A new design approach of Fuzzy Hashing schemes for Fingerprints

Student Name: Surabhi Garg

IIIT-D-MTech-CS-IS
June, 2015

Indraprastha Institute of Information Technology
New Delhi

Thesis Committee
Dr. Donghoon Chang, IIIT Delhi (Chair)
Dr. Somitra Sanadhya, IIIT Delhi
Dr Praveen Gauravaram, QUT, Australia

Submitted in partial fulfilment of the requirements
for the Degree of M.Tech. in Computer Science,
with specialization in Information Security

Keywords: Fuzzy hashing, Minutiae, Voronoi diagram, Fuzzy Extractor

Certificate

This is to certify that the thesis titled **"A new design approach of Fuzzy Hashing schemes for Fingerprints"** submitted by **Surabhi Garg** for the partial fulfillment of the requirements for the degree of *Master of Technology* in *Computer Science & Engineering* is a record of the bonafide work carried out by her under our guidance and supervision in the Security and Privacy group at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Dr. Donghoon Chang
Indraprastha Institute of Information Technology, New Delhi

Abstract

Fuzzy hashing is a technique commonly used to compare two different data files which share identical sets of bits in same order, by determining the level of similarity between them. This is done by generating same hash output for the similar files (similar in terms of bits). We provide a new design approach of Fuzzy Hashing schemes, for protecting fingerprint templates of individuals from being misused by attackers when stored on servers or any Cryptographic module by introducing a binary representation for templates. The representation scheme outputs a bit string from the unique features present on human fingertip and is non-invertible, invariant to change in position of fingerprint, and captures most of the discriminatory information which can define a fingerprint. As a particular individual has unique set of features known as minutiae, the bit string generated is unique for every individual. The user is authenticated through simple hamming weight calculation between two bit strings representing fingerprint templates. We then mapped this bit string obtained from design approach to a fuzzy extractor which is used to generate strong, secure Cryptographic key that is used for encrypting personal data.

Acknowledgments

I would like to express my deepest gratitude to my advisor Dr.Donghoon Chang for his guidance and support. The quality of this work would not have been nearly as high without his well-appreciated advice. I would also like to thank Dr. Somitra Sanadhya, Neha Gupta, Sweta Mishra and Anush Sankaran for devoting their time in discussing their ideas with me and giving their invaluable feedback. I would like to dedicate this thesis to my loving and supportive parents who have always been with me, no matter where I am.
I thank Dr. Praveen Gauravaram for accepting to be a part of my thesis committee as the external examiner, and for enriching this thesis with his valuable suggestions and feedback.

Contents

1 Introduction **1**

2 Foundations and definition **4**

3 Existing schemes for protecting fingerprint templates and their limitations **7**

3.1 Fuzzy Commitment scheme and its limitations 7

3.2 Fuzzy Vault scheme and its limitations 8

3.3 Existing Fuzzy hashing schemes and their limitations 8

4 Proposed design approach of Fuzzy Hashing **11**

4.1 Minutiae Extraction 13

4.2 Voronoi diagram generation 13

4.3 Intermediate String Generation 13

4.4 Grouping and Shingling 15

4.5 Output hash generation using Bloom filter 16

4.6 Security Analysis 17

5 Design Implementation **18**

6 Applications of our design approach **23**

6.1 User authentication 23

6.1.1 Enrollment phase 23

6.1.2 Authentication phase 24

6.2 Key generation 25

7 Conclusion and Future work **27**

A Minutiae extraction from fingerprint template **30**

List of Figures

2.1 Fuzzy extractor construction from Secure Sketch [3] 6

4.1 Proposed Design approach block diagram 11

5.1 Pre-Processing steps for Fingerprint Template 19
5.2 Voronoi diagram generated from Fingerprint 20
5.3 Distance computation within a Voronoi cell 20
5.4 String generated using Voronoi diagram 20
5.5 Set of elements in each group 21
5.6 Final set of elements after performing shingling 22
5.7 Bloom filter output represented in form of bits 22

6.1 User authentication using output of our proposed design approach [3] 24
6.2 Key Generation using Fuzzy Exractor [3] 26
6.3 Protecting personal data using key generated from Fingerprint template [3] . . . 26

List of Tables

6.1 User Authentication results 25

Chapter 1

Introduction

Human verification and identification using biometric data is an active topic of research these days. Passwords are very common means of authentication during logging in on any site or during banking transactions etc. However, passwords are not a secure solution as they need to be memorized by human beings which is very difficult if passwords are long. Also, an attacker can guess a password easily. With these limitations, authentication using biometric data came into existence. Among that fingerprint recognition is the most widely used. Fingerprint recognition is a method of verifying a match between different fingerprint templates. A fingerprint sensor is used to capture a digital image of the fingerprint pattern. Fingerprints are unique for every individual and do not require any kind of memorization like passwords. A fingerprint is represented as a pattern of ridges and valleys visible on the fingertip. The ridges are the foreground structure while valleys are the background structure of the fingerprint.The ridges of the finger form the minutiae points: the ridge endings (terminals of ridge lines) and the ridge bifurcations (fork-like structures) which represents unique features for every individual's fingerprint.

Many techniques have been proposed for Fingerprint recognition with the help of minutiae that are extracted from fingertip. The location and orientation of minutiae represents a unique fingerprint. In the present techniques, minutiae of a particular individual's fingerprint template are stored on system servers, or any Cryptographic security module (smartcard or USB token) during enrollment phase in the raw form (i.e. its coordinates and orientation) or a vector is constructed from these unique features which is then used to represent fingerprint template. Another technique includes storing distances between a pair of minutiae or some other feature derived from minutiae on server or Cryptographic module. If the distances are similar between two templates, i.e. if the set of distances obtained from two fingerprint templates are equal, then matching is successful. Most of the techniques require alignment between two fingerprint templates in order to perform matching between them which needs storage of original fingerprint template during enrollment. However, Storing fingerprint templates in raw form for fingerprint recognition is not secure as it can be stolen easily by an attacker. Also, Minutiae positions and count are not exactly same due to different pressure and surface tension applied on sensor during fingerprint capturing. Thus these features will not be exactly similar for same individual and matching is not completely accurate. Camera is also used in many schemes for Fingerprint

template capturing. However, extraction of unique features from template is not accurate.

For fingerprint template protection, few techniques encrypt fingerprint data using hash functions or a long key. Storing hashed fingerprint data using a Cryptographic one-way hash function such as SHA-2 is not a good solution. As hash functions generate completely different output hash if input data is slightly modified, two different fingerprint templates which are close to each other but not exactly identical in terms of minutiae location will result in different output hashes and thus matching will not take place. Besides this, it is not a good approach to encrypt fingerprint data using a password-based encryption because guessing a password enables an attacker to decrypt the fingerprint data. However, it is mentioned because it is commonly used technique in practice to secure biometric database on server, even if it is not a secure practice. Encrypting our fingerprint data using difficult-to-remember high entropy secret key such as 128-bit random secret key is not a good option either, because the key has to be stored somewhere, and once the key is disclosed to an attacker, the fingerprint data can be decrypted easily.

Existing schemes for fingerprint template protection like Fuzzy Commitment and Fuzzy vault are proposed in [10] [3]. Fuzzy Commitment scheme is used to secure biometric data represented in the form of binary vectors. It uses Fuzzy Extractor and secure sketch concept for securing data and for user authentication. Strong key is generated using Fuzzy Extractor. In Fuzzy vault scheme, the basic idea is that Alice places a secret S in a vault and locks it using an unordered set A (e.g. minutiae of fingerprint). Bob uses an unordered set B to unlock the vault (and thus access S): successful if B and A overlap substantially.

Thus, there is a need of novel Fingerprint representation scheme which is sufficiently robust and satisfies three main requirements-

- Template security- pre-image resistant and non-invertible template
- Matching and accuracy- user can be authenticated with low false positive rate and low false negative rate.
- Revocability- An attacker cannot obtain the original biometric features of the user

Motivation: Fingerprint recognition for user authentication is an active topic for research in biometric as well as cryptographic areas. However, the potential vulnerability here is the leakage of biometric template information, which may lead to serious security and privacy threats. Existing fingerprint template protection techniques possess some challenges like capturing maximum discriminatory information from templates, representation of fingerprint in a way so that matching can be done easily using simple metric. The information obtained from representation scheme should be invariant to changes in finger position and other parameters like pressure and surface tension. Thus there is a need for secure representation scheme for fingerprint templates which does not reveal any information about original template and provides secure authentication.

