamics." In *Proc. 6th ACM Conference on Computer and Communications Security*, pages 73-82, 1999.

[46] G. I. Davida, Y. Frankel, and B. J. Matt. "On enabling secure applications through on-line biometric identification." In *Proc. 1998 IEEE Symposium on Privacy and Security*, pages 148-157, 1998.

[47] Y. Dodis, L. Reyzin, and A. Smith. "Fuzzy extractors: how to generate strong keys from biometrics and other noisy data." In *Proc. International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 523-540, 2004.

[48] 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.

[49] Lin Hong, Yifei Wan, and Anil K. Jain. "Fingerprint image enhancement: Algorithm and performance algorithm." In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 777–789, May 1998.

[50] Ratha, N., Chen, S., and Jain, A. "Adaptive flow orientation based feature extraction in fingerprint images." *Pattern Recognition 28*, 11 (1995), 1657–1672.

[51] Christian Rathgeb and Andreas Uhl. "A survey on biometric cryptosystems and cancelable biometrics." In *Rathgeb and Uhl EURASIP Journal (a springer open journal) on Information Security 2011, 2011:3.*

[52] N. Lalithamani and K.P. Soman. "Irrevocable Cryptographic Key Generation from Cancelable Fingerprint Templates: An Enhanced and Effective Scheme." In *European Journal of Scientific Research, ISSN 1450-216X Vol.31 No.3 (2009), pp.372-387.*

[53] D. V. Klein. "Foiling the cracker: a survey of, and improvements to Unix password security." In *Proc. United Kingdom Unix User's Group*, 1990.

[54] Jain AK, Nandakumar K, Nagar A: “Biometric template security.” *EURASIP J Adv Signal Process 2008, 1-17*.

[55] Jain, A. K., and Farrokhnia, F. “Unsupervised texture segmentation using Gabor filters.” *Pattern Recognition 24*, 12 (1991), 167–186.