Contribution: We propose a new design approach of Fuzzy Hashing scheme by representing fingerprint templates in a binary string. The aim of our approach is to take fingerprint template

as input and generate a bit string as output hash value, such that this bit string is similar for same individual and completely different for different individuals in terms of hamming weight. Using this binary string it is very easy to compare two fingerprint templates by calculating hamming distance between them, to find if they are similar or not without the need for actual fingerprint template, thus providing user authentication. We also construct the mapping between fuzzy hashing output- bit string and fuzzy extractor so as to generate secure Cryptographic key from fuzzy extractor.

Organization: The thesis is organized as follows. In Chapter 2 Foundations and definitions, we present the important preliminaries necessary for understanding the specification. This is followed by Existing schemes for protecting fingerprint templates and their limitations explained in Chapter 3.Our design approach is presented in Chapter 4. Chapter 5 presents design implementation. Subsequently, the applications of our design approach are described in Chapter 6. Finally, in Chapter 7 conclusion and future work are mentioned.

Chapter 2

Foundations and definition

The terminologies used in our thesis are discussed below.

- **Fuzzy hashing:** Hash functions are used in Cryptography to map arbitrary large input to a small fixed length output. The key feature of these cryptographic hash functions is that even if only one bit of the input is changed the output will be completely different and therefore similar files cannot be identified even if there is a change of a single bit. Fuzzy hashing, introduced by Jesse Kornblum 2006, is the ability to compare two different data files by determining the level of similarity between them. It allows matching between two files, which share identical sets of bits in same order, that may not be possible with the traditional hashing methods. This technique takes an input file in bits and generate output hash value such that the hash is same for similar files in terms of hamming distance or edit distance and is completely different for different files. ssdeep [9] is a fuzzy hashing technique proposed and is the most widely used technique in Digital forensics.

- **Voronoi diagram(VD):** It is named after Georgy Voronoy and it is the partitioning of a plane with N points into convex polygons such that each polygon contains exactly one point P, and every point in a given polygon is closer to its point P than to any other point. Each polygon is called as a Voronoi cell and the set of all Voronoi cells for a given point set is called a Voronoi diagram.
  VD has some distinct properties [7]

  - It is unique for a given point set
  - If some points are removed or inserted, the change will remain local and other cells would not be affected.

  Basically, Voronoi diagram is constructed by taking close pair of points together and then drawing a line perpendicular to the line that connects those two points of each pair such that the line drawn is equidistant from both the points. Voronoi diagram generation is explained in chapter 7.

- **Bloom filter:** A Bloom filter conceived by Burton Howard Bloom in 1970 is defined as representation of a set $S = \{x_1, x_2, \cdots, x_n\}$ of n elements from a large universe U using an array of m bits, initially all set to 0. It uses k independent hash functions $\{h_1, h_2, \cdots, h_k\}$ with range of size of bloom filter, such that for each element belongs to set S, the output of k hash functions indicating bit positions of filter are set to 1.

  Then, to check if an element is present in S, it is checked whether all bit positions corresponding to k hash functions applied on that particular element are set to 1. If not, then element is not a member of S. If all those bits are set to 1, we can assume that element belongs to set, and hence a Bloom filter may produce a false positive if 1 or more bits out of all those bits could be set due to some other element also.

  In case of uniform data distribution, the probability that a certain bit is set to one when an element is inserted in bloom filter is $1/m$, thus, the probability that a bit is zero is $1 - 1/m$. If n elements are inserted into the Bloom filter, the probability of a given bit position to be one is $1 - \big(1 - 1/m\big)^{kn}$. For false positive, all the k array positions need to be set to 1. Hence, the probability p for a false positive is [8]

$$p = \big(1 - \frac{1}{m}\big)^{kn} \approx e^{\frac{-kn}{m}}$$

- **BCH Code:** The Bose, Chaudhuri, and Hocquenghem (BCH) codes [4] are random error-correcting cyclic codes. For any positive integers $(m \geq 3)$ and $(t < 2m - 1)$, there exists a binary BCH code with the following parameters:

  Block length: $n = 2^{\big(m-1\big)}$

  Number of parity-check digits: $n - k \leq mt$

  Minimum distance: $dmin \geq 2t + 1$

  This is called as a t-error-correcting BCH code. This means that maximum t-errors can be correcting using these BCH code. Chapter 6 explains the use of BCH code in generating random codeword which is used to generate a secure value s from a given input w.

- **Secure sketch:** Defined by Y.Dodos [3] in 2008. Let M be a metric space with a metric known as hamming distance d which is defined as number of bit positions in which two strings differ. A Secure sketch is given by two procedures [3] sketching procedure$\big(SS\big)$ and recover procedure$\big(Rec\big)$, stated as follows:

  Sketch SS, takes input string w and a random codeword c and it returns another string s where s is secure and public. Here, s is a secure value as the probability to recover w is very less [3] if c is not known, inspite that s is public.

  Recover Rec, takes another string w' and string s. If t is the maximum number of errors in bit string that can be corrected, then if $d\big(w, w'\big) \leq t$, then $Rec\big(w', SS\big(w\big)\big) = w$, else if $d\big(w, w'\big) > t$, there is no guarantee for the value of the output of Rec.

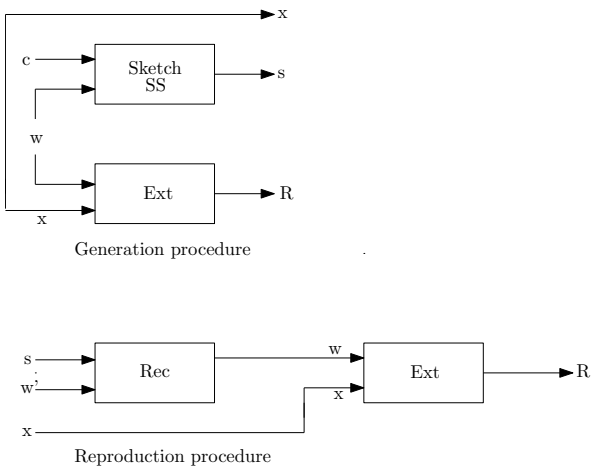

Figure 2.1: Fuzzy extractor construction from Secure Sketch [3]

- **Fuzzy extractor:** Defined by Y.Dodos [3] in 2008, A fuzzy extractor is a defined by two procedures Generation $\big(Gen\big)$ and Reproduction $\big(Rep\big)$. [3]

  Gen procedure consists of Sketch $\big(\mathrm{SS}\big)$ procedure of Secure sketch and a strong Extractor. A strong Extractor is one which outputs a highly uniformly distributed, randomized string which is independent of weakly random input string with low entropy [3]. A universal hash function can be taken as an example of Strong Extractor.
  Taking an input string w such that $w \in M$ and random codeword c, the Sketch will generate public string s. Also, The strong Extractor takes input w such that $w \in M$ and a random number x to output a uniform random string $R \in \big\{0,1\big\}$ which is used as a secure key. $\mathrm{P}\big(s;x\big)$ is taken as public information to be stored on server as given in our construction 6.

  Rep procedure consists of Recovery procedure of Secure Sketch that takes $w' \in M$ and s such that $d\big(w,w'\big) \leq t$ and it recover w which will be then given as input to Extractor along with x to output a string R, i.e. for all $w, w' \in M$ which satisfy $d\big(w,w'\big) \leq t$, if $Gen\big(w\big) \rightarrow \big(P,R\big)$, then we have $Rep\big(w',P\big) = R$.
  Construction of fuzzy extractor from secure sketch is shown in the Figure 2.1 [3]. Here, c is the random codeword generated from BCH codes. x is the random number generated using pseudo-random number generator. Further Chapter 6 explain the construction of Fuzzy Extractor by taking Fingerprint data as its input string.

Chapter 3

Existing schemes for protecting fingerprint templates and their limitations

Biometric template protection is a wide research area. Many schemes have been introduced to protect biometric templates from being stolen or modified by attackers. Also, biometric template stored on server should not reveal any information about an individual. Fuzzy extractors are used to convert less random biometric data into highly randomized strings, which allows cryptographic techniques to provide biometric template security. The first biometric system for template protection, Fuzzy commitment, where ordered biometric data is used to generate strong keys, was designed by Juels and Wattenberg. Later, Juels and Sudan introduced Fuzzy vault scheme which works on unordered biometric data and uses a ReedSolomon code. Also, few existing fuzzy hashing schemes like ssdeep, sdHash etc. can be used to hash the fingerprint templates but they have some limitations as explained below.

3.1 Fuzzy Commitment scheme and its limitations

In Fuzzy Commitment scheme, biometric input is taken in the form of binary string and a strong key is generated from the input data using Extractor. A Fuzzy Extractor turns noisy information into keys usable for any cryptographic application and also helps in Reliably securing and authenticating biometric data. For biometric input w, which represent any biometric data in binary form, a random error correcting codeword c is selected which can correct maximum t errors and a secure value s is obtained using secure sketch. When input w is given, which is close to w with errors less than t, c is obtained from it by xoring w with s which is then decoded to get c. Then, w can be recovered from secure sketch using recovery procedure of secure sketch.

Limitations of using Fuzzy commitment scheme are:

- In Fuzzy commitment scheme, Fuzzy extractor takes input in binary form, but there is no existing representation scheme that could represent fingerprint template in the form of binary string.
- Also, [3]provides mapping of edit distance to hamming distance with low distortions. With this mapping, edit distance between fingerprint minutiae can be mapped to hamming distance metric, thus representing fingerprint in binary form but the problem with this approach is that it is good only when t, the number of errors to be corrected in fingerprint template, is quite small which is not practically possible as biometric data is noisy. This means that two fingerprint templates from same individual are close in terms of location of unique features, but not exactly identical.

3.2 Fuzzy Vault scheme and its limitations

In Fuzzy vault scheme, Alice hides secret k using an unordered set A (e.g. minutiae of fingerprint). A polynomial $P(x)$ is selected which encodes k by using key elements as coefficients of polynomial. Elements of set A (Template minutiae) are plotted by evaluating the polynomial on each value of minutiae. After that large numbers of random chaff points are added to the computed values such that they do not lie on the polynomial $P(x)$ to form the fuzzy vault V. Chaff points are used to hide the genuine points (elements of set A) from the attacker. During decoding, Bob can only found out the secret key k only if provides sufficient number of points equal to that in set A. For that Bob will provide an unordered set B (Query minutiae). If the points in set B substantially overlap with points in set A then polynomial can be recreated and secret key k will be given to the user. Its limitations are:

- Fuzzy vault scheme assume that the template during enrollment and authentication are per-aligned, which is not a realistic assumption in practical fingerprint authentication systems.
- Chaff points are added in the scheme which are used to hide genuine fingerprint data from attackers. The more the chaff points, the greater is security but increasing Chaff points take huge memory space.
- Also, matching accuracy is degraded due to alignment issues and nonlinear distortions.

3.3 Existing Fuzzy hashing schemes and their limitations

A lot of fuzzy hashing techniques have been proposed to generate similar hash output for similar input data with few differences in the bits. Fuzzy hashing techniques are used to find similarity between similar looking files (semantically similar but not syntactically similar) which have very few dissimilar bits. All the existing fuzzy hashing techniques are used for digital forensic

background to identify similar files, or for malware detection. In forensics, the challenge of an forensic examiner is to identify a small file from a huge amount of dataset so that it can distinguish good and the bad files. The bad files are those which are modified by attacker in such a way that it looks very similar to the original bad file but when stored as a hash using Cryptographic hash-functions, its hash is completely different from the hash of original bad file due to the property of hash functions that a single bit change in the input will result in a completely different output hash value. Thus, it cannot be recognized as a bad file. Here, the idea of fuzzy hashing came into existence.

In fuzzy hashing, if two files are similar in terms of bits, having few bits modified, the resulting hash-value is almost same with high degree of similarity. Thus, fuzzy hashing is used to detect false positive and false negative introduced by attackers in digital documents by modifying only few bits. The existing fuzzy hashing techniques includes: In ssdeep [9], the main idea is to split the input into variable length blocks called segments or chunks. This is done by taking of window of 7 bytes and sliding it over the input file and a pseudo-random function is used which is calculated at every byte. If this function hits a given value, the end of a chunk is determined. Once whole input is broken into chunks, all these chunks are hashed independently using Cryptographic hash functions and their hash-values are added to final hash output. Thus, if a bit changes in input file, only the corresponding chunk or atmost 2 chunks will be affected, resulting in similar hash output. Sdhash introduced by [11], identify statistically improbable features from the whole input file and then use those features as input to bloom filter. The bloom filter output is then taken as output hash value. Similarly, bbHash [2] use external data structures called building blocks of the same length as window size. The window is slided over input file and hamming distance between all the building blocks and window content is calculated one by one. The offset of building block which has minimum hamming distance with window content in input file is appended to the final hash output.

The limitations of using these techniques for generating hashes for fingerprint templates are:

- All the above mentioned fuzzy hashing techniques and few others proposed in [2] work on the input file represented in binary form or ascii characters which can we easily processed further for generating hash output. But, fingerprint minutiae are represented in form of coordinate axis. Presently, work have been done on finding distances between pair of minutiae or between minutiae and core point [15] and taking the distance as unique features, but all these have limitations as the minutiae positions changes every time due to different pressure and surface tension applied on the surface of sensor. So, the probability that the same fingerprint has completely two different sets of distances is very high.

- These techniques are mainly proposed for digital forensics applications and work on assumption that only few bits of input file are modified by attacker in order to prevent semantic detection and thus the resulting hashes will be similar due to fuzzy hashing property. However, in fingerprints due to completely different set of minutiae for same individual, we cannot use existing techniques as hash output generated will be very different

because here not few, but large number of values that are different between two fingerprint templates of same individual at syntax as well as semantic level.

Thus, there is a need of protecting fingerprint template which are stored on biometric device. 3 fundamental challenges faced in protecting templates are-

- Automatically alignment of the fingerprints obtained during enrollment and matching, without revealing any information
- Appropriate representation scheme that is invariant to change in position of fingerprint, pressure applied on sensor and captures most of the discriminatory information.
- The fingerprint information obtained from original fingerprint template should be invariant to changes in finger position and other parameters like pressure and surface tension.

Chapter 4

Proposed design approach of Fuzzy Hashing

The new design approach for fingerprints generates output hash w, given the pre-processed Fingerprint Template i.e minutiae extracted in Appendix A as input data. The output hash is represented in the form of bit string. As our aim is to secure Fingerprint data from unauthorized access, the user authentication and key generation for secure communication is done without storing the actual fingerprint template on server. For making this to be possible, the design approach generates an output such that even if Fingerprint's minutiae set is not exactly similar for a particular individual, the final output generated using fuzzy hashing is similar so that matching could takes place using Hamming weight. That is why we call our design approach as a fuzzy hashing scheme approach for fingerprint template representation.

Figure 4.1 shows the complete block diagram of the design approach. The input is the minutiae points extracted using existing schemes and output is the binary string which will represent fingerprint template. The minutiae points are used to generate Voronoi diagram. Using novel scheme of generating string from cells of Voronoi diagram, a string is generated upon which grouping and Shingling is performed to increase the randomness in the fingerprint data. The set of elements obtained after Shingling is used as input given to the bloom filter. Bloom filter results in output string in binary form which represents fingerprint template of an individual.

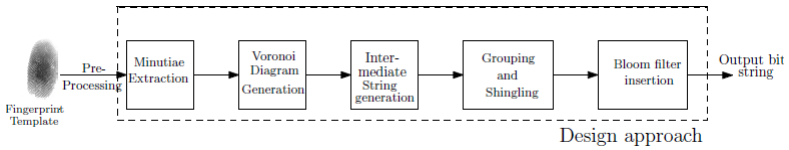

Figure 4.1: Proposed Design approach block diagram

Following is the step-by-step description of Algorithm 1 which explains our approach:

First we need to fix the following parameters:

V	:	Voronoi diagram generated from the minutiae points
m_i	:	Minutiae points present in i-th block
min	:	array consisting of Minutiae after removing false points
str	:	Unique string generated from Voronoi diagram
sh	:	set generated after performing grouping and 2-shingling
w	:	Final output hash generated from Algorithm 1

Algorithm 1: Output [w] Construction

Input: Pre-processed Fingerprint template with extracted minutiae.

Output: m-bit hash value w obtained from bloom filter

▷ Voronoi diagram generation: generates Voronoi diagram V from Input fingerprint template in steps 2 to 5

Divide input image into b blocks of size 32×32

for $i = 1$ to b

  min[i]=avg(m_i)

Generate V using min array. V consists of cell array having c cells

▷ Intermediate String generation: create string str in steps 7 to 10

for $i = 1$ to c

  Compute centroid (C_x, C_y) of each cell

  Compute distance d_x=$(min[i]_x - C_x)$, d_y=$(min[i]_y - C_y)$

Divide each cell into 4 regions and generate str using Algorithm 2.

▷ Grouping and Shingling: Generate g Groups within str, each Group labeled as $Group_g$ in steps 12 to 18 and create a set sh consisting of elements which are used as input to bloom filter in steps 19 to 21

$size(Group_g)$=number of str elements within the $Group_g$

set x=1

put str[1] under $Group_x$

for $j = 2$ to $c - 1$

  if intersection(cell[j],cell[j+1]) is not null

    put (str[j],str[j+1]) under same Group label as of str[j]

  else

    put (str[j+1]) under $Group_{++x}$

for $k = 1$ to l

  for $i = 1$ to $size(Group_k) - 1$

    $sh = sh.append(k \| str[i] \| str[i+1])$

▷ Output hash generation using Bloom filter: Generates hash w from bloom filter, bloom[m] having h hash functions, initially $bloom = 0$ in steps 23 to 27

for $i = 1$ to s

  for $j = 1$ to h

    $m = hash_j(i)$

    $bloom[m] \leftarrow 1$

$w = bloom$

The following sections describe complete design approach in detail:

4.1 Minutiae Extraction

Every fingerprint consists of minutiae on its tip which represent the unique features of fingerprint. These minutiae play an important role in fingerprint recognition for authentication purpose.Previously, in [6] [12], a lot of work has been done related to minutiae extraction from fingerprints for fingerprint matching. We used few previous algorithms [5] [12] to process the given fingerprint image in order to extract minutiae from fingerprints as explained in A. These minutiae are then used to generate Voronoi diagram which is explained in next section.

4.2 Voronoi diagram generation

After we get fingerprint minutiae from the pre-processing step, the fingerprint image with extracted minutiae is divided into blocks of size 32×32. The minutiae present in each block of image are averaged and the average of each minutiae in one block then represents the final minutiae points of whole fingerprint template by taking one minutiae from each block. This is done in order to remove false minutiae which are generated due to inaccuracies in template capturing via sensor or camera device and better feature recognition.

Voronoi diagram is then generated for every fingerprint image. For a particular individual, the fingerprint minutiae are unique. But due to pressure and surface tension irregularities while placing finger on fingerprint sensor, the minutiae may get dislocated and may change in number as some may get added or lost and thus they can be different for same individual. Matching is not possible with existing algorithms [7] [14] without aligning the actual image with the stored image template. In order to get unique features for every individual, we used Voronoi diagrams due to their properties as explained in Chapter 2.

In our implementation, we used MATLAB built in function voronoin to plot Voronoi diagram for given set of minutiae points. Thus, each cell of Voronoi diagram consists of exactly one minutiae point.

4.3 Intermediate String Generation

Instead of directly using minutiae from every Voronoi cell as a unique feature, we generate a string from the diagram. This is done because minutiae positions are not exactly same in each Voronoi cell of an individual even if Voronoi diagram is similar. In order to have an accurate measure of minutiae position, we proposed the concept of labelling each Voronoi cell as 4 different regions so that nearby minutiae points could be mapped to same label.

First we compute the centroid of each Voronoi cell. For each cell in Voronoi diagram, we find the distance between centroid of each cell and minutiae point of that particular cell. The centroid of a non-self-intersecting closed polygon/cell is defined by n-vertices [1] $\{(x_0, y_0), (x_1, y_1), \cdots, (x_{n1}, y_{n1})\}$ given by point (C_x, C_y), where

$$C_x = \frac{1}{6A}\sum_{i=0}^{n-1}\big(x_i + x_{i+1}\big)\big(x_i y_{i+1} - x_{i+1}y_i\big)$$

$$C_y = \frac{1}{6A}\sum_{i=0}^{n-1}\big(y_i + y_{i+1}\big)\big(x_i y_{i+1} - x_{i+1}y_i\big)$$

where A is the Area of polygon given by

$$A = \frac{1}{2}\sum_{i=0}^{n-1}\big(x_i y_{i+1} - x_{i+1}y_i\big)$$

As the Voronoi cells are unique for an individuals fingerprint, we use this property to compute unique string from each individuals fingerprint.

The string is computed as

If the distance computed between centroid and minutia is $\big(d_x, d_y\big)$, where d_x=$\big(min_x - C_x\big)$, d_y=$\big(min_y - C_y\big)$ then each Voronoi cell can be divided into four regions with four different labels, let say 1 $\big($representing right side of cell$\big)$, 2$\big($representing upper side of cell$\big)$, 3$\big($representing lower side of cell$\big)$ and 4$\big($representing left side of cell$\big)$. 0 represents the unbounded Voronoi cell.

Now, String str can be created using Algorithm 2. ***str:*** Intermediate string generation from Voronoi diagram

Algorithm 2: str generation

Input: Distance $\big(d_x, d_y\big)$ computed in Algorithm 1.
Output: Intermediate string str composed of 0,1,2,3 and 4

▷ Case 1: $\big(d_x > 0\big)$ and $\big(d_y > 0\big)$
if $\big(d_x > d_y\big)$, label the region as 1 else
    label the region as 2
▷ Case 2: $\big(d_x > 0\big)$ and $\big(d_y < 0\big)$
if $\big(d_x > d_y\big)$, label the region as 1 else
    label the region as 3
▷ Case 3: $\big(d_x < 0\big)$ and $\big(d_y < 0\big)$
if $\big(d_x > d_y\big)$, label the region as 4 else
    label the region as 3
▷ Case 4: $\big(d_x < 0\big)$ and $\big(d_y > 0\big)$
if $\big(d_x > d_y\big)$, label the region as 4 else
    label the region as 2
▷ Case 5: cell is unbounded
label the region as 0

Thus, for each fingerprint image, our proposed string composed of 0,1,2,3,and 4 is received. Figure 5.4 represents the string obtained from Voronoi diagram generated.

Now, this string will be used to generate fuzzy hash output represented by w which will be used as input to fuzzy extractor. However, this string has very low entropy. Thus, for introducing

randomness to get uniform random input from biometric data, we use bloom filter which require further processing of string.

Thus, string generation from Voronoi diagram in present invention can be generalized as follows: Given an input set of minutiae extracted from fingerprint template, Voronoi diagram is generated from minutiae as unique points. The general idea here is that even if minutiae points from two fingerprint template of same individual are slightly dislocated from their positions, and thus having different coordinates points, the data extracted from them must be identical, tolerating the errors in positions and number of minutiae. Hence, the Voronoi cell is further divided into sub-cells or regions so that minutiae points from different Voronoi diagrams which fall under same sub-cell of a particular cell are labelled with same value, thus improving the degree of accuracy in matching. For each cell in the Voronoi diagram, distance is computed between any distinct parameter of cell and its minutiae point. Based on the distance vectors, cell can be divided into various regions and thus for each cell, a label is assigned to each minutiae point based on its position in cell. These labels will form a string. However, as the entropy of this string is very low, we proposed grouping and shingling to increase randomness.

4.4 Grouping and Shingling

Grouping is done on the string as we found that for every individual, the pattern in which the cells are connected to each other is similar for Voronoi diagram of other fingerprints of same individual. This is due to the property that Voronoi diagram is similar for same set of points and as minutiae are similar, the cells are almost at similar positions for different Voronoi diagrams of same individual. Grouping is done by taking all the cells which are connected to one another, in a consecutive manner, under one group. Each group is indexed as 1, 2, and 3 and so on. If the intersection of edges between two consecutive Voronoi cells is not null, implies cells are connected, thus considered under same group index, else different group indexes are assigned to the two cells. The output of the Grouping step is different groups labelled with a unique group index each consisting of cell labels of the cells which are connected to each other consecutively.

Shingling is the process of constructing a set of unique, "continuous subsequences of characters" known as shingles in a string of size n. Here, c denotes the number of characters in each shingle in the set. Thus, for a string of size n, $\left(n - c + 1\right)$ c-shingles will be obtained after shingling. For each group, we generate 2-shingles, i.e. every shingle contains 2 characters. Group number is then appended to every shingle. Appending group number to the shingles increases the range of shingle values and uniqueness as grouping is unique for every individual. This set of elements combining grouping and shingling will serve as an input to our final step.

4.5 Output hash generation using Bloom filter

Bloom filters are widely used in large number of applications. As they are mostly used to find if an element is present in a set or not, we apply this property to our technique. For same input set of elements, bloom filter will produce same output string as hash functions perform mapping of input elements to the output string. The output of bloom filter is represented as w and is a string of 1 and 0. This string w will represent fingerprint template for every individual. Taking input as a set of elements $\big($obtained after appending group label on shingles $\big)$, we map every element to the bit position of bloom filter. The bit positions obtained from mapping are set to 1 in bloom filter. All other bits remain 0. Thus w is the final output string of 0s and 1s representing fingerprint.

In our work, bloom filter of size 256 bits is taken with all bits initially set to 0 and SHA1 hash function is used for mapping elements of input set to bloom filter array. For each element x of the input set, if number of hash functions used are k, for i from 0 to $k-1$, $SHA1\big(i\|x\big)$ is calculated and value is obtained in the range of bloom filter size by truncating hash output to 8 bits only. This output value defines the bit positions of bloom filter which is then set to 1 in bloom filter. All other bits remain 0. The final array of $0s$ and $1s$ is the bit string representing a fingerprint template uniquely.
We use bloom filter in our work because For same input set of elements, bloom filter produces same output string as hash functions perform mapping of input elements to the output string. Also, its Randomness is high thus high entropy. One cannot invert bloom filter output to get the original fingerprint template of an individual. Thus providing secure fingerprint template as an output.

Advantages of proposed representation scheme over other exisitng schemes are-

- The output obtained from the proposed scheme is in binary form, thus a fingerprint template is represented in a binary form which allows authenticating two individual through simple hamming weight metrics between their fingerprints templates represented in bit string, which is a novel technique of fingerprint recognition.

- No alignment is needed between two fingerprint templates during matching. It was earlier required by existing techniques and requires storing of original template or features in raw form. Any attacker can access the template or information from system causing leakage of biometric template information, which may lead to serious security and privacy threats.

- The fingerprint representation in bit string is non-invertible. No information can be revealed from the bit string to get original fingerprint template. Thus it is highly secure.

- Fuzzy commitment scheme was proposed for authenticating users through biometric data and it uses fuzzy extractor to generate a strong random secure key from input data. However, it required biometric data in the binary representation. No practical implementation of that scheme is proposed in existing work. With the proposed scheme, users can be eas-

ily authenticated through Fuzzy commitment scheme using hamming weight metric along with key generation.

- Fuzzy vault was also introduced for securing secret on system with the help of biometric data. However, it uses chaff points for hiding genuine data from attacker which takes a lot of space in terms of memory which is a disadvantage.

4.6 Security Analysis

In our thesis work implementation, we have used bloom filter of size 256 bits. Due to the small size of bloom filter, we are truncating the output of hash function to a fixed length which affects the overall security of the design approach. For n-bit output of hash function,the number of trials required to find collison are $2^{n/2}$. In our design implementation, we are taking only 8 bits out of 160 bits hash output which will take very less number of trails to get a collision for each hash function during insertion of shingle in bloom filter. Also, length of output of hash function will decide the size of bloom filter output. If the size of bloom filter is increased,the security will improve because the probability of occurence of collision will be small. But, larger the size of bloom filter, larger will be the output binary string, and thus, it will require a large space to store the string on server. Thus, there is always a security trade-off between storage space and time required to find a collision. So, we are not claiming bloom filter implementation parameters as secure parameters. We need to perform proper analysis through testing to get efficient values of parameters, which will be our first target for future work.

The output binary string generated from bloom filter depends on the number of inputs provided to bloom filter, the size of bloom filter array, and the hash functions used. For estimating the security of output binary string, we will test our scheme with proper fingerprint database.

Chapter 5

Design Implementation

We used Matlab to extract fingerprints minutiae from the original fingerprint image as given in Figure 5.1. We collected 47 Fingerprint images samples in total. From Each person, 3 templates were taken. The fingerprints were taken using Fulcrum Biometric FS80 Fingerprint sensor. To eliminate pressure applied on sensor and surface tension constraints, the sensor was improvised. A rubber band was wound around the surface to apply a uniform pressure on finger. Pre-processing steps used for minutiae extraction starts with histogram equalization and image enhancement. After enhancement, image cropping is done by cropping the image taking core point as its center. We used Matlab function 'imcrop' to crop the image with core point as refernce. Once the image is cropped, binarization and thinning is done on the image. Thinning is performed directly by using 'thin' function in Matlab. We use the code proposed by [12]for minutiae extraction from fingerprint. The fingerprint image after minutiae extraction is shown in Figure 5.1.

The fingerprint image may consists of false minutiae due to the irregular pressure applied by user and surface tension due to which some false minutiae may be added. In order to have an accurate measure of minutiae, we took average of all minutiae in a block of image. In Matlab, built in function 'voronoin' is used to generate Voronoi diagram from an image. The Voronoi diagram of give fingerprint template after removing false minutiae is shown in Figure 5.2. Here, each cell of voronoi diagram is labelled, cell 1 as $c1$, cell 2 as $c2$ and so on.

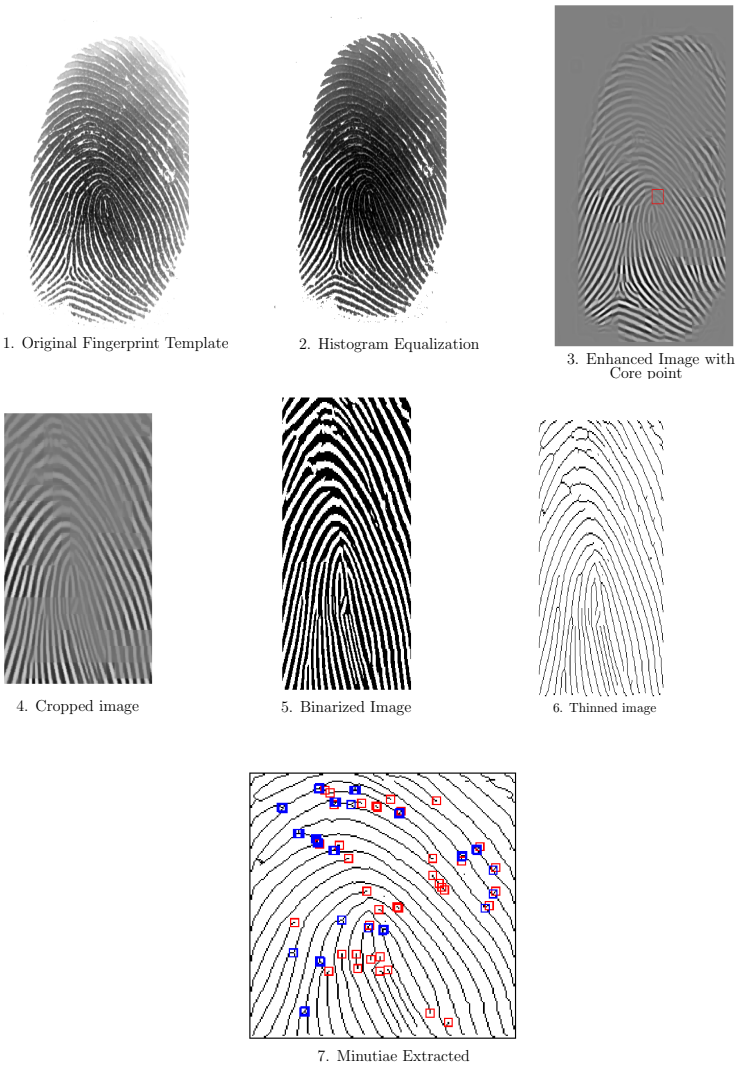

Figure 5.1: Pre-Processing steps for Fingerprint Template

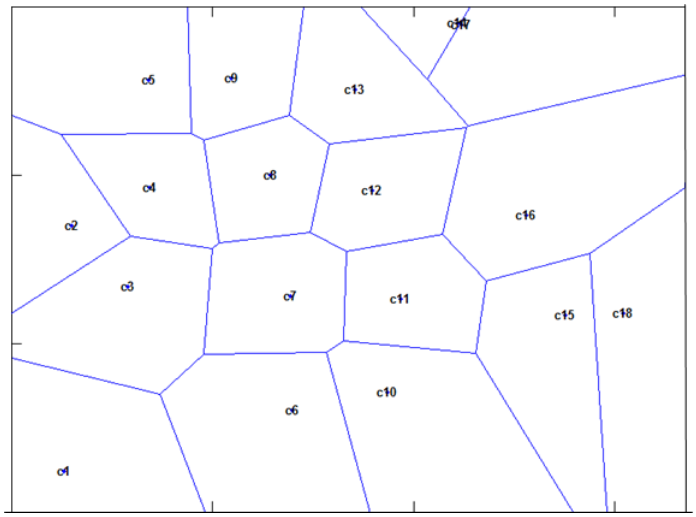

Figure 5.2: Voronoi diagram generated from Fingerprint

Now, on each cell, we apply our proposed Intermediate string generation algorithm 4, and the output string we got is represented as given in Figure 5.4.

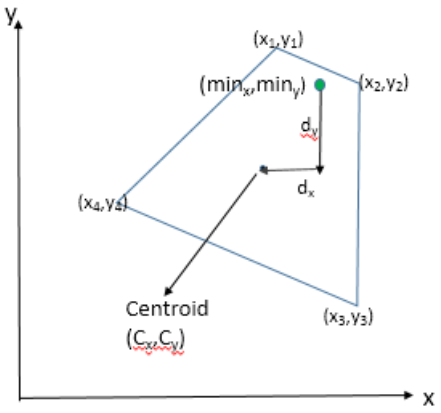

Figure 5.3: Distance computation within a Voronoi cell

As shown in Figure 5.3, the distance d_x, d_y is computed between minutiae and centroid of a particular cell. Based on distance computation, the string is generated using Algorithm 2.

0 0 4 2 0 1 3 4 2 1 4 4 2 0 4 4 2 0

Figure 5.4: String generated using Voronoi diagram

Now, for grouping, we find intersection between every consecutive pair of Voronoi cells from the Voronoi diagram. The pair of cell is connected if intersection is not null. Here in Figure 5.2, Cell $c1$ is not connected to next cell $c2$. Similarly, , cell $c2$ is connected to

its next cell $c3$, $c3$ to $c4$ and $c4$ to $c5$ thus, creating a group. Cell $c6$, $c7$, $c8$ and $c9$ are connected consecutively forming another group. Cell $c10$, $c11$, $c12$, $c13$, $c14$ are connected consecutively forming another group. Cell $c15$, $c16$ and $c17$ are connected consecutively forming another group. Cell $c18$ is not connected to any consecutive cell before or after. Thus, we get one connected set of cells under one group, next connected set of cells under second group and so on. The groups then consist of cell labels derived from intermediate string generation of each cell under them as described in Figure 5.5. After grouping is done, we apply shingling to each group. Here, in our implementation we applied 2-shingling i.e. 2 characters join together to form a shingle. So, for each group, shingles are created from the cell labels. Then a unique Group number is appended to all the shingles for each group to get the final set of elements as shown in Figure 5.6.

```
Group 0
0
Shingles for this Group

Appending group number


Group 1
0    4    2    0
Shingles for this Group
  04  42  20
Appending group number
  204  242  220

Group 2
1    3    4    2
Shingles for this Group
  13  34  42
Appending group number
  313  334  342

Group 3
1    4    4    2    0
Shingles for this Group
  14  44  42  20
Appending group number
  414  444  442  420

Group 4
4    4    2
Shingles for this Group
  44  42
Appending group number
  544  542

Group 5
0
Shingles for this Group

Appending group number
```

Figure 5.5: Set of elements in each group

204, 242, 220, 313, 334, 342, 414, 444, 442, 420, 544, 542

Figure 5.6: Final set of elements after performing shingling

In order to have low false positive rate in bloom filter, by default let 0.01, we used 9 hash functions to map each input element to bit position of bloom filter.
No. of hash functions $k = (m/n) * ln(2)$ where,
m= bloom filter array size in bits
n= number of elements inserted
For false positive rate 0.01, $k = 9$ if n is approximately 20.
Every element we get after Shingling and grouping step is given as input to bloom filter and output is given in the form of 0 and 1 as shown in Figure 5.7

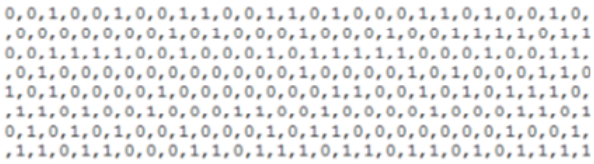

Figure 5.7: Bloom filter output represented in form of bits

This w is generated as an output from Fingerprint template.

Chapter 6

Applications of our design approach

In the previous chapter 4, on providing Fingerprint template as input, we generated fuzzy hash output w using our design approach. This output is similar for same individual and is completely different for different individual. Also, it is not possible to recover original fingerprint template from hash output w. In [3], Fuzzy Extractor and Secure Sketch was introduced to generate secure uniform random key. However, it does not provide any practical implementation on using Fuzzy Extractors and Secure Sketch to generate key from Biometric data. Using output in binary form, applications of our design approach as described in [3] are:

6.1 User authentication

The fingerprint template constructed from our representation scheme is similar for same individual and is completely different for different individual in terms of hamming weight. So, a user can be authenticated using binary string as a unique template. As proposed by [1], the Fuzzy commitment scheme example, is used for user authentication purpose. 0n providing input w, which we obtained, a random error correcting codeword c is selected and a secure value s is obtained using secure sketch. When input w obtained using another fingerprint template using our design approach is given which is close to w with errors less than t, the c obtained from it by xoring w with s which is then decoded to get c. Then, w can be recovered from secure sketch using recovery procedure of secure sketch.

6.1.1 Enrollment phase

During enrollment, user will provide w, input representing fingerprint. Using a message M generated from k bits, codeword c is generated using BCH codes. These k bits are generated randomly using randi function in Matlab which is used to generate Uniformly distributed pseudorandom integers. We implement BCH codes as error correcting codes instead of using simple codes such as hamming code as used in because we need to correct multiple bit errors. We can easily control the number of errors to be corrected using codeword. Also BCH codes are very

easy to decode using syndrome decoding method. Then, we create a value s by xoring w with c as

$s = w \oplus c$

We create a random number x of size same as size of s using random number generator. This random number x along with value s and $hash(c)$ is then stored in database on server side for each user with $User\_id$.

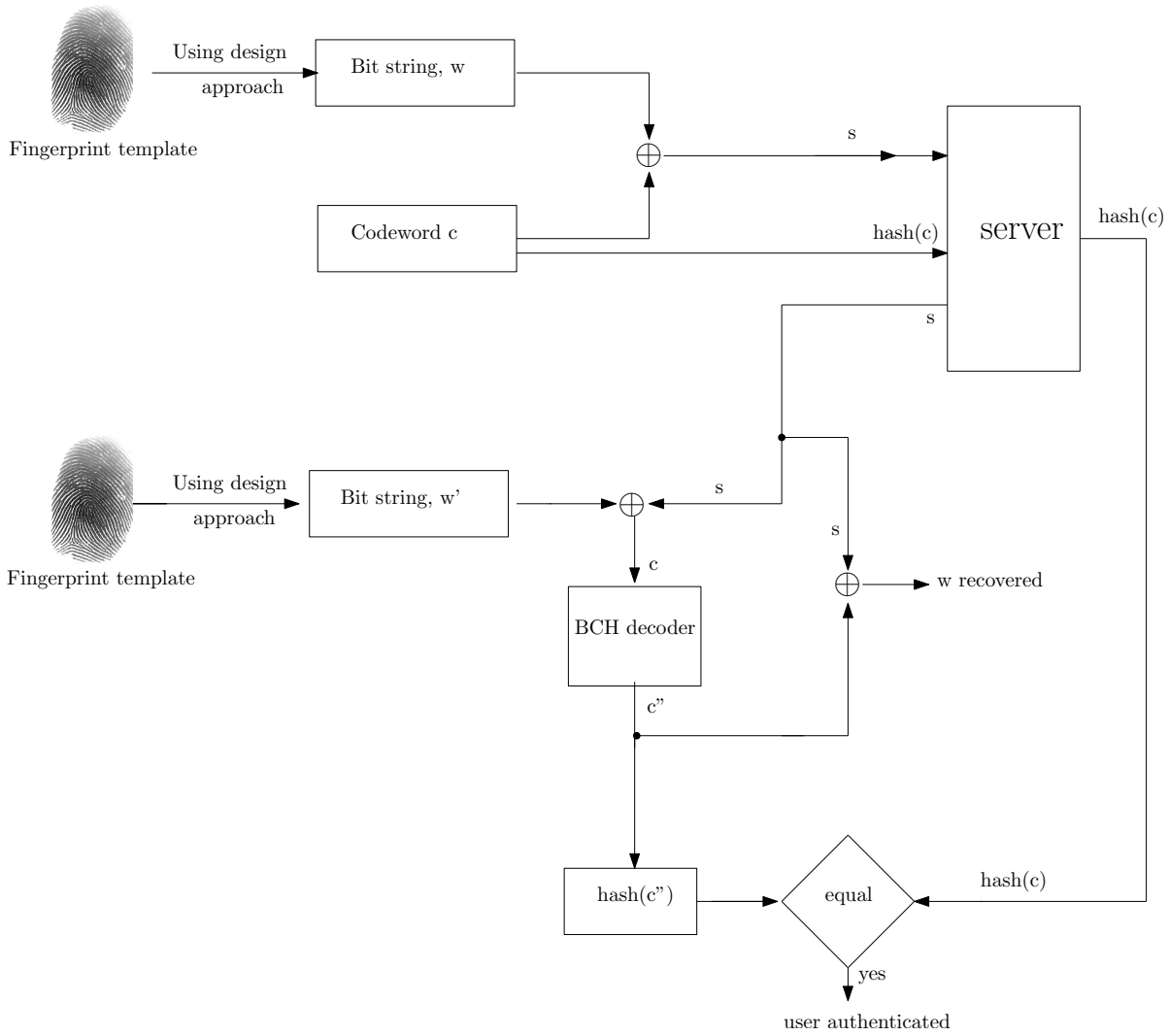

Figure 6.1: User authentication using output of our proposed design approach [3]

6.1.2 Authentication phase

In this phase, when a user provides $User\_id$ and w which is close to w such that $dis(w, w) \leq t$, where t are the number of errors that can be corrected in w so that for any $[n, k, 2t + 1]$ BCH code, we could recover w from w'.

From the server database, the value s stored corresponding to $User\_id$ will be retrieved and s is xored with w to get a value c.

$c = w \oplus s$
As w, w satisfies $dis(w, w) \leq t$, c is decoded using t-error-correcting BCH code to get the codeword c'' which is equal to codeword c. We then apply hash function on codeword c'' to obtain a value $hash(c'')$.
Now, if $hash(c)$=$hash(c'')$, the user is authenticated otherwise authentication fails, where $hash(c)$ is stored in database during users enrollment phase. Figure 6.1 shows the user authentication application.

The error correcting code BCH code is used for correcting errors in codeword obtained during authentication. In our implementation, we used [511,10] BCH code which can correct $t = 121$ errors in the codeword. We tested our scheme on 47 users. For each user, we store 2 templates in database during enrollment and test the third template during authentication.

The user authentication results are shown in Table 6.1. If the user is verified successfully, a secure key is generated as explained in Figure 6.2.

Table 6.1: User Authentication results

Verification rate	False positive rate	False negative rate
0.90	0.08	0.02

6.2 Key generation

For storing personal data on system or servers, a secure random key is required. Traditional keys and secret values generated are not perfectly uniformly random and are unreliable. Passwords such as long phrases are difficult to memorize. Also, every time the key or secret value must be exactly same in order to authenticate the person. Biometric data such as fingerprints, iris scan can be used to authenticate a user but the biometric data is noisy and non-uniform. Also, two readings of biometric data from same individual is not completely identical, even though it is close. Thus, there is a need of tolerating some errors in biometric data while preserving security. Fuzzy Extractor is used to extract strong, reliable, uniform random data from the less-uniform noisy biometric data. Fuzzy Extractors allow extraction of some randomness R which is highly uniform random and can serve as a key from input biometric data w and then successfully reproduce R from any other biometric data w that is close to w. The output of fuzzy extractor is strong key that is exactly same if the input data is sufficiently close to the input data during enrollment. [3]

After the user is authenticated, key is generated for the user. First we recover the value w for the authenticated user by simply xoring s with decoded codeword c'' as
$w = c'' \oplus s$
Next, we create the key k by taking hash of concatenation $(\|)$ of parameter w and x which is stored in database as

$k = hash\big(w \| x\big)$. Here $hash(n)$ is any secure one-way Cryptographic hash function which generates output hash from input n. Figure 6.2 illustrates the key generation procedure using Fuzzy extractor.

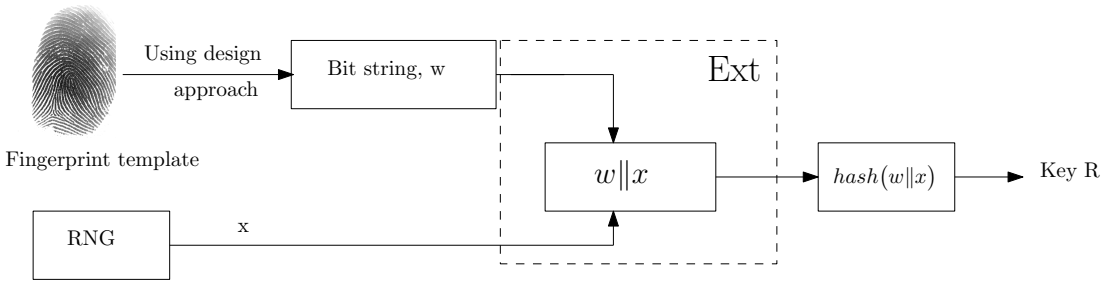

Figure 6.2: Key Generation using Fuzzy Exractor [3]

Figure 6.3 is the block diagram described by [3]showing protection of personal data stored on system using Strong key generated from Fingerprint template using our Representation System. User A wants to encrypt private data which is stored on Users system. Now using proposed scheme, a binary string w will be generated from users fingerprint template. The user then enrolls him on the system and using Fuzzy extractor, a strong random key R is generated. Now, user A using key R encrypts the data and store it on the system in encrypted form $E_R\big(DATA\big)$. The data can be stored on personal system or on network servers. So data is protected from unauthorized users [3]. Now, when the same user A wants to decrypt the data stored on system, using fingerprint template w which is close to the original fingerprint template but not exactly same, by authentication procedure, the original codeword c is generated which is then used to recover w from w. Now, key R generated will be exactly same as earlier. This key is used to decrypt data as $D_R\big(E_R\big(DATA\big)\big)$.

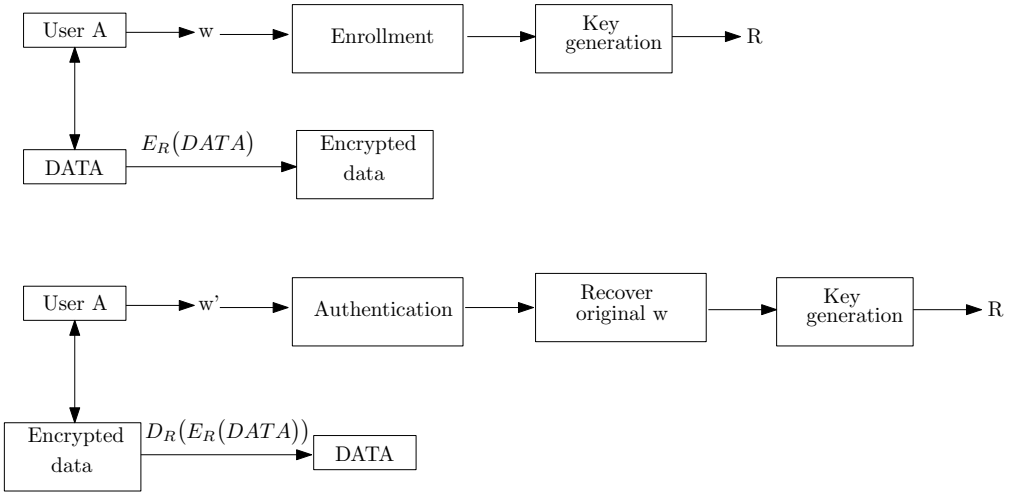

Figure 6.3: Protecting personal data using key generated from Fingerprint template [3]

Chapter 7

Conclusion and Future work

A new design approach has been introduced for representing Fingerprint templates. The Fingerprint template captured in digital format is represented in the form of binary string using our proposed approach. The output is highly random and it is impractical to recover original fingerprint template from binary output. Thus, attacker is not able to retrieve any information about fingerprint of an individual from the fuzzy hash output, thereby securing the template. We used the binary output for authenticating users. The results are accurate to a large extent. Our scheme thus, provides secure authentication as the original fingerprint templates no longer need to be stored on server for matching. Also, using the concept of Fuzzy Extractor and secure sketch, a strong random Cryptograpic key is generated from Fingerprint template which can be used to protect personal data.

In future, we will perform security analysis of our proposed representation scheme as discussed in Section 4.6. We will work on improving the accuracy of present results for user authentication. Besides the usage of secure authentication and key generation, we would like to apply this scheme to secure communication and other areas where biometric traits can be used for authentication purposes by extending our work on other biometric traits like iris, face etc.

Bibliography

[1] Centroid, wikipedia.

[2] Frank Breitinger. Security aspects of fuzzy hashing. Master's thesis, University of Applied Sciences Darmstadt, 2011.

[3] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[4] Yunghsiang S. Han. Bch codes.

[5] Lin Hong, Yifei Wan, and Anil K. Jain. Fingerprint image enhancement: Algorithm and performance evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):777–789, 1998.

[6] Anil K. Jain, Lin Hong, and Ruud M. Bolle. On-line fingerprint verification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4):302–314, 1997.

[7] Hamzeh Khazaei and Ali Mohades. Fingerprint matching algorithm based on voronoi diagram. In Marina L. Gavrilova, Osvaldo Gervasi, Antonio Laganà, Youngsong Mun, and Andrés Iglesias, editors, *Selected Papers of the Sixth International Conference on Computational Sciences and Its Applications, ICCSA '08, Perugia, Italy, June 30 - July 3, 2008*, pages 433–440. IEEE Computer Society, 2008.

[8] Adam Kirsch and Michael Mitzenmacher. Less hashing, same performance: Building a better bloom filter. *Random Struct. Algorithms*, 33(2):187–218, 2008.

[9] Jesse D. Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, 3(Supplement-1):91–97, 2006.

[10] Karthik Nandakumar, Anil K. Jain, and Sharath Pankanti. Fingerprint-based fuzzy vault: Implementation and performance. *IEEE Transactions on Information Forensics and Security*, 2(4):744–757, 2007.

[11] Vassil Roussev. Data fingerprinting with similarity digests. In Kam-Pui Chow and Sujeet Shenoi, editors, *Advances in Digital Forensics VI - Sixth IFIP WG 11.9 International Conference on Digital Forensics, Hong Kong, China, January 4-6, 2010, Revised Selected*

Papers, volume 337 of *IFIP Advances in Information and Communication Technology*, pages 207–226. Springer, 2010.

[12] Raymond Thai. Fingerprint image enhancement and minutiae extraction, 2003.

[13] Yangsheng Wang Weiwei Zhang. Core-based structure matching algorithm of fingerprint verification. *IEEE Trans. Pattern Recognization*, pages 70–74, 2002.

[14] Wencheng Yang, Jiankun Hu, and Song Wang. A delaunay triangle-based fuzzy extractor for fingerprint authentication. In Geyong Min, Yulei Wu, Lei (Chris) Liu, Xiaolong Jin, Stephen A. Jarvis, and Ahmed Yassin Al-Dubai, editors, *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2012, Liverpool, United Kingdom, June 25-27, 2012*, pages 66–70. IEEE Computer Society, 2012.

[15] Weiwei Zhang and Yangsheng Wang. Core-based structure matching algorithm of fingerprint verification. In *16th International Conference on Pattern Recognition, ICPR 2002, Quebec, Canada, August 11-15, 2002.*, pages 70–74, 2002.

Appendix A

Minutiae extraction from fingerprint template

The steps involved in the minutiae extraction are:

Step 1: Image enhancement [12]: It refers to removal of noise and perform sharpening of ridges in fingerprint template. The image enhancement step includes normalization of image using Histogram Equalization and Enhancement using Gabor filters.

- **Histogram Equalization:** It is a technique for adjusting image intensities i.e. pixel values to enhance overall image contrast. This allows areas of lower local contrast to gain a higher contrast by spreading out the most frequent intensity values. We used histogram equalization instead of standard normalization [12]that scales pixel intensity on the basis of minimum and maximum intensity pixel in the image to normalize the input image. But for more even distribution of intensity levels equalization is a better choice as it effectively spreads out the most frequent intensity values.

- **Enhancement using Gabor filtering [12]:** A two dimensional Gabor filter consists of a sinusoidal plane wave of a particular orientation and frequency, multiplied by a Gaussian function. A Gabor filter is defined a [5]

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

$$x_\theta = xcos\theta + ysin\theta$$

$$y_\theta = -xsin\theta + ycos\theta,$$

  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 are given by $\sigma_x = \sigma_y = 0.5F\big(i, j\big)$ [12] and x_θ and y_θ define the x and y- axis of the filter coordinate frame, respectively.
  There are two fundamental steps performed before applying gabor filter on image- Orienta-

tion field calculation and ridge frequency calculation. The orientation field of a fingerprint image defines the local orientation of the ridges contained in the fingerprint and is calculated by first dividing the image into blocks of size $W \times W$. Here, we use block size of 17×17 [12]. Then for each block, gradient vectors $\partial_x(i,j)$ and $\partial_y(i,j)$ are calculated by taking the partial derivatives of each pixel intensity in Cartesian coordinates. The local orientation at pixel $\big(i,j\big)$ can then be estimated using the following equations [5]:

$$G_{xy} = \sum_{h=-8}^{8} \sum_{k=-8}^{8} \partial_x\big(x_i + h, y_j + k\big)\partial_y\big(x_i + h, y_j + k\big),$$

$$G_{xx} = \sum_{h=-8}^{8} \sum_{k=-8}^{8} \partial_x\big(x_i + h, y_j + k\big)^2,$$

$$G_{yy} = \sum_{h=-8}^{8} \sum_{k=-8}^{8} \partial_y\big(x_i + h, y_j + k\big)^2,$$

$$\theta\big(i,j\big) = 90 + \frac{1}{2} tan^{-1} \frac{2G_{xy}}{G_{xx} - G_{yy}}$$

where $\theta\big(i,j\big)$ is the local orientation at the block of size 17×17 centered at pixel $\big(i,j\big)$. Gaussian smoothening using low pass filter is then performed as

$$\Phi'_x\big(i,j\big) = \sum_{u=\frac{-w_\Phi}{2}}^{\frac{w_\Phi}{2}} \sum_{v=\frac{-w_\Phi}{2}}^{\frac{w_\Phi}{2}} L\big(u,v\big)\Phi_x\big(i - uw, j - vw\big),$$

$$\Phi'_y\big(i,j\big) = \sum_{u=\frac{-w_\Phi}{2}}^{\frac{w_\Phi}{2}} \sum_{v=\frac{-w_\Phi}{2}}^{\frac{w_\Phi}{2}} L\big(u,v\big)\Phi_y\big(i - uw, j - vw\big),$$

Where L is Gaussian low pass filter of size $w_\Phi \times w_\Phi$ and the default size of the filter is 5×5 [5]. Here, Φ_x and Φ_y are the resultant x and y component of the vector field obtained after conversion of orientation image so as to perform low pass filtering on it.

$$\Phi_x\big(i,j\big) = cos\big(2\theta\big(i,j\big)\big),$$

$$\Phi_y\big(i,j\big) = sin\big(2\theta\big(i,j\big)\big),$$

Thus, The final smoothed orientation field O at pixel $\big(i,j\big)$ is defined as

$$O\big(i,j\big) = \frac{1}{2} \tan^{-1} \frac{\Phi'_y\big(i,j\big)}{\Phi'_x\big(i,j\big)}$$

After computing ridge orientation $O\big(i,j\big)$, ridge frequency estimation is done [5] [12]. The ridge frequency of an image represents the local frequency of the ridges in a fingerprint. Ridge frequency is the reciprocal of the inter ridge distance that shows the number of

ridges within a unit length of a image. This is calculated by dividing the image into blocks of size 16×16 and projecting the grey level values of all the pixels located inside each block along a direction orthogonal to the local ridge orientation. This projection forms sinusoidal-shape wave with the local minimum points representing the ridges in the fingerprint. 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 centered at pixel (i,j) is defined as

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

Once the ridge orientation and ridge frequency information has been determined, these parameters are used to construct the Gabor filter.

The Gabor filter is applied to the fingerprint image by spatially convolving the image with the filter. the enhanced image E can be given as [5]

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

where O is the orientation image, F is the ridge frequency image, N is the fingerprint image after Histogram Equalization step, and p_x and p_y are the width and height of the Gabor filter given by $p_x = 6\sigma_x$ and $p_y = 6\sigma_y$, respectively.

Step 2: Image cropping, Binarization and thinning [12]: Fingerprint core point can be defined as the point of maximum curvature in the fingerprint image. Core point can be find out by using several algorithms given in [15] [13]. Core point is used to align two fingerprint templates in fingerprint recognition techniques.

The enhanced image is cropped to a size 160×256 in which the core point is at the center of the image.

After cropping is done, image is converted to binary form using a process called Binarization that converts a grey level image into a binary image. This improves the contrast between the ridges and valleys which further help us in minutiae extraction. If the grey level value of image pixel is greater than the global threshold of 0 due to use of Gabor filter, then the pixel value is set to a binary value one. Else it is set to zero. Thus we get a binary image containing, the foreground ridges set to 1 and the background valleys set to 0.

Further processing of image includes thinning, a morphological operation that successively erodes away the foreground pixels until they are one pixel wide. In Matlab, this can be performed using thin function [12].

Step 3: Minutiae Extraction: The most commonly used method of minutiae extraction is the Crossing Number (CN) concept [12]. The fingerprint image is divided into blocks of 3×3 where each block represents a window with pixels P_1 to P_8. The minutiae are extracted by

scanning the local neighbourhood (eight pixels) of each ridge pixel in the image window.

The CN value is calculated as half the sum of the differences between pairs of adjacent pixels in the eight-neighbourhood [12].

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

The ridge pixel can then be classified as a ridge ending, bifurcation by using property of CN value as

- if CN=1, minutiae extracted is ridge ending
- if CN=3, minutiae extracted is ridge bifurcation

Thus every fingerprint template can be represented as a set of minutiae present on its fingertip.